Title
Synthetic Data Generation for Fraud Detection

Author
Shan, Jonathan

Publication Date
2023

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Synthetic Data Generation for Fraud Detection

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics and Data Science

by

Jonathan Shan

2023

ABSTRACT OF THE THESIS

Synthetic Data Generation for Fraud Detection

by

Jonathan Shan
Master of Science in Statistics and Data Science
University of California, Los Angeles, 2023
Professor Ying Nian Wu, Chair

This paper applies various synthetic data generation techniques to create synthetic fraud data for buy now, pay later (BNPL) financial institutions that mimic the statistical properties of real data. We utilize both statistical and deep learning methods to accomplish this task, contrasting each different framework's respective qualities. We evaluate the efficacy of our approaches by using our generated data to enhance the training sets of a fraud detection model and analyze the effects on validation results. Our results show that including synthetic data in existing datasets can improve the accuracy of fraud detection systems.

The thesis of Jonathan Shan is approved.

Guido Montufar

Guang Cheng

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

Buy now, pay later (BNPL) is a short-term loan option that serves as an alternative to credit cards. BNPL methods are similar to credit cards in that they both allow customers to make purchases and delay payment until a later time. However, BNPL firms distinguish themselves from credit card companies by requiring repayment in a series of installments, without needing a credit line that is repeatedly paid off. This allows shoppers to make purchases and spread the cost over time, generally spanning anywhere from two to eight weeks. BNPL utilizes credit checks that are not as strict as those used for traditional credit loans, allowing customers with poor or no credit history to qualify for loans. Moreover, BNPL companies collaborate with merchants to provide customers with a comprehensive and flexible payment option, which assists merchants in attracting more business.

In recent years, the usage of BNPL to pay for online purchases, and more recently, in-store purchases, has increased significantly. A Consumer Financial Protection Bureau study on the top five BNPL firms reported that the number of BNPL purchases grew from 16.8 to 180 million, while the total dollar amount lent grew from $2 billion to $24.2 billion [2]

With BNPL becoming increasingly prevalent, there is a growing need for effective systems that can analyze large volumes of data and identify fraudulent patterns. When a BNPL order is placed, relevant data is generated and utilized as input for machine learning models aimed at fraud detection. These models assess the potential risk associated with the order being fraudulent by considering various input variables pertaining to the order and returning a fraud score to rank order the transactions by riskiness. These variables can encompass

customer-related information, order details, or data obtained from external fraud detection agencies. Several examples of fraud predictors encompass the count of phone numbers linked to the customer's account, the monetary value of the order, or an identity verification score.

However, as fraud detection procedures improve, the quantity of legitimate transactions begins to heavily outnumber the number of fraudulent transactions. This introduces a strong class imbalance issue when designing machine learning models for fraud detection. Synthetic data is a promising solution to this problem, as it allows for the creation of synthetic orders that can be used to enhance existing tabular data. Additionally, similar studies have shown that the inclusion of synthetic fraud data can lead to improvements in model performance [3].

The purpose of this thesis is to investigate the viability of synthetic data generation methods for fraud detection in BNPL systems. Models explored in this thesis include Bayesian Networks, Gaussian Copulas, Variational Autoencoders (VAE), and Conditional Tabular Generative Adversarial Networks (CTGAN). We will explore the efficacy of the different data generation methods and their interactions with variable types. Each method will be evaluated by how well the synthetic data matches the real data, as well as how much the inclusion of the synthetic data improves the our baseline model's performance.

CHAPTER 2

Methodology

2.1 Overview

Our paper investigates whether adding a tabular synthetic data generation step to a machine learning model development process could improve fraud detection metrics. The proposed development framework is shown below.

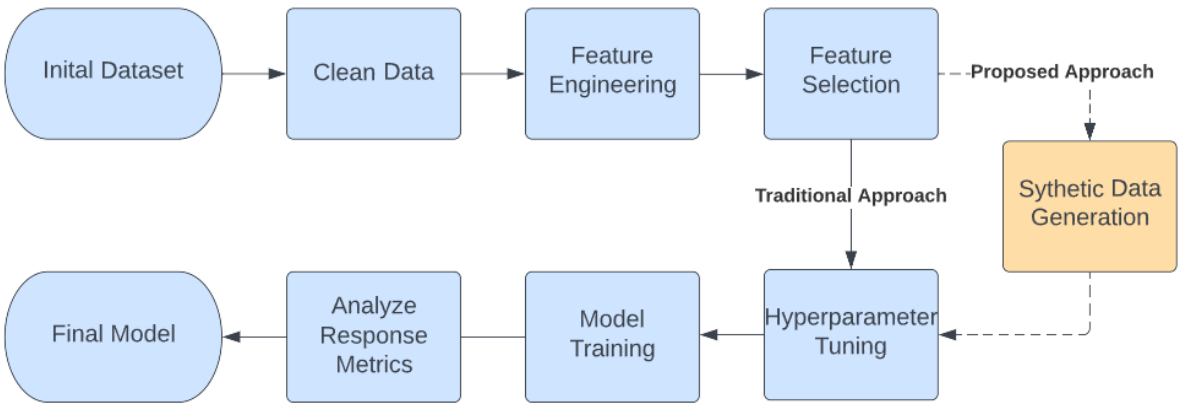

Figure 2.1: Proposed Framework

Using the fraudulent data from an imbalanced BNPL transaction dataset, we will proceed to generate additional instances of fraudulent orders. These synthetic fraudulent transactions will be generated employing the various synthesizers discussed earlier in this study. Subsequently, we will assess the impact of incorporating synthetic data on the performance

of a model designed to predict fraud within the aforementioned BNPL imbalanced dataset. To achieve this, we will train new models using training data enriched with varying proportions of synthetic fraudulent data. This approach not only allows for an investigation into the effectiveness of different synthesizers, but also enables the determination of the optimal quantity of synthetic data required to enhance model performance.

The following sections will describe the individual methods used to generate data, as well as the metrics used to determine quality of synthetic data and predictive performance.

2.2 Synthetic Data Generation Methods

2.2.1 Bayesian Network

Bayesian networks (BN) are probabilistic graphical models used to represent a set of variables and their conditional dependencies through a directed acyclic graph (DAG). BNs have become increasingly common in the field of biology, where they have been widely applied, with gene regulatory networks being a particularly popular area of research [6] [12] [20]. More recently, BNs have been used for data imputation or to generate and anonymize synthetic data [14] [19] .

Each vertex in the DAG corresponds to a variable and each edge represents a conditional dependency between the variables. The key feature of Bayesian Networks is that they allow for the representation of the joint probability distribution of a set of variables in terms of their conditional probabilities.

Formally, a Bayesian Network for random variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ is defined as a DAG $G = (V, E)$ where V is the set of vertices representing $\mathbf{X}$, and E is the set of edges representing the conditional dependencies between $\mathbf{X}$. For each vertex in set V, there exists a conditional probability for a random variable X_i that can be defined as $P(X_i|P(E_i))$ where $P(E_i)$ is the probability of all vertices with a directed edge pointed towards X_i.

The joint probability distribution of $\mathbf{X}$ that is used to generate synthetic data can then be represented as:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|P(E_i))$$

We will use the DataSynthesizer [14] package to build a generative Bayesian Network in Python. One key setting for the DataSynthesizer Bayesian Network is the "mode" of the model with three possible options: differential privacy, independent attributes, and correlated attributes. For the purposes of this paper, we will be using the correlated attributes setting as there are variables in our dataset that are the same metric measured over different periods of time (e.g. number of payments made in the last day/week/month).

2.2.2 Gaussian Copula

Copulas are a statistical tool that allows the modeling of the dependence structure between random variables separately from their marginal distributions. The Gaussian Copula, rather infamously [9], has been widely used in finance and risk management, particularly in the modeling of credit risk and the pricing of credit derivatives.

The Gaussian copula is defined as the copula function associated with a multivariate normal distribution with zero means and a correlation matrix that represents the dependence structure between the variables. The copula function of a Gaussian Copula is given by the cumulative distribution function of a multivariate normal distribution with standard normal marginals.

Formally, let Φ be the the cumulative distribution function of a standard normal distribution and $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be a vector of random variables with $F_1, F_2, \ldots, F_n$ as their marginal distributions. Then, the Gaussian Copula C, which represents the dependence structure of $\mathbf{X}$, is defined as:

$$C(F_1, F_2, \ldots, F_n) = \Phi(\Phi^{-1}(F_1(X_1)), \Phi^{-1}(F_2(X_2)), ..., \Phi^{-1}(F_n(X_n)); \Sigma)$$

The resulting copula C takes the joint distribution of the variables and maps it to a multivariate Gaussian distribution with a correlation matrix Σ.

For this study, we used the Gaussian Copula Synthesizer implementation provided by the Synthetic Data Vault (SDV) package [13]. Their implementation follows the copula function shown above to model the data, and uses the Σ matrix as the generative model.

2.2.3 Tabular Variational Autoencoder

A variational autoencoder (VAE) combines the statistical techniques from Bayesian variational inference with the framework of machine learning based autoencoders to approximate distributions and generate data. As is typical for autoencoder structures, VAEs are comprised of two halves, an encoder and a decoder. The encoder is responsible for mapping the set of input variables to a latent variable space. From there, the decoder tries to map the latent variable space back into a variable space that is as similar as possible to the original variable space. In most applications, VAEs use neural networks as the encoder and decoders.

The fundamental distinction between a VAE and other autoencoders lies in the objective they aim to achieve. For traditional autoencoders, the primary goal is to create an efficient representation of the original variable set, similar to using Principal Component Analysis for variable reduction. In contrast, VAEs aim to learn the parameters of an encoder and decoder such that new data that resembles the original data can be generated. Additionally, the encoder in a VAE aims to model the input variables to a low dimensional latent space that is a combination of a predetermined distribution.

Formally, an encoder E_ϕ with parameters ϕ maps a given a set of variables x to a latent space z. This latent space z is assumed to follow a prior distribution $p(z)$ that is most commonly the standard Gaussian distribution. This encoder represents the posterior latent

distribution $q_\phi(z|x)$, which approximates the true posterior latent distribution with parameters θ, $p_\theta(z|x)$. Data is sampled from $q_\phi(z|x)$ and passed to the decoder which then maps the sample to x', a reconstructed representation of the original variable space. This mapped sample is denoted as $D_\theta(z)$ with D_θ representing the decoder. This decoder represents the conditional likelihood distribution $p_\theta(x|z)$.

The VAE aims to learn the optimal parameters ϕ of the encoder and θ of the decoder. This objective is accomplished by minimizing the VAE loss function, which is comprised of a reconstruction term and a regularization term. The reconstruction term is applied at the output layer of the decoder to minimize the distance between x and x'. This part of the loss function ensures that the generated data follows a distribution similar to that of the input data. The regularization term, otherwise known as the Kullback-Leibler divergence (KL), encourages the latent space to follow its assumed prior distribution. This helps the encoder shape the latent space into a structured and desirable distribution. With regularization, the encoder is encouraged to map data in a tight distribution across the latent space preventing the model from overfitting and allowing for smooth transitions and continuity between different latent points.

Example VAE loss function with standard Gaussian prior distribution:

$$\text{Loss} = \underbrace{||x - D_\theta(z)||^2}_{\text{reconstruction}} + \underbrace{KL[N(\mu_x, \sigma_x)\,||\,N(0, I)]}_{\text{regularization}}$$

Although VAEs are often used for tasks such as image generation, SDV has adapted a VAE for tabular data generation called the Tabular Variational Autoencoder (TVAE) Synthesizer [16]. In this implementation, both E_ϕ and D_θ components are fully connected neural networks. Additionally, $p_\theta(z|x)$ and $q_\phi(z|x)$ are assumed to follow Gaussian distributions while $p(z)$ is assumed to be a standard Gaussian distribution. TVAE employs the Evidence Lower Bound (ELBO) as an objective function to optimize parameters ϕ and θ of the encoder and

decoder, respectively.

$$ELBO = \ln p_\theta(x|z) - KL[q_\phi(z|x) \,||\, p(z)]$$

The ELBO tries to maximize the log-likelihood of the generated data while also trying to minimize the divergence between the posterior latent distribution and the standard Gaussian distribution. The log-likelihood is back-propagated through the network to update the model weights.

Before inputting the data into the TVAE, a preprocessing step is performed to ensure proper data representation. A technique named mode normalization [16] is used to transform the continuous variables. Mode normalization utilizes Variational Gaussian Mixture Models [11], allowing each variable distribution to be expressed as a combination of multiple Gaussian distributions centered around the modes of the original data.

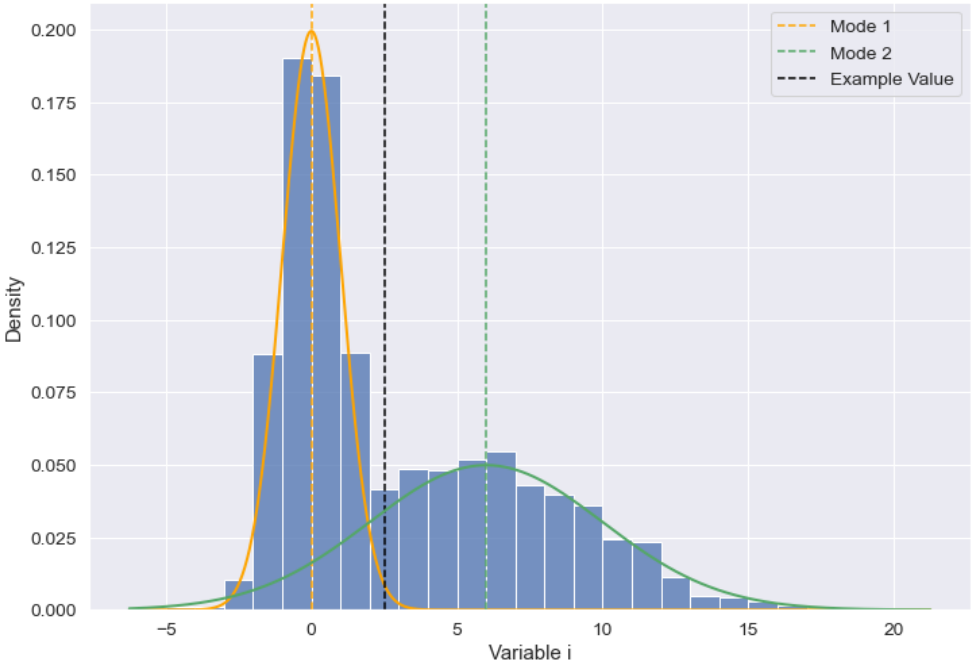

Figure 2.2: Mode Normalization Example

For example, in Figure (change), the distribution of variable i is modelled as a combination of two Gaussian distributions. According to the Gaussian Mixture Model, the example value

is more likely to originated from the second Gaussian, thus that distribution is used to normalize the value. In addition, discrete variables are encoded into one-hot vectors to facilitate their handling within the framework. The final representation of a row j of the data is

$$r_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus \ldots \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus d_{1,j} \oplus \ldots \oplus d_{N_d,j}$$

with $\alpha_{1,j}$ as the normalized value, $\beta_{1,j}$ as the mode used normalize, $d_{i,j}$ as the one-hot vector, N_c as the number of continuous variables, a N_d as the number of discrete variables.

The TVAE structure for a given a row j of the input data can be seen in Figure (change). The encoder and decoder are both fully connected neural networks that feature two ReLU layers leading to multiple output layers. The encoder network outputs both a mean vector μ and a variance vector σ. These resultant vectors are then used to model the latent space, which is assumed to follow the distribution $\mathcal{N}(\mu, \sigma I)$. Samples z_j are drawn from the latent distribution and passed to the decoder, which outputs $\bar{a}_{i,j}$, $\hat{\beta}_{i,j}$, and $\hat{d}_{i,j}$ for each variable i in row j. $\hat{a}_{i,j} \sim \mathcal{N}(\bar{a}_{i,j}, \delta_i)$ is modelled from the output data, with δ_i being a parameter learned by the decoder. Finally, a joint distribution approximating the distribution of the input data is created:

$$p_\theta(r_j|z_j) = \prod_{i=1}^{N_c} P(\hat{\alpha}_{i,j} = \alpha_{i,j}) \prod_{i=1}^{N_c} P(\hat{\beta}_{i,j} = \beta_{i,j}) \prod_{i=1}^{N_d} P(\hat{d}_{i,j} = d_{i,j})$$

Synthetic data is then generated by sampling from this joint distribution and transforming the mode regularized data back into its original representation. For our implementation, we trained our TVAE for 100 epochs.

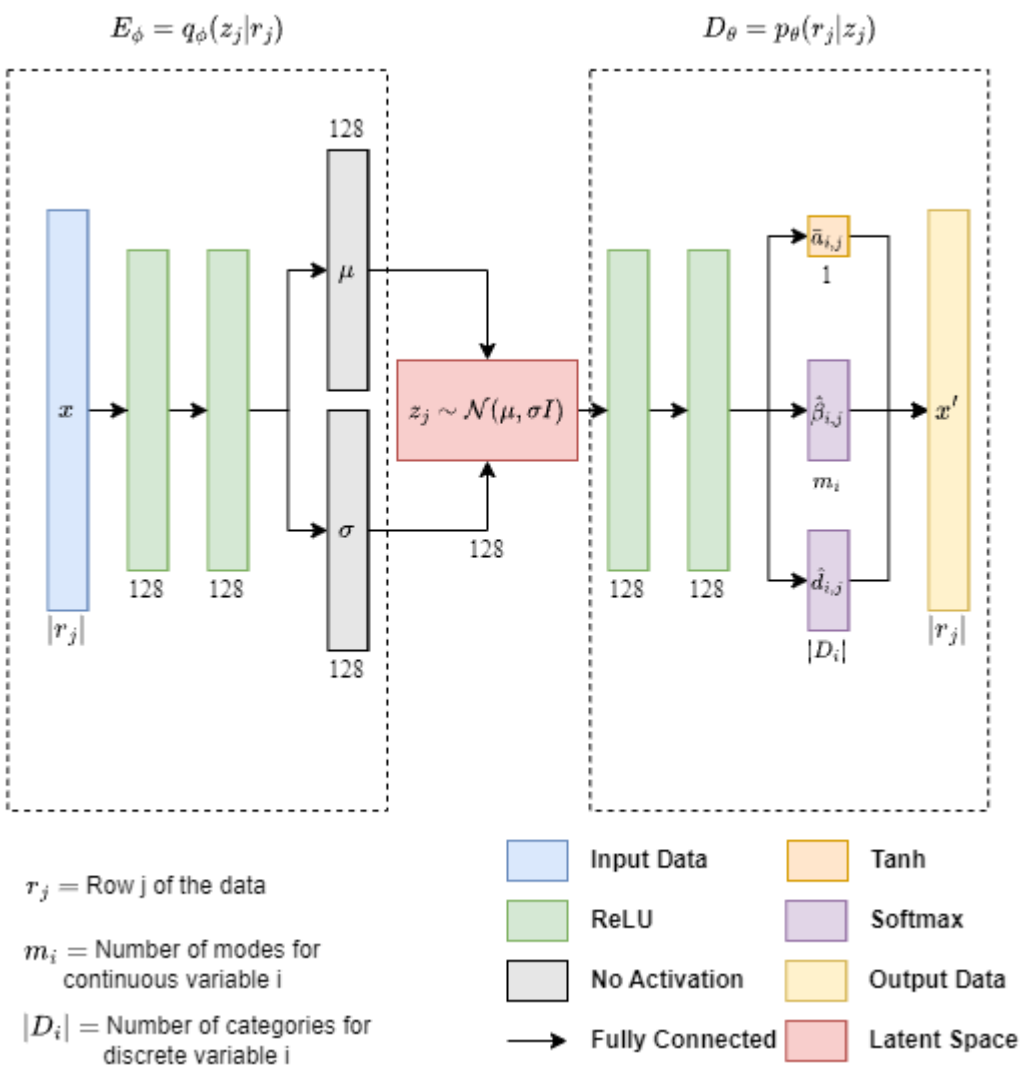

Figure 2.3: SDV TVAE Architecture

2.2.4 Conditional Tabular Generative Adversarial Networks

Introduced in 2014, Generative Adversarial Networks (GANs) emerged as a powerful generative framework primarily aimed at image generation [1] [7] [18]. Over the recent years, GANs have gained significant traction and witnessed a surge in popularity, leading to the development of diverse variations tailored to address specific problem domains [4] [17]. The fundamental architecture of an adversarial network consists of two opposing models that are pitted against each other and trained simultaneously. The generative model G generates syn-

thetic data, while the discriminative model D attempts to differentiate generated data from real data. GANs distinguishes themselves from other frameworks through the generator's loss function. Rather than generate data that aims to follow a similar distribution to the original data, G creates data that maximizes the possibility of tricking D into thinking the data is real. This process allow GANs to offer the distinct advantage of not on any inference during the learning process or having to using Markov chain Monte Carlo methods for data sampling. When both G and D are both neural networks, the system can be optimized using forward and back propagation.

Formally, a GAN consists of:

1. $G(z, \theta_g)$: A multilayer perceptron generator with parameters θ_g that takes samples random data from a prior noise distribution $p_z(z)$ and outputs x.

2. $D(x, \theta_d)$: A multilayer perceptron discriminator with parameters θ_d that takes data x and returns $D(x)$, the probability that x comes from the original data and not the generator.

During training, we employ a joint optimization process for both the generator network (G) and the discriminator network (D). The objective is to optimize the generator's ability to fool the discriminator by minimizing $\log(1 - D(G(z)))$, while simultaneously improving the discriminator's accuracy in distinguishing between generated and original data samples. The objective function $V(G, D)$ of the GAN can be written as:

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

The variation of GAN used in this paper is the Conditional Tabular GAN Sythesizer offered by the SDV package [16]. As traditional GANs are geared towards image generation, they can struggle to model certain aspects of mixed-type tabular data [16]. The CTGAN implementation improves upon traditional GANs by addressing these common issues.

In traditional GANs, the same activation function is typically used for both continuous

and discrete variables when generating tabular data. CTGAN addresses this limitation by applying the *tanh* activation function specifically to the continuous variables, while adopting the more appropriate Gumbel-Softmax [5] activation for the discrete variables. Previous research has also demonstrated that GANs are susceptible to a phenomenon known as mode collapse, which refers to the GAN's limited ability to model multimodal distributions [15]. Mode collapse signifies that the generator fails to capture the full complexity of the underlying data distribution, often by reducing a multimodal distribution to a Gaussian-like distribution. To address this, CTGAN uses the same mode normalization preprocessing as described in the TVAE section.

Another area that GANs struggle with is mode collapse in highly imbalanced categorical columns. During training, underrepresented classes can be incidentally omitted when resampling, leading to a lack of learning opportunities for minority classes. To tackle this issue, a solution involves enabling the generator to generate synthetic data conditioned on a discrete variable and then sampling data during the training phase based on the logarithmic frequency of each category. This novel approach, dubbed "training-by-sampling", ensures that the generator explores all possible discrete values.

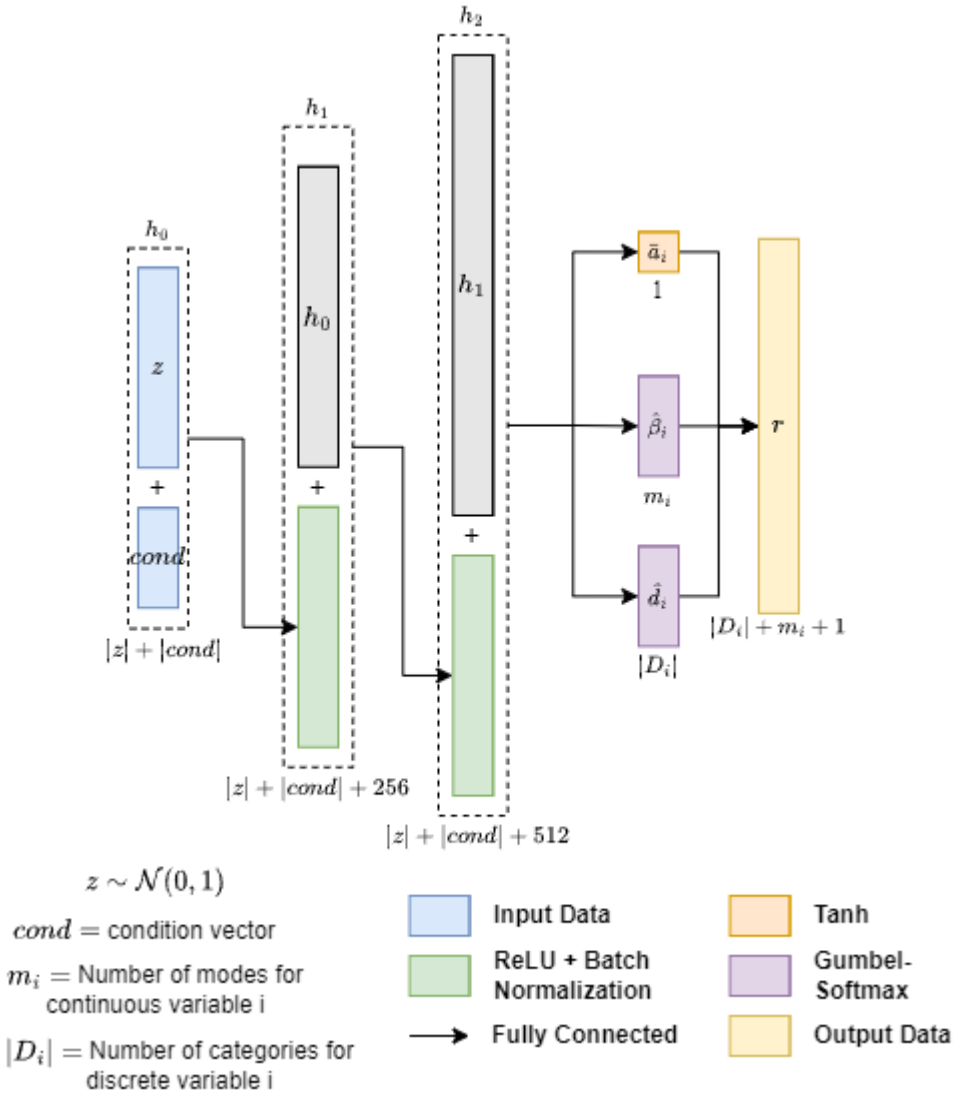

Figure 2.4: SDV CTGAN Generator Architecture

In the CTGAN implementation, the generator is capable of generating synthetic data that is conditioned on a discrete column. This conditioning is represented by a vector called *cond*. To construct the condition vector *cond*, the mask vectors for each discrete variable are concatenated. The mask vector m for a discrete variable D_i is its a one-hot vector if D_i is the variable being conditioned on, and a zero vector otherwise. For example, if there are two discrete variables D_1 and D_2 with possible values $\{1, 2, 3\}$ and the condition is $D_2 = 1$, then the mask vectors are $m_1 = [0, 0, 0]$ and $m_2 = [0, 1, 0]$ and the conditional vector is then

$cond = [0, 0, 0, 0, 1, 0]$.

The neural network structure of SDV CTGAN follows a architecture resembling their TVAE implementation. Both the generator G and discriminator D consist of two fully connected hidden layers that lead to output layers. Similar to the TVAE decoder, the generator G produces outputs scalar values for the continuous variables $\hat{a}_{i,j}$, mode indicators of the continuous variables $\hat{\beta}_{i,j}$, and one-hot vectors for the discrete variables $\hat{d}_{i,j}$. Finally, these generated rows r are fed into the discriminator for classification.

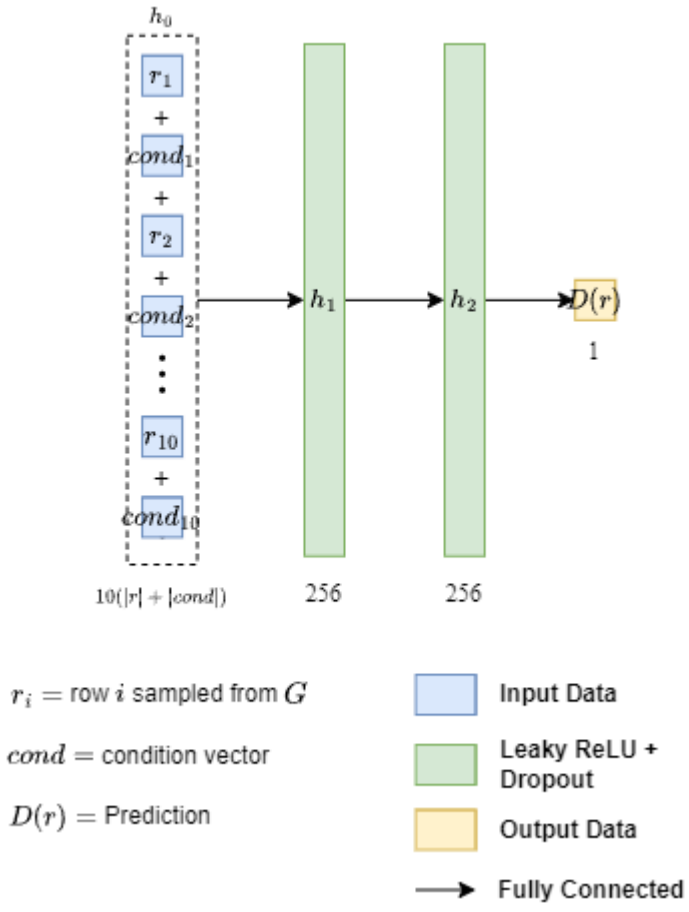

Figure 2.5: SDV CTGAN Discriminator Architecture

The discriminator show in Figure 2.5 uses a PacGAN architecture [8] with a pac size of 10 samples. PacGANs takes in multiple samples at the same time and evaluates the data's legitimacy as a single entity rather than individually. This helps the discriminator prevent

model collapse during training. Our final CTGAN was trained for 50 epochs.

2.3 Evaluation Metrics

2.3.1 Fraud Model Performance

In order to assess the impact of synthetic data on the model's performance, we will conduct experiments involving the inclusion of varying proportions of synthetic data in the model's training set. Subsequently, we will analyze the resultant changes in the model's out-of-time performance, specifically focusing on two widely used evaluation metrics for binary classification tasks: Precision-Recall Area Under Curve (PR AUC) and Receiver Operating Characteristic Area Under Curve (ROC AUC). These metrics provide valuable insights into the model's ability to correctly classify positive (fraud) and negative (non-fraud) instances, thereby aiding in the evaluation of its overall performance.

PR AUC assesses the quality of a binary classification model by measuring the area under the precision-recall curve.

ROC AUC assesses the discriminative power of a binary classifier by computing the area under the Receiver Operating Characteristic (ROC) curve.

Both PR AUC and ROC AUC are robust evaluation metrics that are insensitive to class imbalance and suitable for imbalanced datasets. In our experiments, we will use the increase in PR AUC and ROC AUC to evaluate the efficacy of our synthetic data generation.

2.3.2 Synthetic Data Quality

Synthetic Data Metrics (sdmetrics) is a Python-based library crafted by the creators of the SDV, DataCebo. This library serves the purpose of producing measurements and visual representations that aid in evaluating the quality of our artificially generated data. We will

examine a couple statistics from sdmetrics to compare our synthetic data to the real data: column shape similarity and column pair trends. These metrics assist us in measuring the extent to which our synthetic data aligns with the statistical attributes inherent to our genuine data.

Column Shape Similarity pertains to the similarity between synthetic data and real data averaged over all of the columns in the dataset. Similarity between distributions is measured using the Kolmogorov-Smirnov statistic [10] for numeric columns and the Total Variation Distance for categorical and boolean columns.

Column Pair Trends measures how similarly the correlation of two columns is between the real and synthetic data. The final statistic is the average difference in Pearson Correlation across all pairs of variables.

2.3.3 Synthetic Data Coverage

Another facet of generating synthetic data is ensuring that the generated data approximates the minimum-maximum value ranges for each variable in the data. We examine the Diagnostic Report from sdmetrics to help us verify if synthetic data is representative of the entire real data sample. This report provides us with the column coverage metric.

Column Coverage gauges how well the synthetic data covers the range of values from the real data for a given column. The metric is calculated differently depending on the variable type. For numeric columns, this metric measures how close the minimum and maximum values of the synthetic data are to those of the real data. For categorical and boolean variables, the coverage is measured by the proportion of unique categories represented in the synthetic data. This metric has a range of $[0, 1]$ where 0 represents no coverage while 1 represents perfect coverage. The final metric is the average across all columns.

CHAPTER 3

Results

3.1 Initial Dataset

The original dataset contains 46 predictors along with 9,300,266 BNPL orders. Among these orders, 8,246 (0.09%) are identified as fraudulent transactions. To protect privacy, the names of the predictor variables have been anonymized. The dataset is partitioned into three distinct subsets: training, testing, and out-of-time (OOT), with the data distribution outlined in Table 3.1. The 8,246 fraudulent orders in the dataset serve as the real data from which we generate synthetic fraudulent data.

Dataset	# Orders	# Fraud Orders
Train	5,797,166	5,521 (0.095%)
Test	1,449,291	1,349 (0.093%)
OOT	2,053,809	1,376 (0.067%)

Table 3.1: Dataset Split

The dataset includes numeric, categorical, and boolean predictors. Among the numeric predictors, there are predictors that are inherently correlated with each other. For example, one numeric predictor could be the number of orders a customer has placed over the last hour, while the another predictor could be the number of orders over the last day. To successfully generate data that emulates the original data, these relationships between variables much remain consistent, with no logical contradictions (e.g. more orders in last hour than last day).

The categorical variables in the dataset were encoded using ordinal encoding, the default

method for LightGBM frameworks. Compared to one-hot encoding, where each category becomes a binary feature, ordinal encoding can help reduce the dimensionality of the data. This is especially important when generating synthetic data, as it can simplify the complexity of the generated dataset and reduce the number of interactions between variables that the synthesizer must learn.

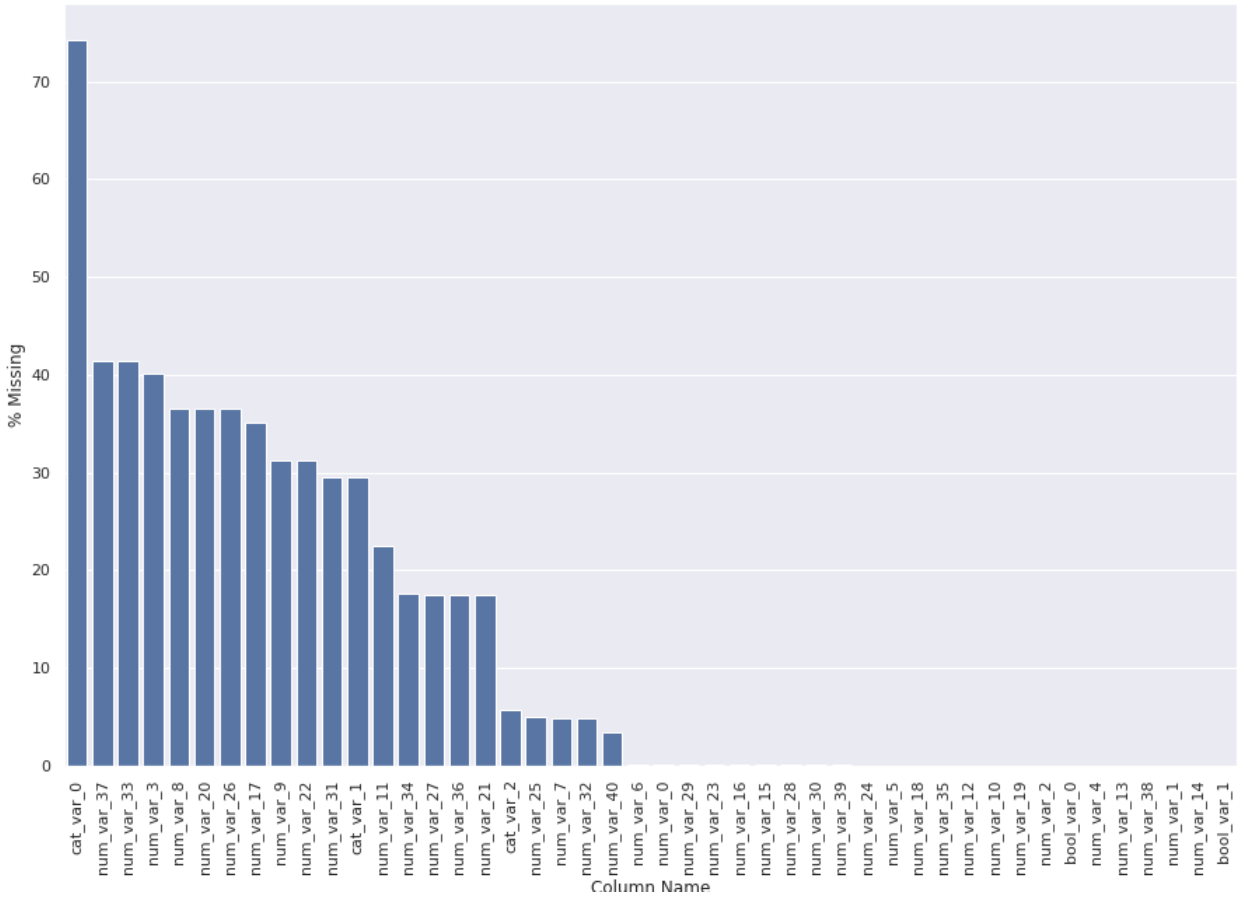

Figure 3.1: Dataset Missing Rates

Furthermore, it's worth noting that the dataset exhibits a significant amount of missing data, with missing rates peaking at 75%, as illustrated in Figure 3.1. No imputation is done to the missing values as the LightGBM framework inherently addresses missing data by ignoring them when making a new split. Subsequently, these missing values are allocated to the side of the split that minimizes the overall loss, effectively treating missing data as its own unique

value. The proportions of missing values existing in each column is another quality of the data the synthesizers must learn in order to successfully recreate the data.

3.2 Bayesian Network

In this project, we experimented with two Bayesian Networks, the first featured one incoming edge per node (degree=1) and the second permitting two incoming edges per node (degree=2). This configuration provided the second Bayesian Network with the capability to represent more intricate relationships between nodes, as variables could be directly associated with two other variables. Although, this comes at the cost of increased complexity of the DAG, and subsequently, the model, as well as an increased risk of overfitting. The first model, with a degree of one, presented a simpler alternative. However, it may not capture all the nuanced patterns present in the original data.

Model	Column Shapes	Column Pair Trends	Coverage
BN(d=1)	0.7217	0.9685	0.9311
BN(d=2)	0.7212	0.9740	0.9784

Table 3.2: Bayesian Network Comparison

The findings presented in Table 3.2 indicate that the Bayesian Network with a degree of two only exhibits slight enhancements in terms of the data's shape and correlations compared to the more straightforward model. Additionally, it experiences a regression in terms of data coverage when compared to the first model. Consequently, we have decided to proceed by incorporating the data generated by the Bayesian Network with a degree of one into the original dataset. This choice was made because the increased complexity introduced by the second model did not yield significant improvements. The directed acyclic graph of the Bayesian Network (d=1) can be seen in Figure 3.2.

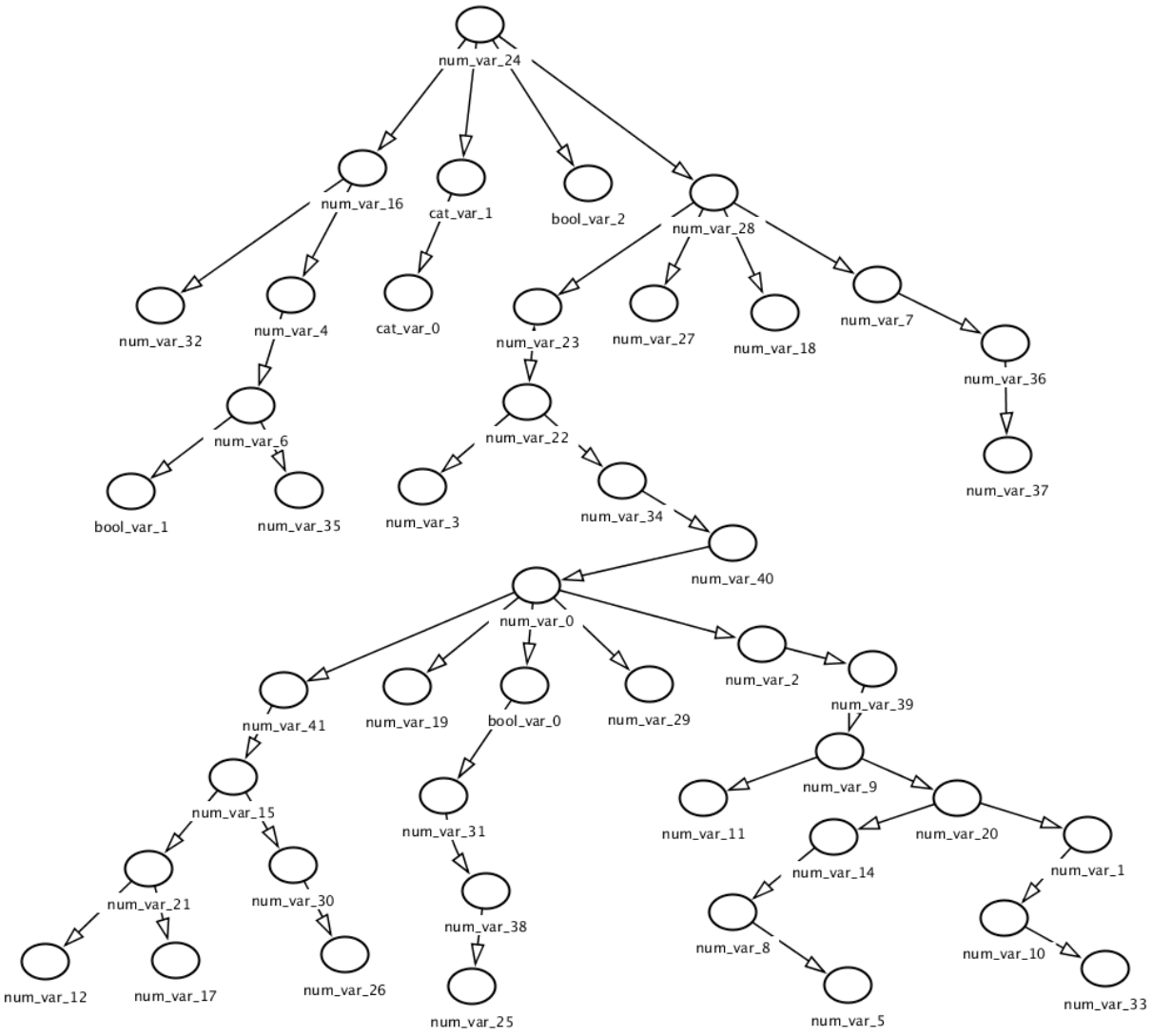

Figure 3.2: Bayesian Network DAG

When examining the results of the Bayesian Network, it is clear that the model can struggle maintaining the correct distributions for certain numeric variables. Figure 3.3 displays the column shape scores by variable. Although the boolean and categorical variables seem to be well fit, the numeric variables score as low as about 0.2, indicating a lack of fit for certain columns.

The Bayesian Network often struggles when trying to fit numeric variables, specifically those with a large maximum and a majority of their values tightly clustered within a narrow range. The distributions for the variable exhibiting the worst fit, num_var_5, is displayed in Figure

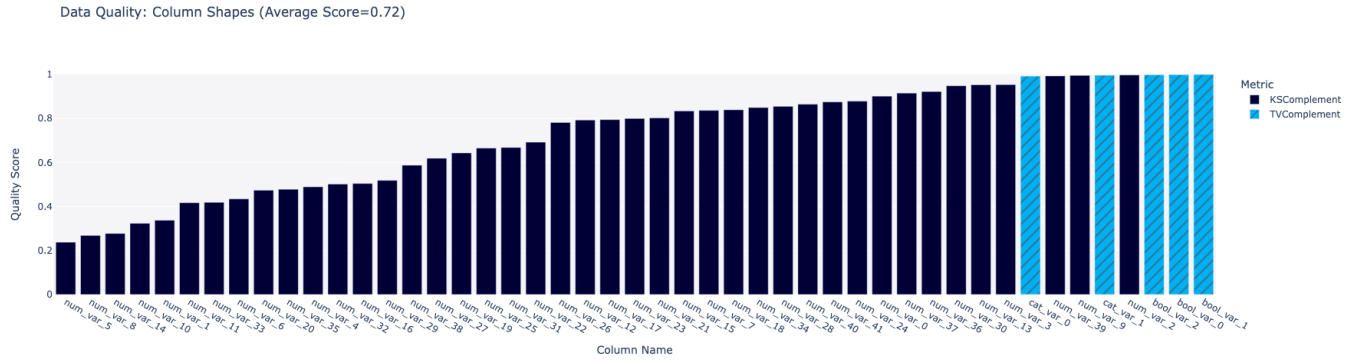

Figure 3.3: Bayesian Network Column Shape Scores

3.4. The actual distribution of this variable is highly centered around 0, with approximately 98% of its values falling between 0 and 10. However, it is worth noting that the maximum value in this distribution is significantly distant at 385.

In comparison, the synthetic data produced by the Bayesian Network scatters this concentration, with only around 53% of its values falling below 10. It is important to highlight that the Bayesian Network consistently encounters difficulties when dealing with numeric variables that exhibit a distribution pattern akin to that of num_var_5, characterized by a concentration around 0 and a large maximum value.

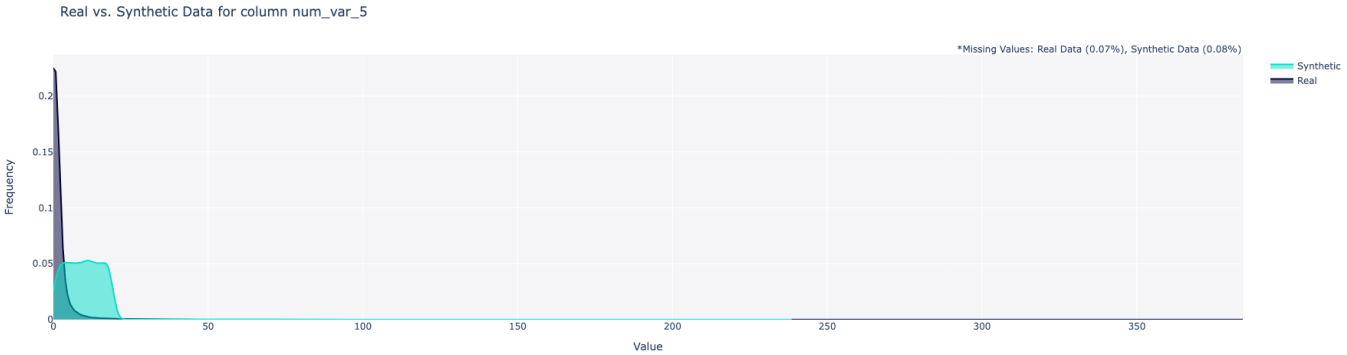

Figure 3.4: Bayesian Network num_var_5 Distribution

The Bayesian Network is not without its merits. For the given data set, the model does

a perfect job at creating synthetic categorical and boolean data. Numeric columns with maximum values close to their means also attained high column shape scores.

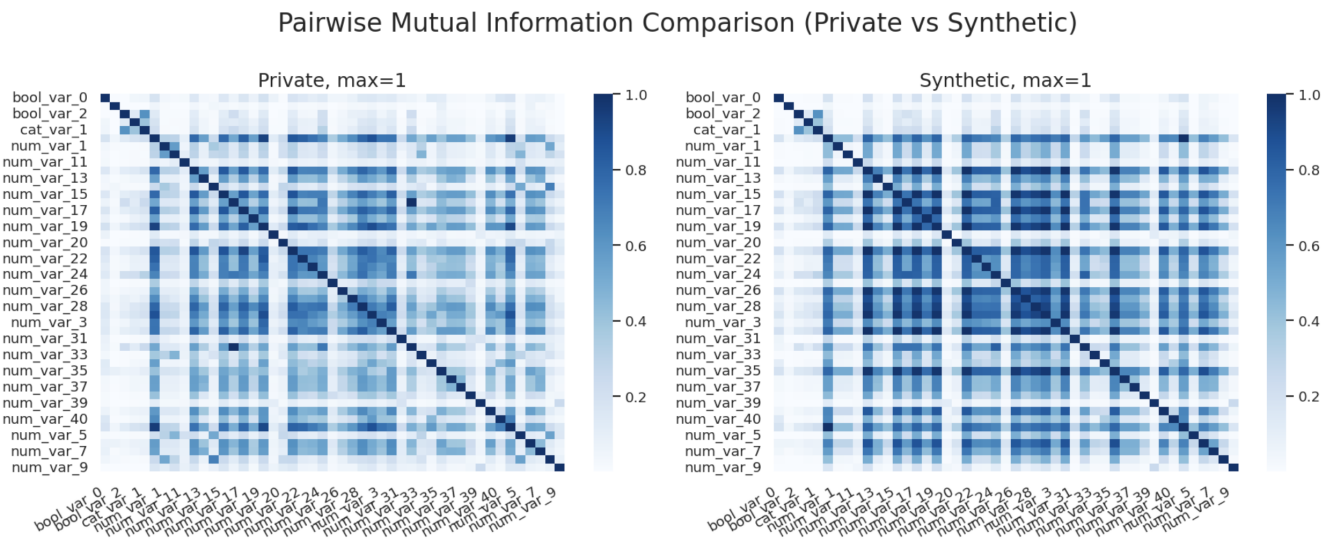

Figure 3.5: Bayesian Network Correlation Heatmap

The column pair trends score of 0.9685 displays the greatest strength that the Bayesian Network, its ability to model correlations between variables. The DAG structure of a BN offers a valuable framework for understanding the dependencies within a dataset, enabling anyone analyzing the data to visualize potential dependencies between variables. The correlations between each pair of features in Figure 3.5 shows the synthetic data follows the original data's trends, possibly accentuating these correlations even further. Additionally, the model's synthesized data spans most of the ranges of the original data, earning a coverage score of 0.93.

3.3 Gaussian Copula

The Gaussian Copula produced generally good fits across all variable types while boasting several advantages over the Bayesian Network. Most notably, the model does well with most of the numeric variables including many of the variables the Bayesian Network struggled to

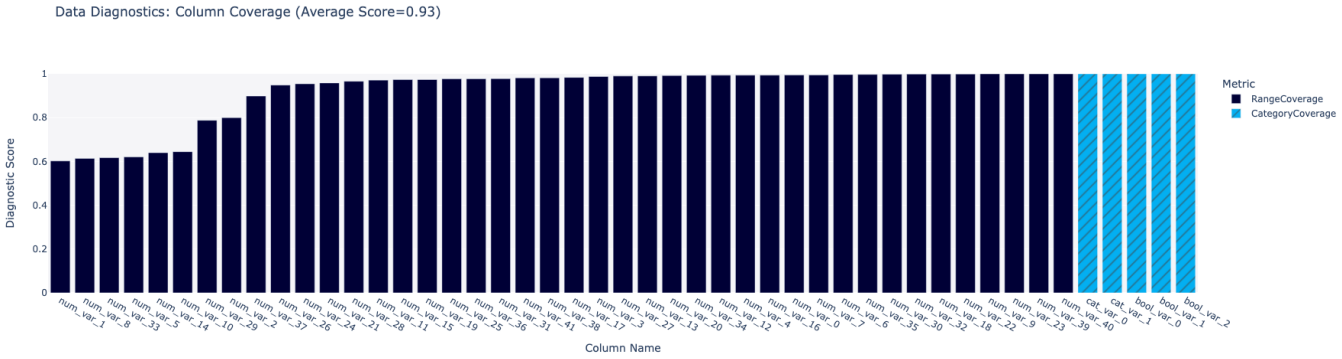

Figure 3.6: Bayesian Network Coverage

model. The column shapes bar plot in Figure 3.7 shows the efficacy of the Gaussian Copula at simulating real numeric data. These improvements led the GC to an overall column shapes score of 0.7844.

Model	Column Shapes	Column Pair Trends	Coverage
GC	0.7844	0.8265	0.709

Table 3.3: Gaussian Copula Results

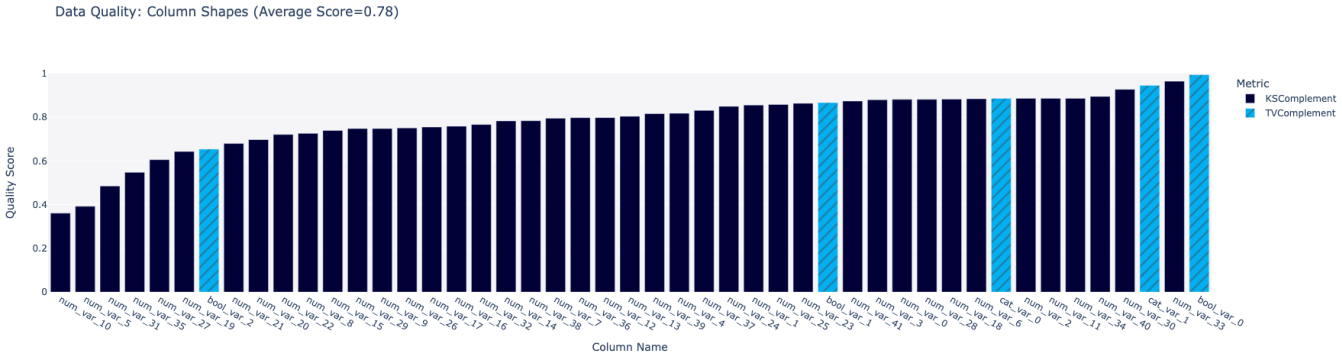

Figure 3.7: Gaussian Copula Column Shape Scores

The improvement over the Bayesian Network for numeric variables is further illustrated in the Gaussian Copula distribution of num_var_5 in Figure 3.8. Although the column shape score for the variable is still relatively low at 0.4, the Gaussian Copula does a better job

at approximating the shape of the distribution compared to the BN. As this method uses Gaussian distributions, it will generally do a good job at modeling Gaussian-like distributions the original data.

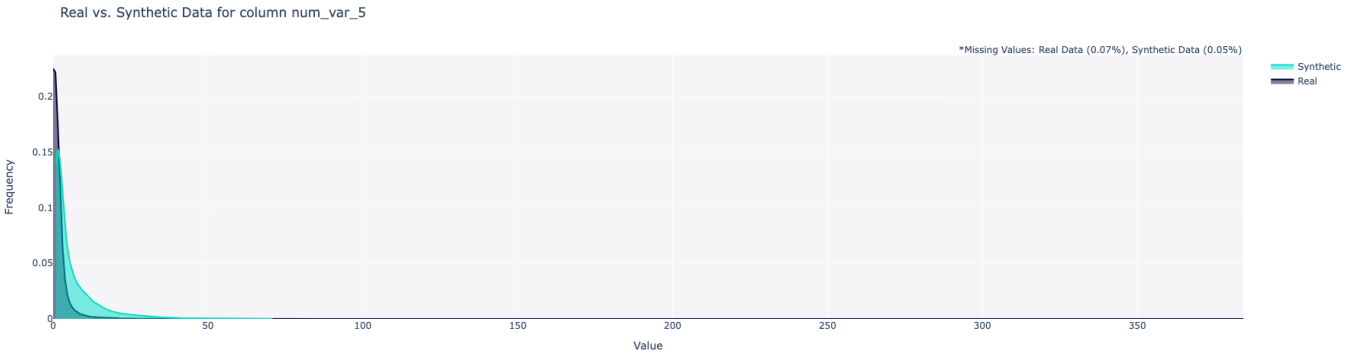

Figure 3.8: Gaussian Copula num_var_5 Distribution

However, the model's overall improvement to variable distributions comes in exchange for a significant loss in coverage. As depicted in Figure 3.9, the coverage scores indicate that the model covers less than 40% of the range for many of the numeric variables. These poorly covered variables are also the same numeric variables that the Bayesian Network struggled to synthesize effectively, characterized by their concentration around 0 and possessing large maximum values. In essence, the model is exchanging broader column coverage for enhanced precision in depicting the distribution's shape.

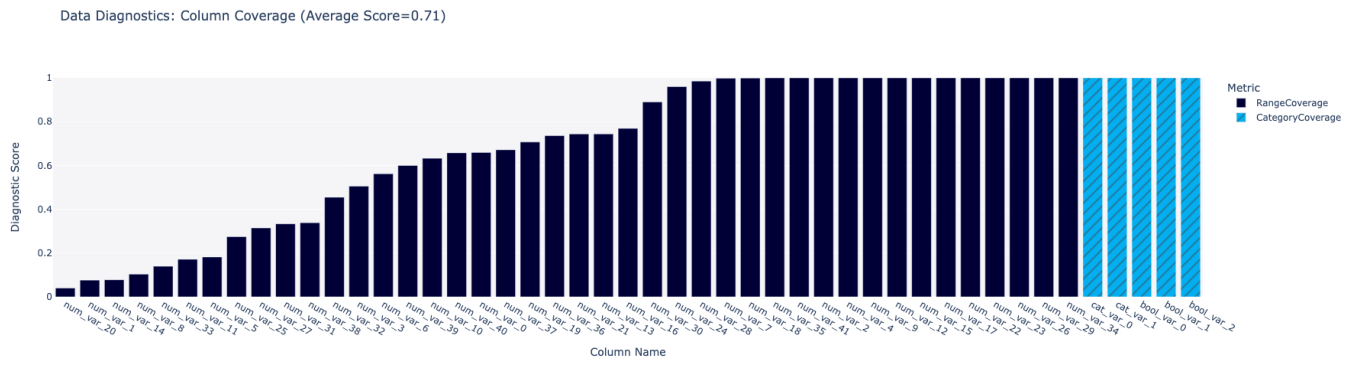

Figure 3.9: Gaussian Copula Coverage

In contrast to the Bayesian Network (BN), the Gaussian Copula encounters challenges when

it comes to modeling categorical and boolean variables. Moreover, it exhibits difficulty in handling multimodal variables, as exemplified by num_var_18, whose distribution is illustrated in Figure 3.10. This particular variable has three distinct Gaussian-like curves, yet the synthesizer retains only one of them. In the context of fraud detection, this behavior can have significant repercussions, as additional modes in the data may serve as potential indicators of fraudulent patterns. The omission of such information could potentially diminish the utility of the synthetic data for fraud detection purposes.

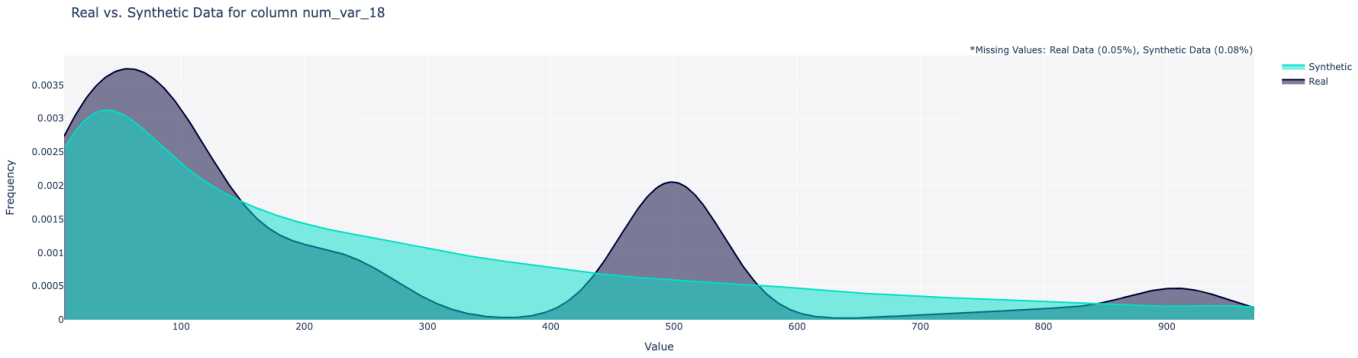

Figure 3.10: Gaussian Copula num_var_18 Distribution

The Gaussian Copula achieved an overall column pair trends score of 0.8265, implying that the model reasonably reproduces the pairwise correlations observed in the original data. While this score falls noticeably short of the Bayesian Network score, the correlation heatmap in Figure 3.11 reveals that the Gaussian Copula effectively captures and accentuates the strong correlations present in the real data. However, it may not have detected the subtle correlations that the Bayesian Network picked up on.

It is also worth noting that the Gaussian Copula exhibits a lower proficiency in generating categorical and boolean variables when compared to the Bayesian Network (BN). This pattern is particularly evident in the synthetic distribution of bool_var_2, which attained a column shape score of approximately 0.6. The distribution of bool_var_2, as depicted in Figure 3.12, illustrates that the synthesizer overemphasizes the occurrence of 1 (true) values in the variable. Moreover, the synthetic data for this variable contains no missing values, in

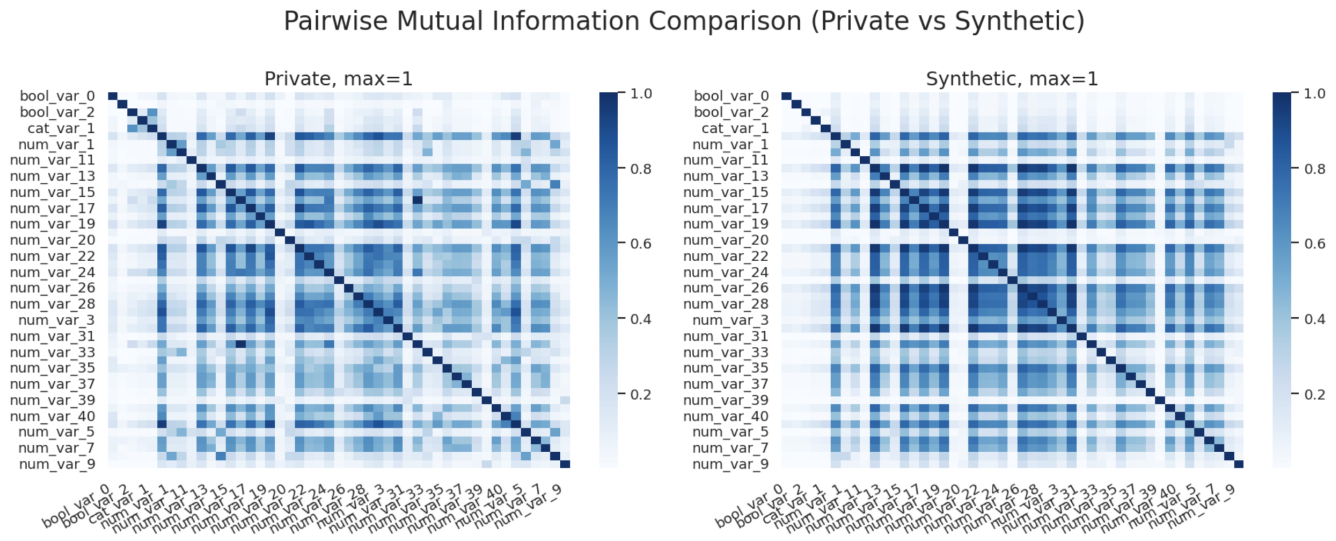

Figure 3.11: Gaussian Copula Correlation Heatmap

contrast to the real data, which exhibits a missing value rate of 29.47%.

Figure 3.12: Gaussian Copula bool_var_2 Distribution

3.4 TVAE

The Tabular Variational Autoencoder is the first of two deep learning methods tested in this project, with the CTGAN serving as the other. Typically, a neural network-based architecture introduces elevated complexity, resulting in longer training times and enhanced accuracy. This association also holds true when comparing the TVAE to the prior methods

as the TVAE achieves the highest average column shape score at 0.8469 indicating that this model is very effective at synthesizing the original data.

Model	Column Shapes	Column Pair Trends	Coverage
TVAE	0.8469	0.8342	0.5691

Table 3.4: TVAE Results

Figure 3.13 underscores the TVAE's exceptional effectiveness in generating data that closely resembles the original dataset. Notably, the model excels in synthesizing numeric columns, with all columns scoring a minimum column shape score of 0.6. Moreover, the TVAE captures patterns that were inaccurately modeled by the previous methods, including tightly clustered distributions with high maximums and multimodal distributions.

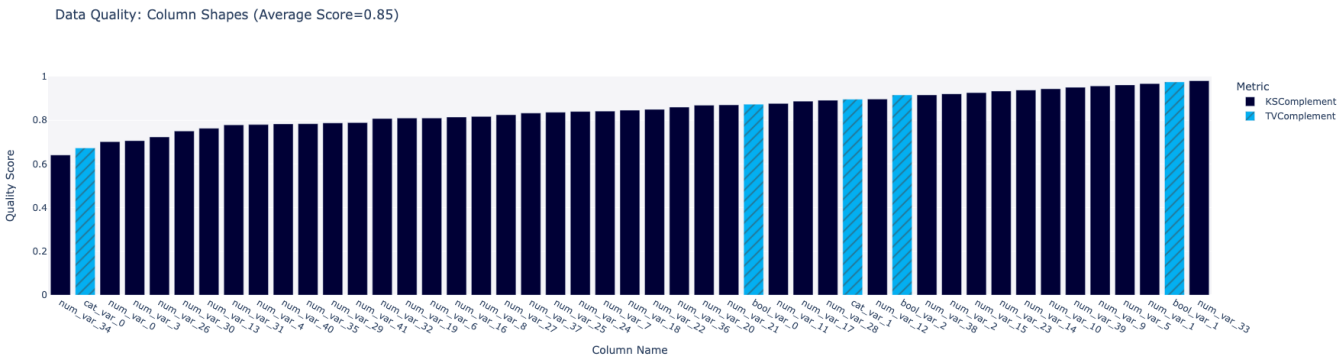

Figure 3.13: TVAE Column Shape Scores

This discrepancy is evident in Figure 3.14 and Figure 3.15, which shows the TVAE's distributions for num_var_5 and num_var_18 respectively. The TVAE exhibits strong modeling capabilities, particularly in the case of num_var_5 and other variables sharing similar distribution characteristics. It succeeds in synthesizing these columns to closely mirror the original data. For num_var_18, the model overemphasizes the first mode at the expense of the third mode. Despite this, it effectively captures and reproduces the general shape of the distribution, demonstrating its proficiency in representing complex data patterns.

The TVAE, however, is similar to the Gaussian Copula as it also struggles a bit with cat-

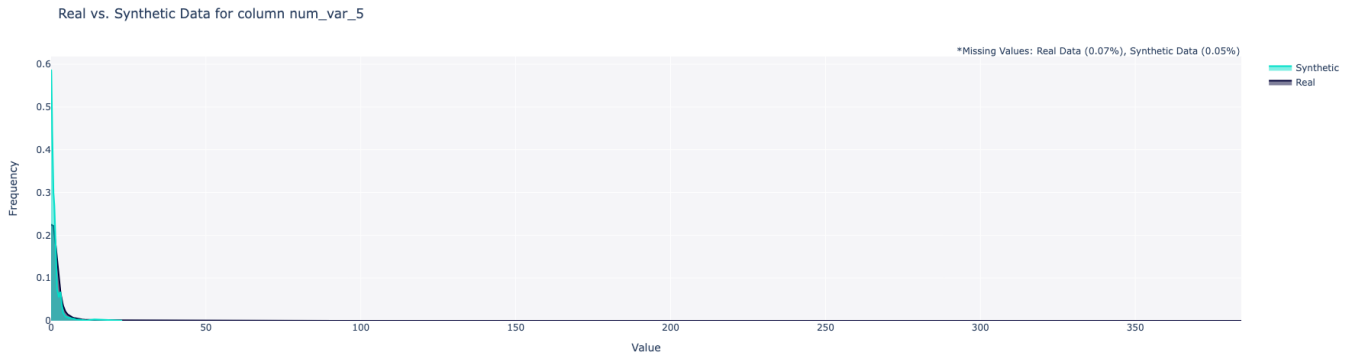

Figure 3.14: TVAE num_var_5 Distribution

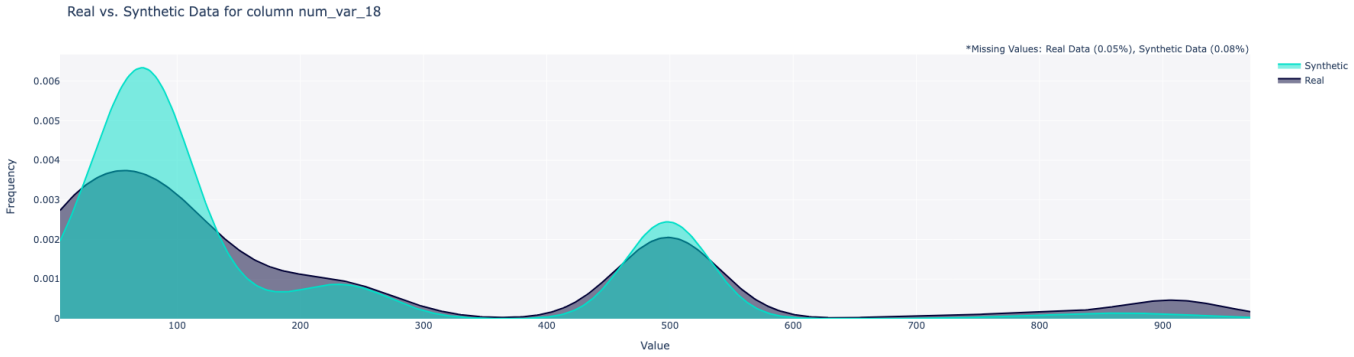

Figure 3.15: TVAE num_var_18 Distribution

egorical and boolean variables. For example, cat_var_0 is the second worst fit according to the column shape scores and its distribution can be seen in Figure 3.16. The synthetic data contains exclusively one value for the whole column, as well as an 8% higher missing rate when compared to the real data.

Furthermore, the model exhibits a notably lower average coverage score of 0.5691 compared to the previous two models. The coverage scores, as depicted in Figure 3.17, highlight the TVAE's limited value diversity in certain columns, even though it generally maintains the original data distributions quite faithfully. While the synthesizer may not incorporate outliers that were present in the original data, it does manage to capture the essence of the real data distributions with a high degree of accuracy. The correlation heatmap in Figure 3.18 displays the pairwise correlations of the TVAE model, which are consistently lower in

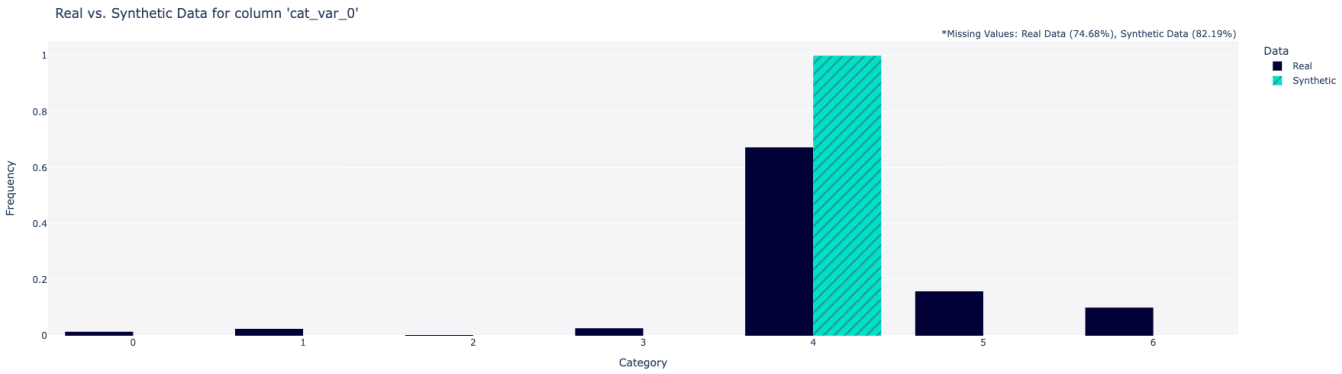

Figure 3.16: TVAE cat_var_0 Distribution

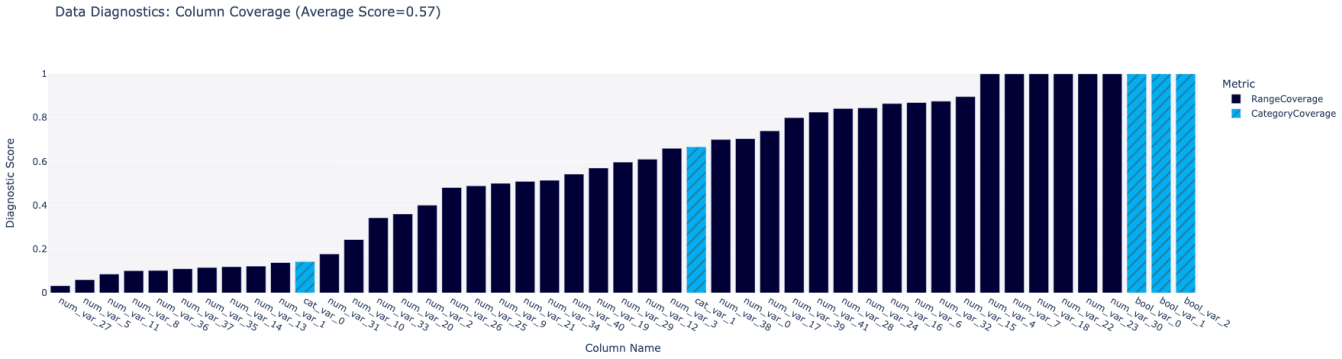

Figure 3.17: TVAE Coverage

comparison to those generated by the Gaussian Copula and Bayesian Network models. The GC and BN models may have magnified certain correlations present in the original data, which is not the case with the TVAE. However, it's worth noting that the TVAE still achieves a respectable column pair trends score of 0.8342.

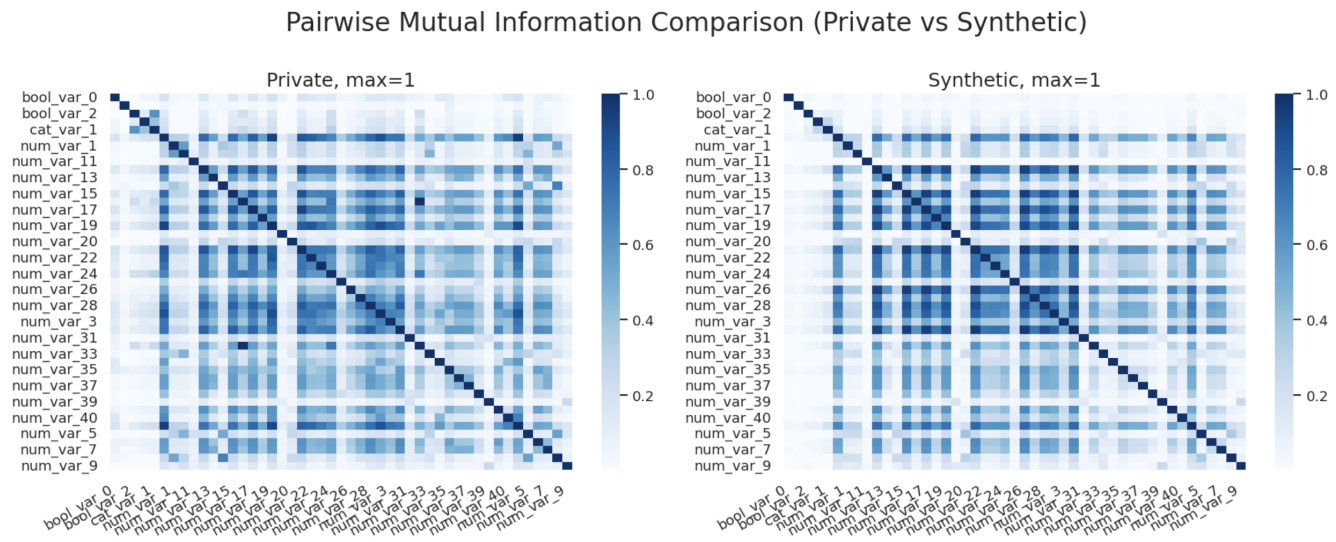

Figure 3.18: TVAE Correlation Heatmap

3.5 CTGAN

With a 0.8116 column shapes score, the CTGAN performs well at generating data that adheres to the original data's distribution. As illustrated in Figure 3.19, the column shape scores reveal similarities between the CTGAN and the TVAE, both of which perform adequately across various data types. However, the model exhibits some difficulty with categorical and boolean variables when compared to the Bayesian Network. Otherwise, the model does well, achieving a column shape score of at least 0.6 for nearly all variables, with just two exceptions.

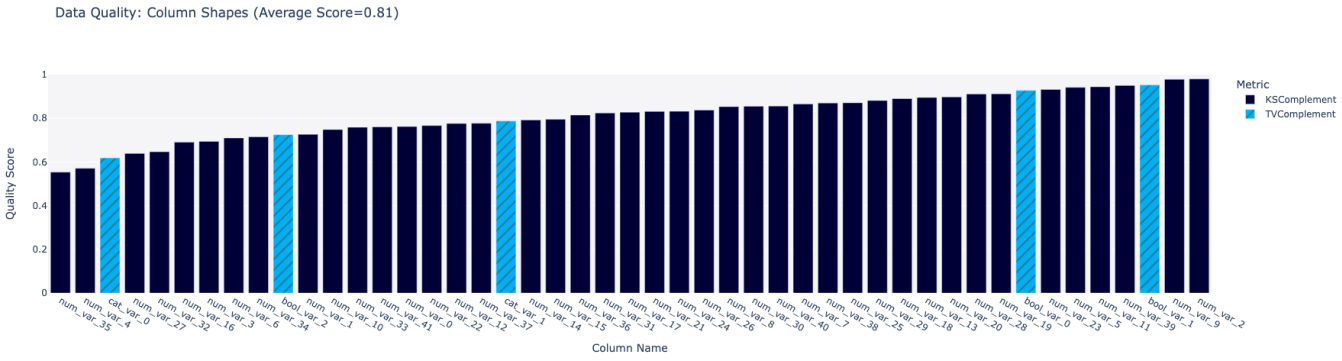

Figure 3.19: CTGAN Columns Shape Scores

Model	Column Shapes	Column Pair Trends	Coverage
CTGAN	0.8116	0.8038	0.5917

Table 3.5: CTGAN Results

When analyzing the interactions between the model and different variables types and distributions, it is clear the CTGAN represents a significant advancement over preceding methodologies. Figure 3.20 effectively illustrates this point by showcasing the distribution of the multimodal variable num_var_18, which both the Gaussian Copula and TVAE struggled to synthesize properly. Remarkably, CTGAN accurately reproduces a distribution with three distinct modes, mirroring the patterns observed in the authentic data. This demonstrates CTGAN's capacity to address complex numeric distribution that posed difficulties for earlier approaches.

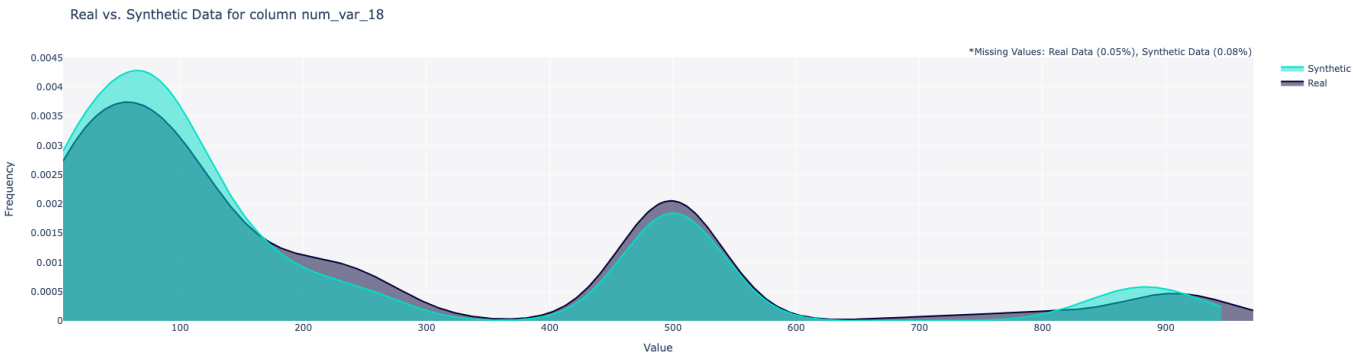

Figure 3.20: CTGAN num_var_18 Distribution

The numeric variable that scored the worst according to the column shape scores was num_var_35, whose distributions can be found in Figure 3.21. Despite its relatively modest score of 0.55, the synthetic distribution succeeds in preserving the fundamental shape characteristics of the original distribution, albeit with a larger proportion of values concentrated near the mean. In addition num_var_35, CTGAN does not significantly distort the shape of any other numeric variables to the same extent as observed with the previous models.

Despite the CTGAN's strengths in modeling numeric variables, the model shares similar

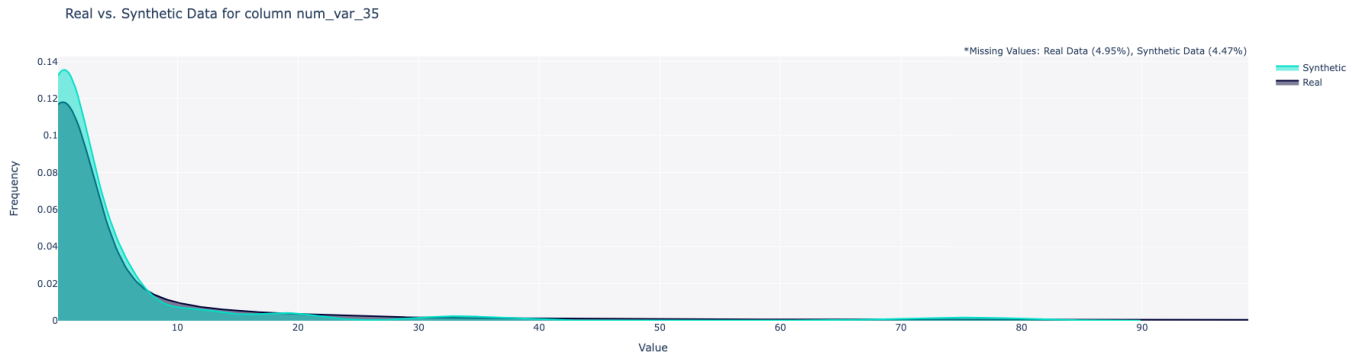

Figure 3.21: CTGAN num_var_18 Distribution

shortcomings as the TVAE. Specifically, the model struggles with synthesizing categorical and boolean columns, poor coverage, and training complexity. For cat_var_0, the CTGAN

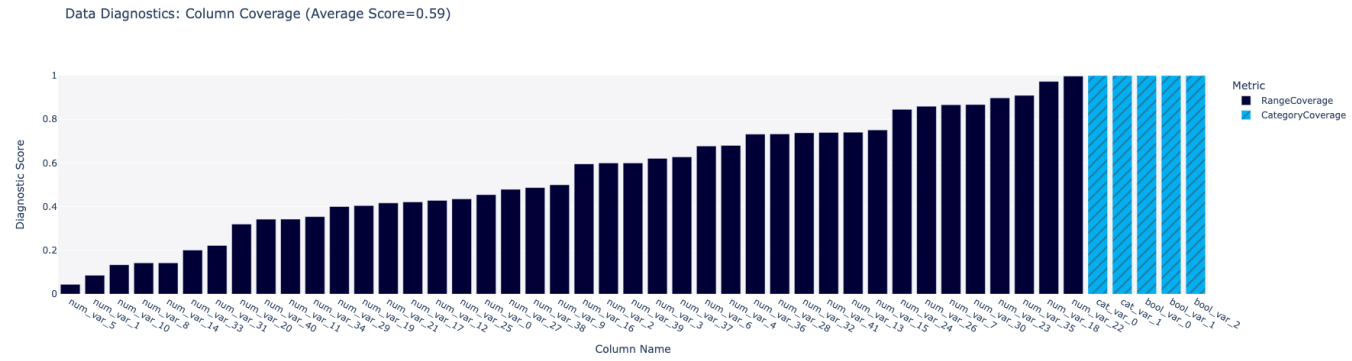

Figure 3.22: CTGAN Coverage

under represents the majority class and missing values as seen in Figure 3.22. Additionally, the model consistently gets lower column shape scores than TVAE when synthesizing categorical and boolean columns. The coverage scores in Figure 3.23 and average coverage score of 0.5917 are reminiscent of the TVAE's coverage metrics. Additionally, CTGAN shares similar column pair trends scores with the TVAE and both models exhibit comparable correlation heatmaps.

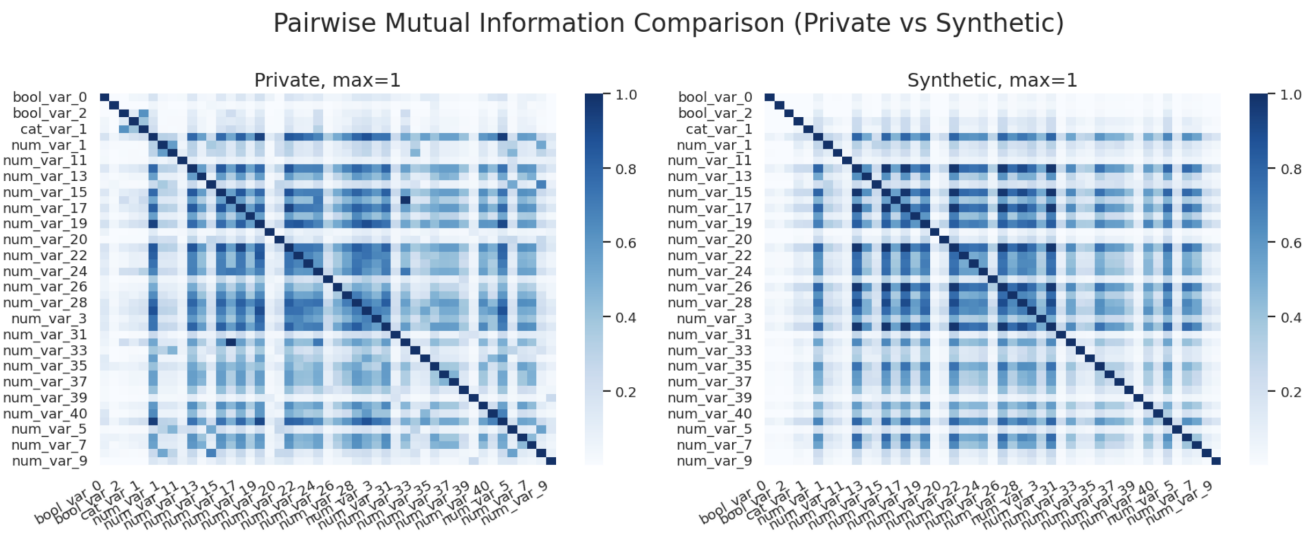

Figure 3.23: CTGAN Correlation Heatmap

3.6 Synthetic Data Comparison

Model	Column Shapes	Column Pair Trends	Coverage
BN	0.7217	0.9685	0.9311
GC	0.7844	0.8265	0.709
TVAE	0.8469	0.8342	0.5691
CTGAN	0.8116	0.8038	0.5917

Table 3.6: Synthetic Data Quality Metrics Table

When conducting a comparative analysis of the various data synthesizers employed, distinct disparities emerge between the outcomes achieved by deep learning algorithms (TVAE and CTGAN) and statistical models (BN and GC). The deep learning models notably distinguish themselves through their ability to produce accurate synthetic data distributions, as evidenced by the TVAE achieving the highest average column shape score at 0.8469. Conversely, the statistical models excel in terms of data coverage, with the Bayesian model, achieving a high score of 0.9311. The metrics in Table 3.6 suggests the existence of a trade-off between column shape and coverage, where higher column shape scores correspond to

lower coverage scores. For the purposes of using synthetic data to enrich machine learning model training sets, preserving the accurate shape of the data distribution may hold greater importance than achieving complete coverage of all potential data points. Nevertheless, maintaining proportional data representation remains a important consideration.

The column pair trends also provide a interesting metric to analyze our models from. The Bayesian Network emerges as the leading model in this regard, effectively replicating pairwise correlations from the original data. This outcome aligns with expectations, given the Bayesian Network's focus on modeling correlations between variables. However, it is important to acknowledge that while the BN excels in maintaining pairwise correlation trends, its performance may vary when confronted with more intricate interactions between variables.

Another pertinent comparison pertains to the deep learning models themselves. The TVAE outperforms the CTGAN in both column shapes and column pair trends metrics. Nevertheless, during the exploration of individual synthetic variable distributions, it becomes evident that the CTGAN successfully addresses certain issues observed in the TVAE's synthesis of numeric and multimodal variables. This suggests that while the CTGAN boasts specific advantages over the TVAE, the latter may offer more precise overall distribution accuracy.

3.7 Model Performance Comparison

Table 3.7 below displays the results of training the baseline model with differing amounts of synthetic data generated with the various synthesizers from the previous section. In the table, p represents the proportion of fraud orders in the training set that were synthetically generated. Table 3.8 displays the values of Table 3.7 relative to the baseline model as percent increases and decreases.

Model	PR AUC	ROC AUC	Best F1 Score
Baseline	0.2884	0.8415	0.3859
BN ($p = 0.05$)	0.2718	0.8422	0.3499
BN ($p = 0.10$)	0.2553	0.8399	0.3353
BN ($p = 0.15$)	0.2513	0.8393	0.3295
GC ($p = 0.05$)	0.2547	0.8348	0.3346
GC ($p = 0.10$)	0.2682	0.8351	0.3514
GC ($p = 0.15$)	0.2723	0.8305	0.3586
TVAE ($p = 0.05$)	0.3082	0.8443	0.3934
TVAE ($p = 0.10$)	0.3085	0.8421	0.3907
TVAE ($p = 0.15$)	0.3050	0.8430	0.3924
CTGAN ($p = 0.05$)	0.3120	0.8391	0.3989
CTGAN ($p = 0.10$)	0.3113	0.8388	0.4004
CTGAN ($p = 0.15$)	0.3075	0.8391	0.3922

Table 3.7: Model Performance Metrics

Based on our research findings, the inclusion of data generated solely from the BN and GC models appears to have a detrimental impact on model performance. Conversely, the utilization of deep learning models proves to be beneficial, enhancing the predictive capabilities of the baseline model. Notably, datasets augmented by the TVAE exhibit improvements in key performance metrics, such as PR AUC, ROC AUC, and F1 score. Datasets augmented

by the CTGAN show substantial enhancements in PR AUC and F1 Score, albeit with minor decreases in ROC AUC.

It is noteworthy that, except for the Gaussian Copula, the model consistently performed optimally when 5% to 10% of the fraudulent orders in the training set were synthetic. This suggests that this range typically represents the most effective proportion for appending synthetic data to the positive class of the training dataset.

Model	PR AUC	ROC AUC	Best F1 Score
BN ($p = 0.05$)	-5.75%	+0.08%	-9.33%
BN ($p = 0.10$)	-11.51%	-0.19%	-13.11%
BN ($p = 0.15$)	-12.86%	-0.26%	-14.62%
GC ($p = 0.05$)	-11.17%	-0.80%	-13.29%
GC ($p = 0.10$)	-7.00%	-0.76%	-8.94%
GC ($p = 0.15$)	-5.58%	-1.31%	-7.07%
TVAE ($p = 0.05$)	+6.87%	+0.33%	+1.94%
TVAE ($p = 0.10$)	+6.97%	+0.07%	+1.24%
TVAE ($p = 0.15$)	+5.76%	+0.18%	+1.68%
CTGAN ($p = 0.05$)	+8.18%	-0.29%	+3.37%
CTGAN ($p = 0.10$)	+7.94%	-0.32%	+3.76%
CTGAN ($p = 0.15$)	+6.62%	-0.29%	+1.63%

Table 3.8: Model Performance Relative Metrics

CHAPTER 4

Conclusion

Synthetic data generation is an evolving field within the realm of machine learning, characterized by continuous development of new methods and applications. In our study, we employed four distinct synthesizers to augment a dataset focused on Buy Now Pay Later (BNPL) fraud. As is typical with many fraud detection datasets, our data exhibits a substantial class imbalance, with a fraud rate of just 0.09%. When training models with datasets enriched by deep learning models, we observed notable performance enhancements, including an up to 8.18% increase in PR AUC and a 3.76% rise in F1 Score. This research establishes a valuable framework for leveraging synthetic data generation to tackle class imbalance challenges in the context of fraud detection.

Among the four synthesizers employed in our study, the Conditional Tabular Generative Adversarial Network (CTGAN) and Tabular Variational Autoencoder (TVAE) demonstrated the ability to generate fraudulent order data closely mirroring the variable distributions of the input data. Notably, the inclusion of fraud orders generated by a CTGAN led to substantial improvements in validation PR AUC and F1 Score, while datasets enriched by a TVAE exhibited across-the-board enhancements in PR AUC, ROC AUC, and F1 Score.

While this project serves as a promising proof of concept, there is room for further development of this framework. Future research endeavors may involve the creation of additional evaluation metrics to comprehensively assess the effectiveness of each method in generating synthetic data that replicates the original data's distribution. Furthermore, considering our research primarily focuses on enhancing gradient boosted tree methods, future investigations

could also delve into the interactions between various statistical and machine learning models and the data generated by these synthesizers.

REFERENCES

[1] Jianmin Bao et al. "CVAE-GAN: fine-grained image generation through asymmetric training". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2745–2754.

[2] Consumer Financial Protection Bureau. *Buy Now, Pay Later Market Trends and Consumer Impacts*. 2022. URL: https://files.consumerfinance.gov/f/documents/cfpb_buy-now-pay-later-market-trends-consumer-impacts_report_2022-09.pdf.

[3] Yinan Cheng et al. "Downstream Task-Oriented Generative Model Selections on Synthetic Data Training for Fraud Detection Models". In: *ACM International Conference on AI in Finance – Workshop* (2023).

[4] Justin Engelmann and Stefan Lessmann. "Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning". In: *Expert Systems with Applications* 174 (2021), p. 114582.

[5] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).

[6] Yifeng Li et al. "The Max-Min High-Order Dynamic Bayesian Network for Learning Gene Regulatory Networks with Time-Delayed Regulations". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 13.4 (2016), pp. 792–803. DOI: 10.1109/TCBB.2015.2474409.

[7] Wentong Liao et al. "Text to image generation with semantic-spatial aware gan". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 18187–18196.

[8] Zinan Lin et al. "Pacgan: The power of two samples in generative adversarial networks". In: *Advances in neural information processing systems* 31 (2018).

[9] Donald MacKenzie and Taylor Spears. "'The formula that killed Wall Street': The Gaussian copula and modelling practices in investment banking". In: *Social Studies of Science* 44.3 (2014). PMID: 25051588, pp. 393–417. DOI: 10.1177/0306312713517157. eprint: https://doi.org/10.1177/0306312713517157. URL: https://doi.org/10.1177/0306312713517157.

[10] Frank J Massey Jr. "The Kolmogorov-Smirnov test for goodness of fit". In: *Journal of the American statistical Association* 46.253 (1951), pp. 68–78.

[11] Nikolaos Nasios and Adrian G Bors. "Variational learning for Gaussian mixture models". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 36.4 (2006), pp. 849–862.

[12] Hasna Njah and Salma Jamoussi. "Weighted ensemble learning of Bayesian network for gene regulatory networks". In: *Neurocomputing* 150 (2015), pp. 404–416.

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

[14] Haoyue Ping, Julia Stoyanovich, and Bill Howe. "DataSynthesizer: Privacy-Preserving Synthetic Datasets". In: *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. SSDBM '17. Chicago, IL, USA: Association for Computing Machinery, 2017. ISBN: 9781450352826. DOI: 10.1145/3085504.3091117. URL: https://doi.org/10.1145/3085504.3091117.

[15] Akash Srivastava et al. "Veegan: Reducing mode collapse in gans using implicit variational learning". In: *Advances in neural information processing systems* 30 (2017).

[16] Lei Xu et al. "Modeling tabular data using conditional gan". In: *Advances in neural information processing systems* 32 (2019).

[17] Jianwei Yang et al. "Lr-gan: Layered recursive generative adversarial networks for image generation". In: *arXiv preprint arXiv:1703.01560* (2017).

[18] Bowen Zhang et al. “Styleswin: Transformer-based gan for high-resolution image generation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11304–11314.

[19] Zhigang Zhou et al. “A Targeted Privacy-Preserving Data Publishing Method Based on Bayesian Network”. In: *IEEE Access* 10 (2022), pp. 89555–89567. DOI: 10.1109/ACCESS.2022.3201641.

[20] Min Zou and Suzanne D Conzen. “A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data”. In: *Bioinformatics* 21.1 (2005), pp. 71–79.