SYNTHESIZING CONTEXTUALLY RELEVANT TABULAR DATA USING CONTEXT-AWARE CONDITIONAL TABULAR GAN AND TRANSFER LEARNING

by

Hesam Fallahian

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2024

Approved by:

Dr. Mohsen Dorodchi

Dr. Kyle Kreth

Dr. Pu Wang

Dr. Christian Kuemmerle

ABSTRACT

HESAM FALLAHIAN. Synthesizing Contextually Relevant Tabular Data Using Context-Aware Conditional Tabular GAN and Transfer Learning. (Under the direction of DR. MOHSEN DORODCHI)

The Context-Aware Conditional Tabular Generative Adversarial Network (CA - CTGAN) introduces an innovative architecture for the generation of synthetic tabular data, distinguished by effectively incorporating context-specific elements into its generative process. This enables the production of synthetic datasets that not only accurately reflect real-world distributions but are also tailored to specific contexts across a variety of experimental domains, including laboratory, field, natural, and clinical experiments, as well as survey research. In many cases, CA-CTGAN can generate data suitable for research purposes, potentially reducing or eliminating the need for certain real-world experiments. By utilizing Transfer Learning the model effectively identifies and exploits complex semantic relationships within the data to ensure the implementation of rigorous contextual requirements and maintains high semantic integrity. Furthermore, a novel auxiliary classifier is implemented, which includes entity embedding and multi-class multi-label capabilities, enabling the creation of enhanced datasets that strictly adhere to the specified contextual requirements. These contributions position CA-CTGAN as a remarkably versatile and efficient tool across multiple scientific disciplines. Its ability to generate high-quality, contextually relevant synthetic data not only streamlines research processes and reduces associated costs but also addresses ethical concerns in sensitive studies. Consequently, CA-CTGAN emerges as an essential resource for researchers, facilitating more ethical, cost-effective, and data-informed experimental design and decision-making.

DEDICATION

To my precious daughter Diana, may this work inspire you to pursue your passions with relentless determination. Remember that knowledge is powerful and anything is possible with hard work and a little bit of magic. My heart is filled with pride for the amazing woman you are becoming.

ACKNOWLEDGEMENTS

I extend my gratitude to my advisor, Dr. Mohsen Dorodchi, for his invaluable guidance and unwavering support throughout this research. His insights and dedication have been pivotal to my development and the completion of this work.

Special thanks are due to Dr. Kyle Kreth, whose expertise and detailed feedback significantly shaped many aspects of this dissertation. His contributions were crucial in refining my arguments and enhancing the overall quality of my study.

I am also thankful to my committee members, Dr. Pu Wang and Dr. Christian Kuemmerle, for their constructive critiques and encouraging guidance, which have been greatly appreciated.

TABLE OF CONTENTS

LIST OF TABLES x

LIST OF FIGURES xii

LIST OF ABBREVIATIONS 1

CHAPTER 1: INTRODUCTION 1

1.1. Problem Statement 2

1.2. Experimental Domains 3

1.3. Research Questions 5

1.4. Objective and Contribution 6

1.5. Overview of the report 8

CHAPTER 2: BACKGROUND AND RELATED STUDIES 9

2.1. Background Knowledge 11

2.1.1. Generative Adversarial Networks (GANs) 11

2.1.2. Conditional Generative Adversarial Networks (CGANs) 12

2.1.3. Autoencoders 13

2.1.4. Conteractive Autoencoder 16

2.1.5. Transfer Learning 18

2.2. Tabular Data Generation Challenges 19

2.2.1. Data type 19

2.2.2. Bounded continuous columns 20

2.2.3. Non-Gaussian Distribution 21

2.2.4. Semantic relationship 22

2.3. Related Studies 23

2.3.1. Data Transformation 29

2.3.2. Distribution Matching 31

2.3.3. Conditional and Informed Generator 32

2.3.4. Comparative Analysis of GAN-Based Methods 37

2.3.5. Tabular GAN Evolution 38

2.3.6. GAN vs. Diffusion Model for Tabular Data Generation 39

CHAPTER 3: METHODOLOGY 42

3.1. Framework Overview 42

3.2. Data Transformation 43

3.2.1. Continuous Columns 44

3.2.2. Categorical Data 46

3.2.3. Mixed-mode Data 48

3.3. Design and Training process 48

3.3.1. Contractive Autoencoder 49

3.3.2. Generator 53

3.3.3. Discriminator 54

3.3.4. Auxiliary Classifier 57

3.3.5. Loss Function 57

3.3.6. Training Process 62

3.4. Contribution and Novelty 63

CHAPTER 4: EXPERIMENTAL STUDIES 66

4.1. Datasets 66

4.1.1. Adult Income Dataset 66

4.1.2. Air Quality dataset 69

4.1.3. Apartment for Rent dataset 70

4.1.4. Bank Marketing dataset 72

4.1.5. Beijing PM2.5 dataset 75

4.1.6. Bike Sharing Dataset 76

4.1.7. Individual Household Electric Power Consumption Dataset 77

4.1.8. Metro Interstate Traffic Volume Dataset 78

4.1.9. MetroPT-3 Dataset 80

4.2. Baselines and Experimental Setup 82

4.3. Evaluation Metrics 84

4.3.1. Data Coverage 85

4.3.2. Data Constraint 86

4.3.3. Data Similarity 87

4.3.4. Data Relationship 89

4.3.5. ML Detection 90

4.3.6. ML Efficiency 91

CHAPTER 5: RESULT AND ANALYSIS 93

5.1. Model Performance 93

5.1.1. Data Coverage 93

5.1.2. Data Constraint 99

5.1.3. Data Similarity 102

5.1.4. Data Relationship 109

5.1.5. Machine Learning Performance 120

5.2. Comparative Study 122

CHAPTER 6: CONCLUSIONS AND FUTURE WORK 132

REFERENCES 135

LIST OF TABLES

TABLE 2.1: Various Generative Models based on the type of learning. 10

TABLE 2.2: The data types that can be used in a tabular data table. 20

TABLE 2.3: Comparative overview of GAN-based vs. traditional methods in tabular data generation. 37

TABLE 2.4: Different tabular GAN architecture and capability. 39

TABLE 4.1: Datasets shape 82

TABLE 5.1: Data Coverage score for Adult, Bank Marketing, and Metro PT-3 datasets. 95

TABLE 5.2: Data Coverage score for Air Quality, Bike Sharing, and Metro Interstate Traffic. 95

TABLE 5.3: Data Coverage score for Apartment Rent, Power Consumption and Beijing PM datasets. 96

TABLE 5.4: Data Constraint metric: category and range adherence score for Adult, Bank Marketing and Metro PT datasets. 100

TABLE 5.5: Data Constraint metric: category and range adherence score for Air Quality, Bike Sharing and Metro Interstate Traffic datasets. 100

TABLE 5.6: Data Constraint metric: category and range adherence score for Apartment Rent, Power Consumption and Beijing PM datasets. 101

TABLE 5.7: Similarity score of real and generated column for KS and TVD statistic for Adult, Bank Marketing and Metro PT datasets. 103

TABLE 5.8: Similarity score of real and generated column for KS and TVD statistic for Air Quality, Bike Sharing and Metro Interstate Traffic datasets. 104

TABLE 5.9: Similarity score of real and generated column for KS and TVD statistic for Apartment Rent, Power Consumption and Beijing PM datasets. 105

TABLE 5.10: Shape similarity for discrete columns using Chi-squared test for Adult, Bank Marketing and Metro PT. 107

TABLE 5.11: Shape similarity for discrete columns using Chi-squared test for Bike Sharing, Metro Traffic and Beijing PM. 107

TABLE 5.12: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Metro PT and Bike Sharing datasets. 108

TABLE 5.13: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Power Consumption, Beijing PM, and Adult datasets. 109

TABLE 5.14: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Metro Interstate Traffic, Bank Marketing, and Apartment Rent datasets. 110

TABLE 5.15: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Air Quality dataset. 110

TABLE 5.16: ML Detection (Logistic Regression) and ML Efficiency (AdaBoost) score for all datasets. 122

TABLE 5.17: A comparative analysis of CA-CTAGN and baseline methods was conducted on five datasets. 123

LIST OF FIGURES

FIGURE 2.1: GANs process flow diagram. 12

FIGURE 2.2: Conditional GAN process flow diagram. 13

FIGURE 2.3: The network architecture for a normal autoencoder. 14

FIGURE 2.4: Visual representation of Conteractive Autoencoders. 18

FIGURE 2.5: Each soccer team in the table corresponds to a particular location and has a specific capacity, foundation year, and year of entry into MLS. (a) and (b) shows two examples of constraint-based and rule-based sample rejection during the data synopsis generation process. 23

FIGURE 2.6: medGAN architecture: Discriminator utilizes autoencoder (which is learned by real data) to receive decoded random noise variable 25

FIGURE 2.7: Pre-processing input data before feeding the discriminator in PNR-GAN 26

FIGURE 2.8: Loss functions representation in table-GAN architecture. 27

FIGURE 2.9: Following vectorization of categorical columns, all vectors will be initiated by 0, then j^{th} category from i^{th} column will be selected, and the value of the corresponding element will be changed to 1. 34

FIGURE 2.10: DATGAN process flow digram. 36

FIGURE 2.11: Tabular GAN-based generators evolution based on their relationship. Yellow boxes are tabular generators, and green boxes introduced for non-tabular data. 38

FIGURE 3.1: Components of CA-CTGAN. 44

FIGURE 3.2: Distribution over a mixed-type column. m_1 and m_4 represent the categorical part or null values of this column, whereas m_2 and m_3 represent modes for numeric parts. The numeric parts are defined by Variational Gaussian Mixture (VGM) model. [1] 49

FIGURE 3.3: In the PacGAN model, the input layer is expanded by a factor of the packing degree (Here $m = 2$). The connections to the initial hidden layer are modified to ensure that the first two layers maintain full connectivity, consistent with the original architecture [2]. 56

FIGURE 3.4: Detailed schematic representation of the CA-CTGAN training process and architecture. 64

FIGURE 4.1: Distribution and correlation of numerical values in the Adult dataset 67

FIGURE 4.2: Frequency of different categories within each categorical column in the Adult dataset 68

FIGURE 4.3: Frequency of different categories within each categorical column in the Adult dataset 69

FIGURE 4.4: Distribution and correlation of numerical values in the Air Quality dataset 70

FIGURE 4.5: Distribution and correlation of numerical values in the Apartment dataset 71

FIGURE 4.6: Frequency of different categories within each categorical column in the Apartment dataset 72

FIGURE 4.7: Distribution and correlation of numerical values in the Bank Marketing dataset 73

FIGURE 4.8: Frequency of different categories within each categorical column in the Bank Marketing dataset 74

FIGURE 4.9: Distribution and correlation of numerical values in the Beijing PM2.5 datasets 76

FIGURE 4.10: Distribution and correlation of numerical values in the Bike Sharing datasets 77

FIGURE 4.11: Distribution and correlation of numerical values in the Household Electric Power Consumption datasets 78

FIGURE 4.12: Distribution and correlation of numerical values in the Metro Interstate Traffic Volume datasets 79

FIGURE 4.13: Frequency of different categories of categorical columns in the Metro Interstate Traffic Volume dataset 80

FIGURE 4.14: Distribution and correlation of numerical values in the MetroPT-3 datasets 81

FIGURE 4.15: Distances are measured between 0 and 1, but the complement of this metric can also be considered. Therefore, a higher score indicates higher quality according to 1-(KS statistic distance) [3]. 87

FIGURE 5.1: Data distribution for Voltage column in Power Consumption dataset. 94

FIGURE 5.2: Frequency of categories in native-country column in Adult dataset and holiday column in Metro Interstate Traffic dataset. 96

FIGURE 5.3: Comparison of data category coverage for native-country column in Adult and holiday column in Metro Interstate Traffic dataset. 97

FIGURE 5.4: Data distribution and comparison of data category coverage for capital-loss column in Adult dataset. 98

FIGURE 5.5: Range adherence for IWS column in Beijing PM dataset. 101

FIGURE 5.6: Range adherence for NOx(GT) column in Air Quality dataset. 102

FIGURE 5.7: Column shape score for Metro PT dataset. 103

FIGURE 5.8: Column shape score for Air Quality dataset. 104

FIGURE 5.9: Column shape score for Bike Sharing dataset. 105

FIGURE 5.10: Column shape score for Beijing PM dataset. 106

FIGURE 5.11: Relationship between a pair of discrete and numerical Columns in the Adult dataset. 112

FIGURE 5.12: Relationship between a pair of discrete columns in the Adult dataset. 113

FIGURE 5.13: Relationship between a pair of continuous columns (NOx/NO2 and PT08) in the Air Quality dataset. 114

FIGURE 5.14: Relationship between a pair of continuous columns RH/T and C6H6/CO in the Air Quality dataset. 115

FIGURE 5.15: Relationship between a pair of continuous (sure feet and price) and a pair of discrete and continuous columns (square feet and cityname) in the apartment dataset. 117

FIGURE 5.16: Relationship between a pair of discrete columns and target label in the Bank Marketing dataset. 118

FIGURE 5.17: Relationship between a pair of continuous and discrete columns and target label in the Beijing PM dataset. 119

FIGURE 5.18: Relationship between a pair of continuous and discrete columns and a high-precision small number in Bike Sharing dataset. 120

FIGURE 5.19: Frequency distribution comparison for 'Education' in Adult Dataset. (a) CTAGN (b) CA-CTGAN. 124

FIGURE 5.20: Frequency distribution comparison for 'LPS' in MetroPT dataset. (a) CTAGN (b) CA-CTGAN 125

FIGURE 5.21: Heatmap of bedrooms vs. fee relationship in Apartment dataset: A real and generated data comparison. (a) Real Data (b) CTABGAN (c) CA-CTGAN 126

FIGURE 5.22: Comparative analysis of 'Price' and 'Square Feet' Relationship in Apartment Rent dataset using pair plot. (a) CTABGAN (b) CA-CTGAN 127

FIGURE 5.23: Distribution analysis of 'previous' column in Bank Marketing dataset. (a) DATGAN (b) CA-CTGAN 128

FIGURE 5.24: Distribution analysis of 'Atemp' in Bike Sharing dataset. (a) DATGAN (b) CA-CTGAN 129

FIGURE 5.25: Heatmap visualization of 'LPS' and 'COMP' Relationship in Metro PT dataset. (a) Real Data (b) CTABGAN (c) CA-CTGAN 130

FIGURE 5.26: Generator loss over the epochs for both training approaches (with and without CAE) - Adult dataset [4]. 131

CHAPTER 1: INTRODUCTION

The rapid advancements in artificial intelligence and machine learning have significantly transformed numerous scientific domains, empowering researchers to address complex problems and efficiently analyze vast volumes of data. However, acquiring sufficient high-quality data for various research endeavors remains a significant challenge. This challenge is evident across diverse experimental contexts such as field experiments, which contend with unpredictable environmental variables; natural experiments, which require data that encapsulates spontaneous events; clinical trials, demanding ethical rigor and precise simulations of patient responses; and survey research, needing to reflect a broad spectrum of human behaviors and opinions. Moreover, in each of these settings, a deep contextual understanding is essential. The intricate relationships between variables in domain-specific data are fundamental for accurate analysis and interpretation of results. The diverse and complex nature of these requirements underscores the necessity for innovative methods capable of synthesizing contextually relevant and high-quality data across various experimental domains.

In recent years, Generative Adversarial Networks (GANs) [5] have emerged as a powerful tool for generating synthetic data, offering potential solutions to some of the challenges faced in data acquisition for scientific research. The ability of GANs to generate high-fidelity data has made them a popular choice for a wide range of applications, such as generating photorealistic images, synthesizing video sequences, and generating natural language text [6]. However, traditional GANs may not fully capture the complex relationships between variables in tabular data and lack the capacity to generate high-resolution data [6].

To address these limitations, this research introduces the Context-Aware Conditional Tabular GAN (CA-CTGAN), a novel variant of GANs specifically designed to be inherently versatile for application across a wide range of experimental domains. Leveraging transfer learning and conditioning on context-specific elements like class attributes using Conditional Generative Adversarial Networks (CGAN) [7], the proposed CA-CTGAN framework aims to synthesize realistic and contextually relevant experimental data, reducing the need for physical experiments while maintaining the quality and fidelity of the generated data. This approach has the potential to accelerate scientific discovery, providing a comprehensive, efficient, and ethically considerate approach to enhance data analysis and decision-making systems.

1.1 Problem Statement

Despite the numerous benefits offered by machine learning techniques, a persistent challenge faced by researchers across various scientific domains is the scarcity of high-quality, contextually relevant data for experimental research. The process of generating such data is often time-consuming, resource-intensive, and cost-prohibitive. Moreover, the complex relationships between variables in domain-specific tabular data necessitate a deep understanding of the underlying structures and dependencies to ensure the accurate analysis and interpretation of results.

While Generative Adversarial Networks (GANs) have shown promise in generating synthetic data, there is a critical gap in their ability to generate high-quality tabular data. This limitation presents a significant challenge in experimental settings, where the synthesis of tabular data that accurately reflects complex real-world scenarios is crucial for advancing scientific understanding and decision-making. Typical implementation of GANs may not adequately capture the intricate relationships between variables in tabular data, and they often lack the ability to generate high-resolution data conditioned on specific information.

Moreover, a critical challenge in the current implementation of GANs is the lack of

control over the generation process. For experimental research, it is imperative to generate data that aligns with specific contextual elements. Moreover, these models often lack the fine-grained control necessary to ensure that the generated data adheres to these specific conditions, making them less effective for applications that require precise and contextually adapted data synthesis.

On the other hand, transfer learning remains a largely underexplored area in GANs, despite the potential to improve learning efficiency and effectiveness. Therefore, the development of GAN architectures that can address these challenges by incorporating transfer learning and contextual conditioning would significantly enhance the utility of synthetic data in this field.

Addressing these intertwined challenges - the limited capability of GANs in generating context-specific tabular data, the need for improved control in the data generation process, and the untapped potential of transfer learning within GAN frameworks - is crucial. This research endeavors to bridge these gaps, aiming to expand the functional scope of GANs and to enhance their precision and applicability in producing contextually rich, controlled, and diverse synthetic tabular data. Such advancements are essential in a wide array of scientific and experimental domains, where the demand for accurate, context-aware synthetic data is continuously escalating.

1.2 Experimental Domains

The Context-Aware Tabular Conditional GAN (CA-CTGAN), with its advanced capabilities, is poised to significantly benefit a wide array of experimental domains. This section details the diverse types of experiments where CA-CTGAN can be effectively applied:

- **Laboratory Experiments:** In controlled laboratory settings, CA-CTGAN can be instrumental in generating high-fidelity, synthetic tabular data that mimics real experimental results. This is particularly valuable in scenarios where actual experimentation may be too costly, time-consuming, or where there are

ethical considerations. CA-CTGAN ability to simulate data under controlled conditions can greatly assist in hypothesis testing and experimental design.

- **Field Experiments:** Field experiments, often characterized by their dynamic and unpredictable environments, stand to benefit from CA-CTGAN's ability to incorporate contextual variables such as location and environmental factors into the data generation process. This capability allows for the creation of realistic data sets that mirror the complexities and variabilities encountered in field research, providing a robust tool for planning and analysis.

- **Natural Experiments:** In natural experiments, where researchers observe the effects of naturally occurring variables, CA-CTGAN can synthesize data that reflects these environmental and societal dynamics. This is particularly useful for studies where control over experimental conditions is limited, allowing researchers to explore various scenarios and their potential outcomes through synthesized data.

- **Clinical Trials:** CA-CTGAN can revolutionize data generation in clinical trials by producing synthetic patient data that simulates diverse treatment responses and patient demographics. This not only aids in the design and planning of clinical trials but also serves as a valuable tool in preliminary testing and hypothesis validation, all while adhering to ethical standards by reducing the initial reliance on human subjects.

- **Survey Research:** For survey research, CA-CTGAN ability to generate contextually rich data is invaluable. It can create synthetic responses that reflect a wide spectrum of human behavior and opinions, helping researchers to pretest surveys, understand potential response patterns, and adjust methodologies accordingly. This application is particularly beneficial in ensuring the representativeness and validity of survey instruments.

Each of these experimental domains presents unique challenges and requirements for data quality and contextual relevance. CA-CTGAN's adaptability and advanced data synthesis capabilities make it an ideal tool to address these challenges, providing researchers across various fields with a powerful means to enhance their experimental design, analysis, and overall research efficacy.

1.3 Research Questions

The primary goal of this research is to develop a novel CA-CTGAN framework (Context-Aware Conditional Tabular GAN) capable of generating high-resolution tabular data for simulating laboratory experiments across diverse domains. To achieve this, the following research questions will be addressed:

1. **Integration of Contextual Awareness:** How can the CA-CTGAN framework be designed to integrate context-specific elements effectively into a GAN architecture, thereby ensuring the generation of high-resolution synthetic data that accurately mirrors the contextual nuances across a spectrum of experimental domains?

2. **Complex Relationship Modeling:** What advanced methodologies and transfer learning techniques can be employed within CA-CTGAN's generator to capture and replicate the intricate inter-variable relationships present in domain-specific tabular data, and how do these methods contribute to the enhancement of synthetic data quality and its applicability to real-world scenarios?

3. **Control and Precision in Synthesis:** In what ways does CA-CTGAN enable precise manipulation and control of the synthetic data generation process through its auxiliary classifier and entity embedding techniques, and how does this precision affect the semantic integrity, accuracy, and utility of the generated datasets in adhering to specified context-specific requirements?

4. **Implications for Research and Application:** Considering CA-CTGAN's innovative approach to synthetic data generation, what are the broader implications for its application across diverse scientific fields, including experimental research, and how might this framework transform future research methodologies, data analysis, and decision-making processes?

1.4 Objective and Contribution

The primary objective of this research is to develop the Context-Aware Tabular Conditional GAN (CA-CTGAN), a novel framework for generating high-resolution, realistic, and contextually relevant synthetic data. This data will effectively simulate a wide range of experimental research scenarios, transcending traditional limitations. Central to achieving this objective is the integration of context-specific elements. This integration is pivotal for creating data rows that are both representative of real-world distributions and tailored to specific experimental contexts, thereby affording researchers greater control over the data generation process. Key Contributions of this research can be divided into following items:

- **Incorporation of Transfer Learning:** A significant advancement in this research is the application of transfer learning techniques to refine the initialization and training of CA-CTGAN. Utilizing a Contractive Autoencoder (CAE) to discern semantic interrelations between columns in domain-specific tabular data, the CAE learns a latent representation of the dataset. This pre-trained CAE is then integrated into the generator's architecture, replacing the traditional reliance on random noise. This method brings a nuanced context awareness to the generator, significantly improving its capability to produce contextually accurate synthetic data.

- **Auxiliary Classifier Network:** Another major contribution is the introduc-

tion of an multi-class auxiliary Classifier network, supplemented by additional loss functions in the GAN training process. This innovative approach not only preserves the semantic integrity of the synthetic data but also facilitates precise control over the data generation, ensuring accuracy and relevance for specific conditions. The Classifier network, operating alongside the generator and discriminator, evaluates the generated data, aligning conditioned and predicted labels to enhance the generation process's fidelity.

- **Advanced GAN Techniques Integration:** Further contributions include adopting techniques from established GAN variants to address specific challenges in data generation. The integration of the PacGAN [2] approach addresses mode collapse issues, promoting a richer diversity in the generated data. The application of the Wasserstein [8] loss function ensures stable and robust training, leading to superior convergence and data quality. Additionally, gradient-based optimization techniques are employed to refine the model's performance, ensuring a faithful and meaningful representation of the original data. These techniques collectively position CA-CTGAN as a state-of-the-art tool in synthetic data generation, tailored to the unique demands of our application domain.

By developing and validating the effectiveness of CA-CTGAN in producing high-quality synthetic data, this research makes a significant contribution to the fields of artificial intelligence and machine learning. The potential impacts of this research are far-reaching, poised to revolutionize experimental research in diverse fields by accelerating scientific discovery, optimizing experimental design, and substantially reducing the costs and resources associated with conducting physical experiments.

1.5 Overview of the report

This dissertation is organized into five chapters, providing a comprehensive outline of the study and its various components. Chapter 2 summarizes the foundational knowledge required to understand the proposed research. It covers the essential concepts of GANs, Autoencoders, and Transfer Learning, as well as a review of related studies in the field. The literature review serves to position the proposed CA-CTGAN within the context of existing research and highlight its unique contributions. Chapter 3 presents the architecture and design of the proposed CA-CTGAN, along with a detailed description of the training process. Additionally, it highlights the novel contributions of the proposed method, such as integrating Contractive Autoencoders by Transfer Learning to generate contextually relevant and high-resolution synthetic data for simulating laboratory experiments and integrating auxiliary Classifier network. Chapter 4 presents an in-depth analysis of the data sources, experimental setup, baseline methods, and evaluation metrics used to assess the performance of the proposed CA-CTGAN against existing approaches. Chapter 5 details the results, demonstrating the method's superior effectiveness in generating high-quality synthetic data across various scientific domains. Through comparative analysis, this chapter highlights CA-CTGAN's advancements over traditional methods, showcasing its potential to revolutionize synthetic data generation by producing more accurate, contextually relevant datasets. Chapter 5 summarizes the key findings of the research and discusses potential avenues for future work. This report offers a structured and organized explanation of developing and evaluating the Context-Aware Conditional Tabular GAN, ultimately contributing to the advancement of knowledge and technology in generating high-quality synthetic data for various scientific domains.

CHAPTER 2: BACKGROUND AND RELATED STUDIES

Generative models have emerged as a fundamental concept in the field of machine learning, serving as the foundation for numerous studies and applications. These models aim to capture the underlying structure and distribution of the data, allowing for the generation of new samples that closely resemble the original data. By learning the intrinsic patterns and dependencies within the data, generative models facilitate a deeper understanding of complex datasets, ultimately enabling researchers to tackle a wide range of tasks, from data synthesis and anomaly detection to feature learning and unsupervised representation learning.

A generative model aims to model a joint probability distribution $P(X, Y)$ where X is the observed variable, and Y is the target variable so that new data can be generated from the conditional probability $P(X|Y = y)$ of this estimated distribution that closely resembles the original data [9]. Therefore, generative models generate a distribution that matches the original distribution of $P(X|Y)$ in order to calculate $P(Y|X)$ using a classifier technique. [10] mentioned the following reasons for studying generative models:

- Using generative models, it is possible to construct high-dimensional probability distributions.
- It is possible to combine generative models with reinforcement learning.
- Semi-supervised learning can be performed reasonably well by generative models, especially GANs.
- Missing data can be imputed by generative models by generating intrinsically realistic samples.

- Generative models can generate multi-modal outputs since an input may correspond to a wide range of possible correct answers.

Over the years, various types of generative models have been proposed, each with their unique strengths and limitations. Some of the most prominent models include Variational Autoencoders (VAEs) [11], Restricted Boltzmann Machines (RBMs), and Generative Adversarial Networks (GANs). Among these, GANs have gained significant attention in recent years due to their ability to generate high-quality synthetic data across a diverse range of domains. The table 2.1 categorizes all of the introduced generative models based on different machine-learning approaches [12].

Table 2.1: Various Generative Models based on the type of learning.

	Shallow Learning	Deep Learning
Unsupervised	Gaussian Mixture Model (GMM) Hidden Markov Models (HMM) Latent Dirichlet Allocation (LDA)	Boltzmann Machines (RBM, DBM) Deep Belief Network (DBN)
Self-supervised		Variational Autoencoder (VAE)
Semi-supervised		Generative Adversarial Network (GAN)

In comparison to GANs, VAEs have a few disadvantages. Since VAEs sample directly from latent space, they oversimplify the objective task. Also, VAEs also

produce samples with significantly lower accuracy than GANs as a result of injected noise and imprecise reconstruction. The GAN architecture has undergone numerous enhancements in recent years as a result of the improvement in the architecture among the research community over the past few years [13]. In this chapter, we will delve deeper into the foundational concepts and techniques that underpin the proposed research, including GANs, Autoencoders, and Transfer Learning. Additionally, we will review relevant studies in the field to position the proposed Context-Aware Tabular Conditional GAN (CA-CTGAN) within the context of existing research and emphasize its unique contributions.

2.1 Background Knowledge

2.1.1 Generative Adversarial Networks (GANs)

GANs are characterized by two multilayer perceptron neural networks, the generator, and the discriminator. The generator is like a person who tries to make fake money, and the discriminator is like the police who try to distinguish real money from fake money. In this game, competition motivates both teams to improve their procedures until the fake money is indistinguishable from the real one [5].

The generator neural network draws a random vector z from the latent space with the distribution $p_z(z)$. The generator $G(z;\theta_g)$ then uses a parameter θ_g to map z from the latent space to the data space. Therefore, $p_g(x)$ the probability density function over the generated data is used by $G(z)$ to generate x_g. Then, the discriminator neural network $D(x;\theta_d)$ receives randomly either x_g the generated sample or x_{data} the actual sample from the probability density function over the data space $p_{data}(x)$. The discriminator neural network $D(x;\theta_d)$ is a binary classification model in which $D(x)$ returns the probability that x is derived from real data. Therefore, the output of this function is a single scalar that indicates if the passed sample is real or fake. Figure 2.1 depicts the described process and GANs architecture. θ_g and θ_d are the weights for the generator and discriminator that are learned through the optimization

procedure during training. The goal of the discriminator in training is to maximize

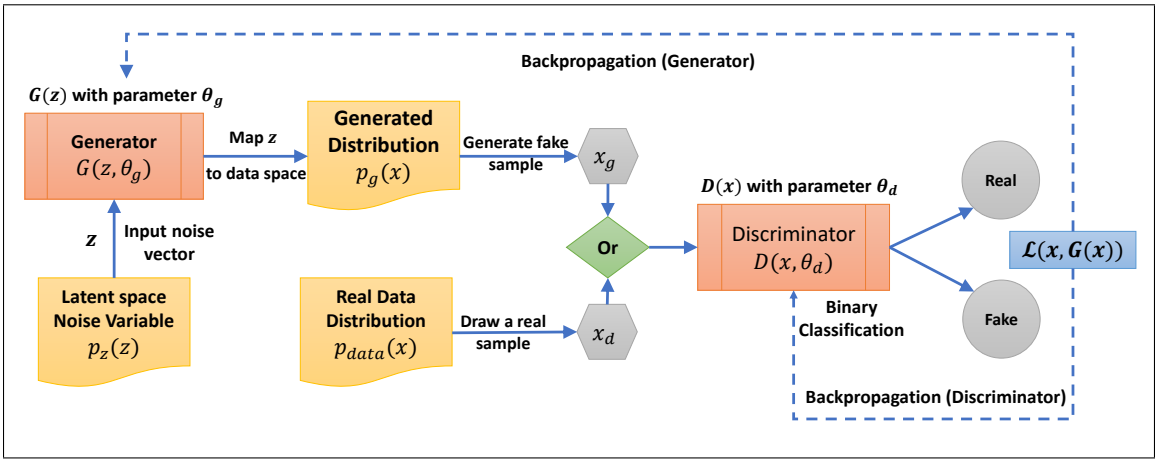

Figure 2.1: GANs process flow diagram.

the probability that a given training example or generated sample has been assigned the proper label, whereas the goal of the generator is to minimize the probability that it has detected real data. Therefore, the objective function can be expressed as a minimax value function, $V(G, D)$, which is jointly dependent on the generator and the discriminator, where:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log(D(x))] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.1}$$

The discriminator performs binary classification, which gives a value of 1 to real samples ($x \sim p_{data}(x)$) and a value of 0 to generated samples ($z \sim p_z(z)$). Therefore, in the optimal adversarial networks, p_g converges to p_{data}, and the algorithm is stopped at $D(x) = 1/2$ which means the global optimum occurs when $p_g = p_{data}$ [5].

2.1.2 Conditional Generative Adversarial Networks (CGANs)

The generating data in an unconditioned GAN is completely unmanageable in multimodal distribution. [7] introduced a conditional version of GAN that can provide generators with prior information so that they can control the generation process for different modes. Achieving this objective requires conditioning the generator and

discriminator on some additional information, y, where y can be anything from class labels to information on the distribution of data (modes). This can be done by giving the discriminator and the generator Y as an extra input layer in the form of a one-hot vector. In fact, the input noise $p_z(z)$ to the generator is not truly random if the information y is added to it, and the discriminator does not only regulate the similarity between real and generated data, but also the correlation between the generated data and input information y. Therefore, the objective function in Eq. 2.1 can be rewritten as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x|y))] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z|y)))] \qquad (2.2)$$

Figure 2.2 illustrates the structure of a CGAN and how to input information is applied during the process. A majority of applications for conditional GAN were concerned

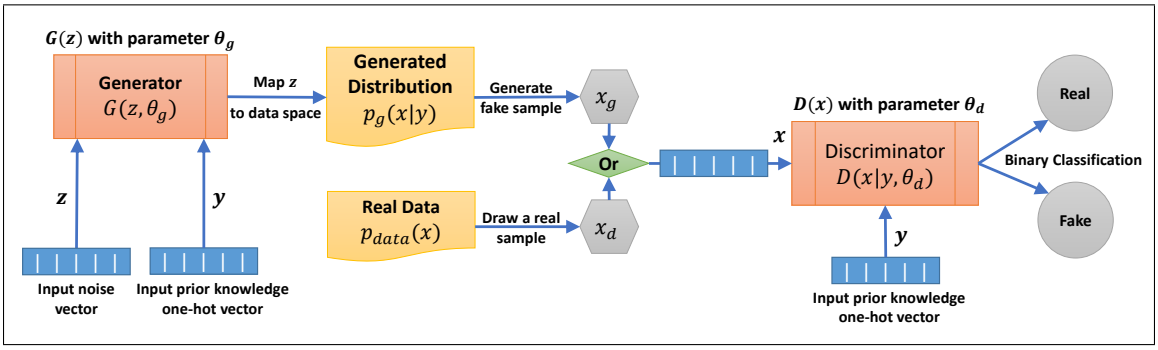

Figure 2.2: Conditional GAN process flow diagram.

with synthesizing images by giving the label for the image that should be generated. Nonetheless, in the case of tabular data, this could be the shape of data on a multi-modal distribution and can be used to inject information as prior knowledge to the generator.

2.1.3 Autoencoders

The goal of autoencoders is to reduce the dimensionality of data and reconstruct compressed data as closely as possible to its original input. Therefore, an autoen-

coder is an unsupervised neural network that learns how to identify the most efficient encoder/decoder that retains the maximum amount of information during encoding and incurs the least loss during decoding. There are three layers in autoencoders, the input layer X, the latent space (bottleneck) Z, and the output layer $\hat{X}$. The inputs are encoded into feature-extracted representations in latent space Z, and the output $\hat{X}$ is generated following the decoding of vector Z. Figure 2.3 represents an autoencoder architecture. An autoencoder's objective is to reduce construction error between real and reconstructed data. An L2 loss function is used to calculate the loss, and then the error is backpropagated through the network, and the weights are updated accordingly [12].

$$L = \| X - \hat{X} \|^2 = \| X - G(Z) \|^2 = \| X - G(F(X)) \|^2 \tag{2.3}$$

Where $F(.)$ is the encoding function that encodes X into the latent space, and $G(.)$ is the decoding function that decodes $\hat{X}$ from latent space Z. The autoencoder is

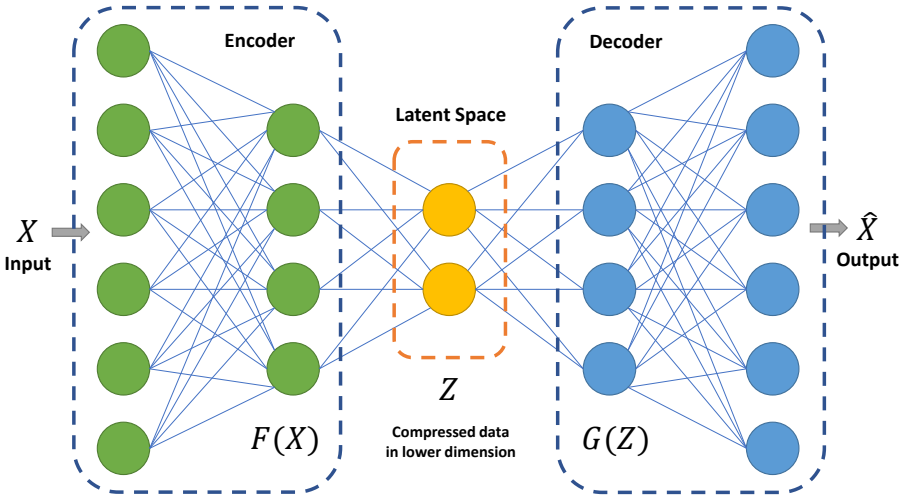

Figure 2.3: The network architecture for a normal autoencoder.

simply taught to encode and decode with the minimum loss so that the decoder cannot generate new data regardless of the method used to create the latent space.

Therefore, the objective function can be expressed as follows:

$$\mathcal{L}_{AE} = \sum_{x \in D_n} L(x, g(f(x))) \tag{2.4}$$

Autoencoders can be classified based on the type of architecture and the training method used.

Based on architecture, autoencoders can be classified as follows:

- **Convolutional Autoencoder:** Uses convolutional layers in the encoder and decoder to handle the spatial structure of the input data, typically used for image data [14].
- **Recurrent Autoencoder:** Uses recurrent layers in the encoder and decoder to handle the temporal structure of the input data, typically used for sequential data [15].
- **Variational Autoencoder:** A probabilistic autoencoder that learns a distribution over the latent space instead of a fixed point. It is used for data generation and dimensionality reduction [11].
- **Denoising Autoencoder:** Trained to remove noise from the input data, typically by adding noise to the input data and reconstructing the original input [16].
- **Adversarial Autoencoder:** A generative model that uses adversarial training to learn a distribution over the latent space that is similar to the true distribution of the data. It is used for data generation and dimensionality reduction [17].
- **Contractive Autoencoder:** Designed to learn a robust and invariant representation of the input data by adding a regularization term to the loss function

that penalizes the sensitivity of the network's output to small changes in the input [18].

Based on the training method, autoencoders can be classified as follows:

- **Supervised Autoencoder:** The standard autoencoder that is trained using labeled data.
- **Unsupervised Autoencoder:** Trained using unlabeled data.
- **Semi-Supervised Autoencoder:** Trained using a combination of labeled and unlabeled data.
- **Contrastive Autoencoder:** Uses a contrastive loss function during training to encourage similar inputs to be mapped close together in the latent space [19].
- **Sparse Autoencoder:** Trained to learn sparse representations of the input data by adding a penalty for the number of active neurons in the latent space [20].
- **Deep Autoencoder:** Autoencoders with multiple hidden layers in the encoder and decoder, typically used for learning hierarchical representations of the input data.

These different types of autoencoders have different strengths and weaknesses depending on the application. By understanding the different types of autoencoders and their properties, researchers can choose the most suitable architecture and training method for their specific problem.

2.1.4 Conteractive Autoencoder

In order to incorporate the contextual information of the dataset and provide a better initialization of the generator, this research proposes utilizing a Contractive Autoencoder (CAE) [18] and transferring a pre-trained model as an alternative to

relying on random noise for initializing the generator's input. The CAE is a specific type of autoencoder that is designed to capture the complex relationships and structures within the data by learning a latent representation in a more robust manner. A Contractive Autoencoder consists of an encoder, which compresses the input data into a lower-dimensional latent representation, and a decoder, which reconstructs the original data from the latent representation. The primary difference between a standard autoencoder and a CAE lies in the training process. In a CAE, a contractive penalty is added to the loss function, which encourages $f(x)$ to be less sensitive to small variations in the input data x. This penalty term is computed based on the Frobenius norm of the Jacobian matrix $J_f(x)$ of the encoder's outputs with respect to its inputs. This sensitivity penalty term is derived by summing the squares of all partial derivatives of the features extracted from the input dimensions when input x is mapped by encoding function f to hidden representation h as follows:

$$\| J_f(x) \|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2 \tag{2.5}$$

By minimizing this penalty, the CAE learns to map similar inputs to similar latent representations, making the model more robust and better equipped to capture the semantic relationships between columns in domain-specific tabular data. Thus, the objective of CAE is to minimize the following function:

$$\mathcal{L}_{CAE} = \sum_{x \in D_n} L(x, g(f(x))) + \lambda \| J_f(x) \|^2 \tag{2.6}$$

Where the strength of regularization is controlled by the positive hyperparameter λ. As shown in Figure 2.4, the first term penalizes sensitivities to reconstructions, while the second term penalizes how much of the movement of the Jacobian matrix comes from x. This basically pushes the energies up in directions that are not required for reconstruction.

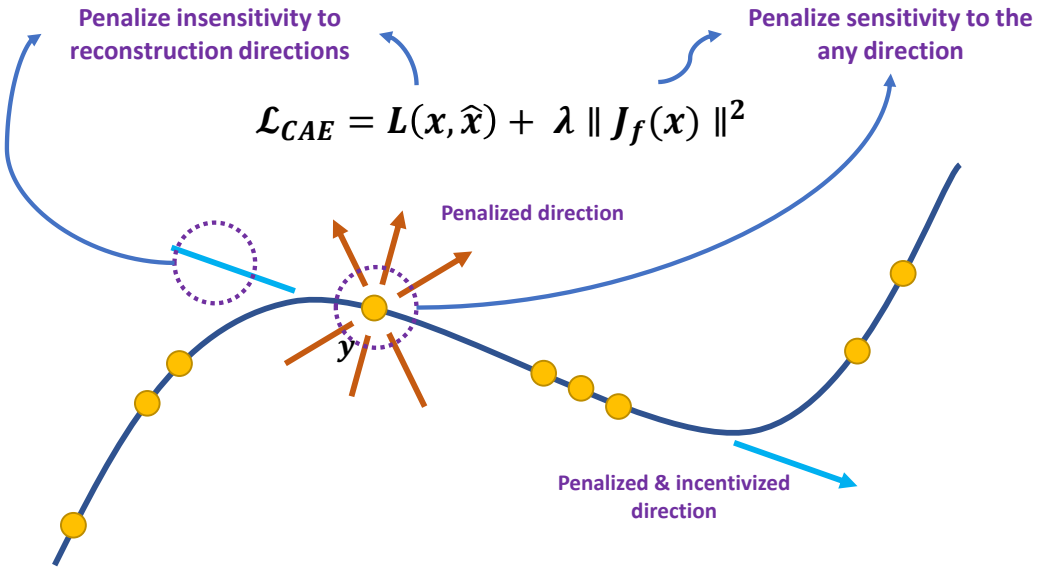

Figure 2.4: Visual representation of Conteractive Autoencoders.

2.1.5 Transfer Learning

Transfer Learning is a widely-used technique in the field of machine learning and deep learning that leverages the knowledge gained from one task to improve the learning process in another, related task. The primary motivation behind transfer learning is to utilize the existing knowledge to reduce the amount of training data, computational resources, and time required for learning new tasks, particularly when the available data for the target task is limited or when the tasks share similar underlying structures.

In the context of deep learning, transfer learning typically involves the reuse of pre-trained neural network models that have been trained on large-scale datasets. These pre-trained models serve as starting points for the target task, providing a set of learned features or initial weights that can be fine-tuned or adapted to the new task. This process often results in improved performance and faster convergence compared to training the model from scratch.

There are several approaches to transfer learning, which can be broadly categorized

into two main types: feature extraction and fine-tuning. In feature extraction, the pre-trained model is used as a fixed feature extractor, and its learned features are fed into a model tailored to the target task. In fine-tuning, the pre-trained model's parameters are fine-tuned or updated using the target task's data, allowing the model to adapt to the specific nuances and characteristics of the new task [21].

Transfer learning has been successfully applied in various domains, such as computer vision, natural language processing, and speech recognition, where large-scale pre-trained models have demonstrated remarkable improvements in performance and efficiency over training from scratch. Some notable examples of transfer learning applications include the use of pre-trained convolutional neural networks (CNNs) for image classification and object detection tasks, as well as the adoption of pre-trained transformer-based models, such as BERT and GPT, for a wide range of natural language processing tasks [22].

In this research, transfer learning is employed to bring the context-awareness into the generator of the proposed Context-Aware Conditional Tabular GAN (CA-CTGAN). This approach not only leverages the learned semantic relationships between the columns of the domain-specific tabular data but also improves the initialization and training of the generator.

2.2 Tabular Data Generation Challenges

According to the data structure and data types of tabular data, the challenges associated with data generation can be categorized into the following significant groups [23].

2.2.1 Data type

It is challenging to generate data that is representative of the entire data table due to the difference in data types. For instance, different activation functions on output are required for the generative models since relational database tables include

numerical, categorical, ordinal, and mixed data types. As an example of a mixed data type, the financial database contains columns for loan debts, where a loan holder may have no debt or debt with a positive value [1]. In data analysis, it can be defined as categorical data using a step function, but in reality, it is continuous data. In this regard, a data generator must be able to detect these types of data in order to avoid adverse effects on the interpretation of the data. The several types of data used in tabular data are broken down in Table 2.2. Mentioning that textual data types are not the case of this study and are therefore ignored here.

Table 2.2: The data types that can be used in a tabular data table.

Data types		
Numerical	Continuous	Numeric intervals on the real number without finite set of values
	Discrete	Finite, countable set of integer number
	Mixed	Numeric, but considered as categorical based on the different range
Categorical	Binary	One-hot encoded
	Textual	One-hot encoding needed
	Numeric	Treats like textual and numbers are meaningless.
Ordinal	Numeric	Numeric categories with a clear ordering (like 1-5 rating)

2.2.2 Bounded continuous columns

Continuous Column C_i is bounded if there are two numbers a and b in which for all $x \in X$ $a \leq x \leq b$. tabular data generator presents a challenge with bounded continuous columns because generating realistic data for these columns requires sampling from a distribution that accurately represents the real data and ensuring that the generated data falls within a specified range. For example, the spent value of a credit card can range from zero to a predetermined amount and generating data from the same distribution may result in generating data outside the bounded values, which

would not be meaningful or useful.

2.2.3 Non-Gaussian Distribution

When dealing with the non-Gaussian distributions that are common in real-world datasets, the assumption of normality often fails in the field of tabular data generation. Such distributions may be multimodal, containing several peaks or modes, which reflects the complexity of underlying data-generating processes. For instance, the distribution of incomes in a socio-economic dataset could exhibit multiple modes, corresponding to different socio-economic classes. Traditional synopsis generation techniques may inadequately capture the multi-modes structure of such distributions, leading to the missing of entire modes. This results in a generated synopsis that fails to represent segments of the population within the original dataset [23].

Moreover, the presence of long-tailed distributions poses additional challenges [1]. These distributions are characterized by a proliferation of infrequent events, such as a customer purchase history where a vast majority of customers make infrequent purchases, while a minor fraction exhibits high purchase frequencies. Synthesizing data from such a distribution requires not only capturing the frequent low-occurrence events but also accurately representing the rare high-occurrence instances. The conventional methods may struggle with this, often either over-representing the tail and creating too many rare events or under-representing it, thus failing to capture the true nature of the underlying data. This misrepresentation can skew the synopsis, rendering it less effective for use in decision-making processes where an understanding of rare events is critical.

2.2.3.1 Imbalance Categorical Column

In tabular data generation, the handling of imbalanced categorical columns presents a significant challenge [23]. Categorical variables in real-world datasets frequently show a skewed distribution in terms of the frequency of occurrence across categories.

The presence of such a disparity indicates that minority categories make only a small contribution to the overall distribution of data, which may result in their under-representation in the generated synopsis. The process of creating synopses is influenced by a lack of representation of certain classes, resulting in a bias towards the majority class due to its higher statistical likelihood. For instance, consider a customer gender column in a retail database with a pronounced imbalance, where 'male' customers vastly outnumber 'female' customers. Generated data from this distribution might reflect this skew, resulting in a synthetic dataset dominated by 'male' entries. However, this skew inaccurately portrays the significance of the 'female' category, which, despite its smaller size, may carry substantial weight in consumer behavior analysis.

2.2.4 Semantic relationship

Tabular data often includes complex semantic relationships that are not easily understood through standard statistical analysis [24]. These relationships can exist between categorical and numerical columns alike and are crucial for maintaining the integrity and usefulness of the generated data. Identifying and encoding such relationships is a challenge due to the heterogeneity of domain-specific constraints and the complex nature of the inter-column dependencies which may not be amenable to simple rule-based generalizations. For instance, semantic relationships may determine that certain numerical values possess validity solely when paired with specific categorical entries, imposing a constraint-based association. Alternatively, a rule-based linkage could suggest a probabilistic co-occurrence pattern between different fields in the data. Hence, it is imperative for a comprehensive process of generating synopses to include mechanisms that can deduce these complex relationships, which can be multi-faceted and deeply embedded within the structure of the data. A failure to do so not only compromises the authenticity of the synthesized data but also limits the operational relevance of the synopsis, as it could lead to the generation of implausi-

ble or inconsistent records that do not adhere to the real-world rules and constraints governing the dataset. Figure 2.5 represents two examples of generated samples from a table that the model should reject semantically. To generate a representative data, the city must be properly associated with the state, and the joined column cannot precede the founded column.

Team	City	State	Capacity	Founded	Joined
Charlotte FC	Charlotte	North Carolina	38,000	2019	2022
Los Angeles FC	Los Angeles	California	22,000	2014	2018

(a) Constraint-based rejected samples

Team	City	State
Charlotte FC	Los Angeles	North Carolina

(b) Rule-based rejected samples

Team	Founded	Joined
Los Angeles FC	2018	2014

Figure 2.5: Each soccer team in the table corresponds to a particular location and has a specific capacity, foundation year, and year of entry into MLS. (a) and (b) shows two examples of constraint-based and rule-based sample rejection during the data synopsis generation process.

2.3 Related Studies

Generative Adversarial Networks (GANs) were initially introduced in the field of computer vision, where they are predominantly used for processing image data through Convolutional Neural Networks (CNNs). Nevertheless, GANs have also demonstrated their capability to generate tabular data. This section aims to present an overview and categorization of various GAN-based methods specifically designed for tabular data generation, followed by a detailed analysis of the most promising state-of-the-art variants in order to identify how these GAN adaptations address the challenges associated with tabular data. Most existing GAN-based solutions for tabular data generation have been developed with the primary objective of adhering to data privacy regulations and preventing data leakage during data sharing or synthetic data generation for imputation and augmentation. However, when generating laboratory experiment data, the focus shifts towards producing realistic data that closely

resembles real observations rather than merely generating synthetic data that adheres to privacy constraints. The challenges of generating tabular data using GANs have been tackled in a limited number of publications since 2017.

The purpose of this section is to introduce the most promising GAN variants for tabular data generation and provide a classification of these proposed solutions based on the specific challenges they address in generating tabular data. This analysis will aid in identifying the key advancements in the field and guide the development of the proposed Context-Aware Tabular Conditional GAN (CA-TCGAN) for simulating laboratory experiments.

Choi et al. [25] proposed the medical Generative Adversarial Network (medGAN) to generate realistic synthetic patient records based on real data as inputs to protect patient confidentiality to a significant extent. The medGAN generates high-dimensional, multi-label discrete variables by combining an autoencoder with a feedforward network, batch normalization and shortcut connections. With an autoencoder, flow gradients are able to end-to-end fine-tune the system from discriminator to decoder for discrete patient records. The medGAN architecture uses MSE loss for numerical columns and Cross-Entropy loss for binary columns, and ReLU activation function for both encoder and decoder networks. The medGAN uses the pre-trained autoencoder to generate distributed representations of patient records rather than directly generating patient records. In addition, it provides a simple and efficient method of dealing with mode collapse when generating discrete outputs using minibatch averaging. Figure 2.6 shows medGAN architecture and defines the autoencoder role in training process.

The generator cannot generate discrete data because it must be differentiable. Mottini et al. [26] proposed a method for generating realistic synthetic Passenger Name Records (PNRs) using Cramer GANs, categorical feature embedding, and a Cross-Net architecture for the handling of this issue (categorical or numerical with null values).

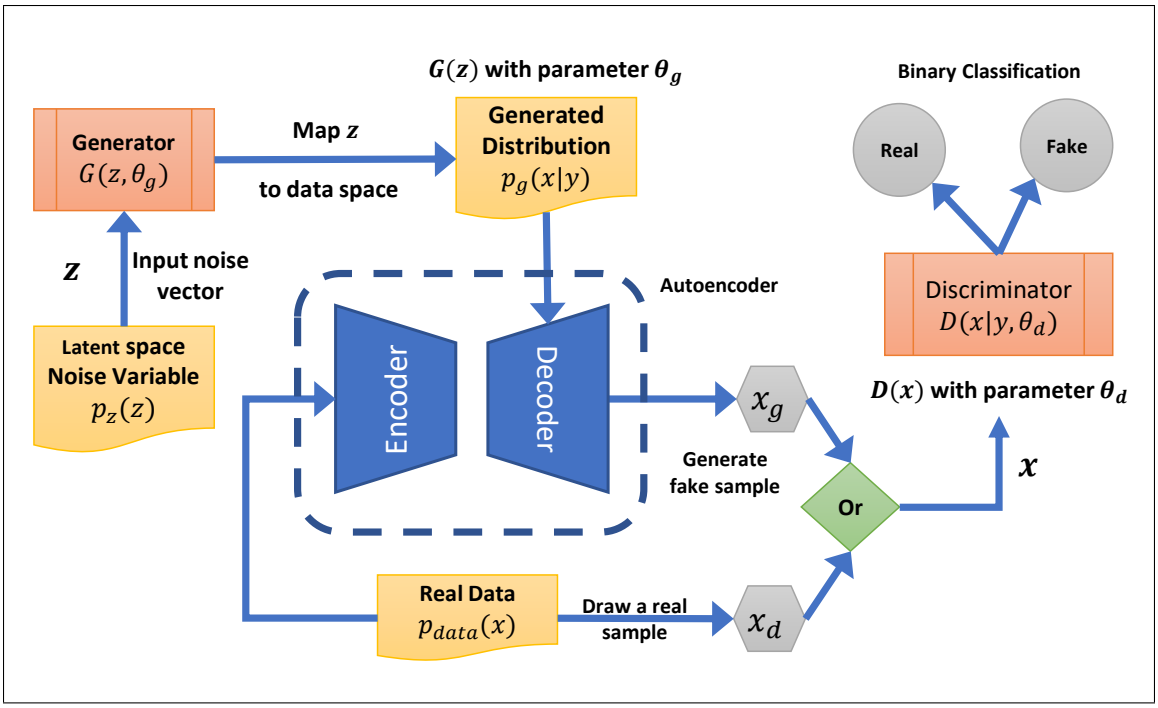

Figure 2.6: medGAN architecture: Discriminator utilizes autoencoder (which is learned by real data) to receive decoded random noise variable

As opposed to simply embedding the most probable category, they used the weighted average of the embedded representation of each discrete category. The embedding layer is shared by the generator and discriminator, resulting in a fully differentiable process as a result of this continuous relaxation. For handling null values, they are substituted with a new category in categorical columns. However, continuous columns fill null values with a random value from the same column and then a new binary column is inserted with 1 for filled rows and 0 otherwise. These additional binary columns are encoded like category columns. It should be noted that in this architecture, both the generator and discriminator consist of fully connected layers and cross-layers. Also, except for the last layer (Sigmoid), all layers of the generator use leaky ReLU activations for numerical features and Softmax for categorical features. However, discriminator uses leaky ReLU activations in all but the last layer (linear). Neither batch normalization nor dropout is used in this architecture like Wasserstein and Cramer GANs [27]. Data pre-possessing in this algorithm is depicted in Figure

2.7. As indicated, discrete values will be embedded using the embedding matrix,

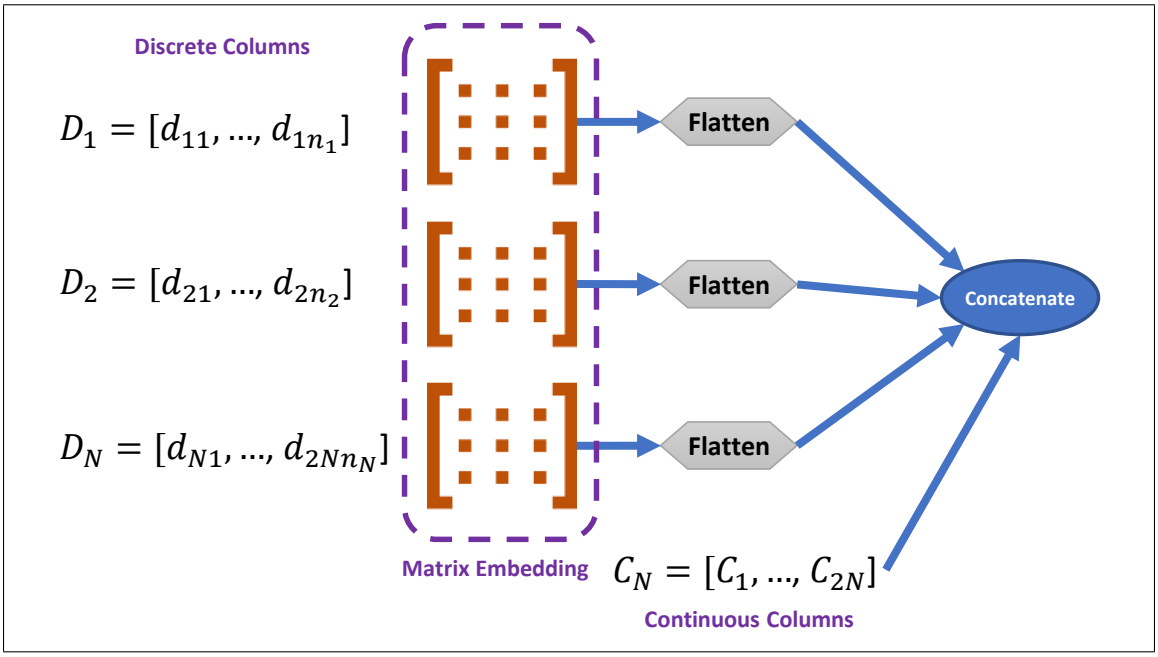

Figure 2.7: Pre-processing input data before feeding the discriminator in PNR-GAN

followed by the concatenation of them with continuous columns of input data.

Table-GAN is a method proposed by Park et al. [28] that uses GANs to create fake tables that are statistically similar to the original tables but are resistant to re-identification attacks and can be shared without exposing private information. Table-GAN supports both discrete and continuous columns and is based on Deep Convolutional GANs (DCGANs) [29]. Besides the generator and discriminator with multilayer convolutional and deconvolutional layers, the table-GAN architecture also includes a classifier neural network with the same architecture as the discriminator. However, it is trained using ground-truth labels from the original table to increase the semantic integrity of the generated records. Information loss and classification loss are two additional types of loss that are introduced during the backpropagation process. The purpose of these functions is to maintain a balance between privacy and usability while ensuring the semantic integrity of the real and generated data. The information loss compares the mean and standard deviation of real and generated

data to measure the discrepancy between them and determine whether they have statistically the same features from the perspective of the discriminator or not, and the classification loss measures the difference between how a record is labeled and how the classifier predicts it should be labeled. Figure 2.8 is a representation of the

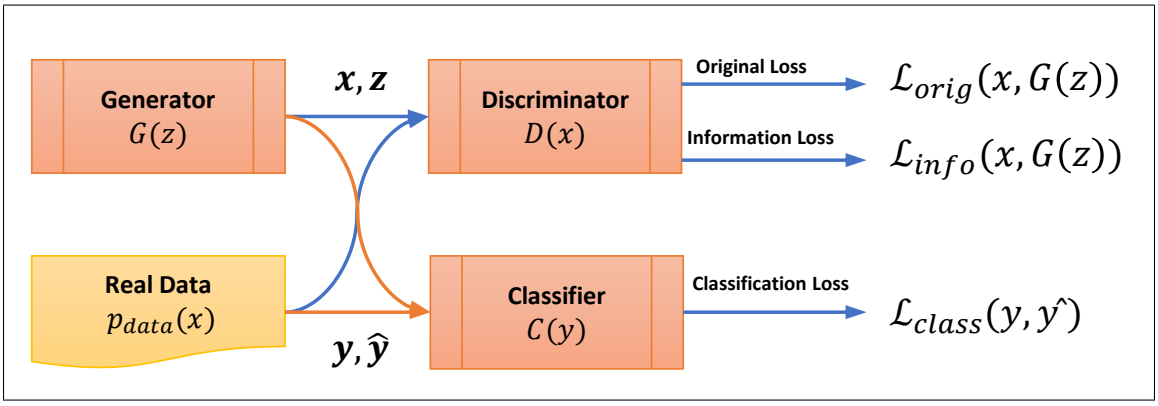

Figure 2.8: Loss functions representation in table-GAN architecture.

loss functions in the table-GAN architecture.

Xu and Veeramachaneni [23] developed TGAN, which is a synthetic tabular data generator for data augmentation that can take into account mixed data types (continuous and categorical). TGAN generates tabular data, column by column, using a Long-Short Term Memory (LSTM) network with attention. The LSTM will generate each continuous column from the input noise in two steps. First, it generates a probability that the column comes from mode m, and then normalizes the column value based on this probability. TGAN penalizes GAN's original loss function by adding two KL-divergence terms between generated and real data for continuous and categorical columns separately. Therefore, generator will be optimized as follow:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim\mathcal{N}(0,1)}[\log(D(G(z)))] + \sum_{i=1}^{N_c} KL(u'_i, u_i) + \sum_{i=1}^{N_d} KL(d'_i, d_i) \qquad (2.7)$$

Where u'_i and u_i are probability distribution over continuous column c_i for generated and real data, respectively, d'_i and d_i are probability over categorical column d_i using

softmax function for generated and real data respectively, N_c is number continuous columns, and N_d is number of categorical columns. Xu et al. [30] also proposed a conditional version of TGAN, named CTGAN, for addressing data imbalance and multimodal distribution problems by designing a conditional generator with training by sampling strategy to validate the generator output by estimating the distance between the conditional distribution over generated and real data.

Zhao et al. [1] introduced CTAB-GAN with the ability to encode the mixed data type and skewed distribution of input data table. CTAB-GAN utilizes conditional generator, information and classification loss functions derived from table-GAN, as well as CNNs for both generator and discriminator functions. Since CNNs are effective at capturing the relationship between pixels within an image, therefore they can be employed in enhancing the semantic integrity of created data. However, in order to prepare data tables for feeding CNN, rows are transformed into the nearest square $d \times d$ matrix, where $d = Ceil(\sqrt{N_c + N_d})$, N_c and N_d are number of continuous and categorical columns respectively in a row of the data table, and then, the extra cells values $(d \times d - (N_c + N_d))$ are padded with zeros.

It is difficult for GANs to control the generation process of data-driven systems; therefore, integrating prior knowledge about data relationships and constraints can assist the generator in generating synopses that are realistic and meaningful. In order to implement this, DATGAN [31] incorporates expert knowledge into GANs generator by matching the generator structure to the underlying data structure using a Directed Acyclic Graph (DAG). Using a DAG, the nodes represent the columns of a data table, while the directed links between them allow the generator to determine the relationship between variables so that one column's generation influences another. It means if two variables have no common ancestors, they will not be correlated in the generated dataset. In relational databases, there is no particular order in which columns appear in data tables. Nevertheless, the DAG enables data tables to have a

specific column order based on their semantic relationship.

Generation challenges can be classified into three categories based on their proposed solutions: Data Transformation, which addresses data type issues; Distribution Matching, which addresses ranges and distributions of data; and Conditional and Informed Generator, which addresses imbalance classes, semantic relationships [6].

2.3.1 Data Transformation

Mode normalization is capable of detecting modes of data by assigning samples to different modes and then normalizing each sample based on the corresponding mode estimator [32]. To deal with multimodal distribution for continuous columns, mode-specific normalization is introduced in TGAN [23]. Using this algorithm, first, the number of continuous columns' modes is calculated using Gaussian kernel density estimation. Then, Gaussian Mixture Model (GMM) can be employed to efficiently sample values from a distribution with multiple modes by clustering the values of continuous columns (C_i). In other words, the weighted sum of the Gaussian distributions over C_i can represent the multimodal distribution over it. A normalized probability distribution over m Gaussian distributions can then be used to represent each continuous column so that each column can be clustered into m fixed Gaussian distributions. As a result, if there are less than m modes in one column, then the probability of that mode is high, and for the rest, it is close to zero. However, in CTGAN [30], first, a Variational Gaussian Mixture model (VGM) should be applied to each continuous column (C_i) in order to fit a Gaussian mixture and find the number of modes (m). Then, a one-hot vector ($\beta_{i,j}$) indicates to which mode a given value belongs, and a scalar ($\alpha_{i,j}$) serves as the value itself within that mode. For the learned Gaussian mixture for column C_i with m modes, the following equation is given:

$$\mathbb{P}_{C_i}(c_{ij}) = \sum_{k=1}^{m} w_k \mathcal{N}(c_{ij}; \mu_k, \sigma_k) \tag{2.8}$$

where $c_{i,j}$: value of j^{th} row from i^{th} column, μ_k and σ_k : the mean and standard deviation of Gaussian distribution for k^{th} mode, and w_k is the weight of k^{th} mode. For each value, the probability density ρ of k^{th} mode is:

$$\rho_k = w_k \mathcal{N}(c_{ij}; \mu_k, \sigma_k) \tag{2.9}$$

Therefore, each value can be normalized according to the mode with highest probability. As an example, the values of α and β related to column $c_{i,j}$ in k^{th} mode will be:

$$\alpha_{i,j} = \frac{c_{i,j} - \mu_k}{\delta \sigma_k} \ , \quad \beta = [0, 0, \ldots, \underbrace{1}_{k^{th} \text{ element}}, \ldots, 0, 0]. \tag{2.10}$$

where δ is a parameter specified by the modeller.

For categorical columns D, the situation is different; TGAN [23] stated to convert these columns (d_{ij}) to a representation using one-hot encoding with added noise ($Uniform(0, \gamma)$, γ is an arbitrary number). To achieve this, after creating the one-hot vector, noise will be added to each element, and the resulting representation will be renormalized. Therefore, each data row can be represented by a concatenation of continuous and categorical columns as follows:

$$row_j = \alpha_{1,j} \oplus \beta_{1,j} \oplus ... \oplus \alpha_{N_c,j} \oplus \beta_{N_c,j} \oplus d_{1,j} \oplus ... \oplus d_{N_d,j} \tag{2.11}$$

where $d_{i,j}$ is one-hot representation of a categorical column, N_c is number of continuous columns and N_d is number of categorical columns D_i.

As previously discussed, columns can be considered mixed if they contain both categorical and continuous values or continuous values with null values. The encoding process for continuous and categorical columns in CTAB-GAN [1] is exactly the same as CTGAN [30] by defining α and β. However, in mixed-type columns, the encoder is defined so that each column is considered a concatenation of value-mode pairs, where

the categorical part of values takes zero for α and is treated as continuous.

2.3.2 Distribution Matching

In order to generate synopses with the same distribution as the underlying distribution, the training algorithm should penalize the generator. Information loss [28] helps generator to generate synopses statistically closer to the real one. It utilizes the statistical characteristics $\mathcal{L}_{mean}$ (first-order statistics, Eq. 2.12) and $\mathcal{L}_{sd}$ (second-order statistics Eq. 2.13) of the extracted features prior to the classifier in the discriminator to penalize the generator for the discrepancy between real and generated data. This makes sense because the extracted features are used to determine the binary decision of the discriminator.

$$\mathcal{L}_{mean} = \| \mathbb{E}[\boldsymbol{f}_x]_{x \sim p_{data}(x)} - \mathbb{E}[\boldsymbol{f}_{G(z)}]_{z \sim p_z(z)} \|_2 \tag{2.12}$$

$$\mathcal{L}_{sd} = \| \mathbb{SD}[\boldsymbol{f}_x]_{x \sim p_{data}(x)} - \mathbb{SD}[\boldsymbol{f}_{G(z)}]_{z \sim p_z(z)} \|_2 \tag{2.13}$$

Where $\boldsymbol{f}$ represents features, $\mathbb{E}[\boldsymbol{f}]$ is the average and $\mathbb{SD}[\boldsymbol{f}]$ is the standard deviation of features over all rows in the data table. The Euclidean norm is used to measure the discrepancy between two terms. As we discussed before, table-GAN [28] was developed to protect confidential data privacy when it is shared with the public. As a result, it should be possible to control the similarity of generated data with real data during the generating process. To this end, information loss for the generator is demonstrated as follows:

$$\mathcal{L}^G_{info} = \max(0, \mathcal{L}_{mean} - \delta_{mean}) + \max(0, \mathcal{L}_{sd} - \delta_{sd}) \tag{2.14}$$

Where δ is a threshold indicating a quality degradation of generated data and max(.) represents the hinge-loss that is zero until δ is reached. However, in AQP, it is not necessary to meet this threshold in order to generate realistic data synopses.

DATGAN [31] uses the improved version of the Wasserstein loss function in WGAN [8] in addition to the Vanilla GAN loss function with gradient penalty [33] and also add the KL-divergence as an extra term to the original loss function. Both of these terms aim to minimize the difference between the probability distributions of real and generated data. WGAN employs an alternative method of training the generator to better approximate real data distribution. This approach replaces the discriminator model with a critic that scores the degree to which a data sample is real or fake rather than using the discriminator as a classifier. Therefore, WGAN considers discriminator output as a scalar score instead of a probability, and Wasserstein loss ensures a greater difference between the scores for real and generated data. As a result, it can prevent vanishing gradients in the generator models. However, the WGAN's primary problem is that it must clip the weights of the critic in order to enforce the Lipschitz constraint. This issue can be addressed by adding a gradient penalty to the critic. Eq. 2.15 shows the Wasserstein objective function, and Eq. 2.16 shows the same with a penalty on the gradient norm for random samples $\hat{x} \sim p_{\hat{x}}$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))] \tag{2.15}$$

$$L_W = \mathbb{E}_{z \sim p_z(z)}[D(G(z))] - \mathbb{E}_{x \sim p_{data}(x)}[D(x)] + \lambda\, \mathbb{E}_{\hat{x} \sim p_{\hat{x}}}[||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2] \tag{2.16}$$

where λ is a parameter defined by the modeler and $\hat{x}$ sampled from $G(z)$ and x.

2.3.3 Conditional and Informed Generator

Imbalances in categorical columns can cause inaccuracies when generating synopses and may result in the generator not being trained to match the distribution of the real data. In CTGAN [30], the conditional generator is introduced (using training-by-sampling) as a solution to this problem. To this aim, the generated value can be interpreted as a conditional distribution of rows given the value of an imbalanced cat-

egorical column. Therefore, the original distribution can be reconstructed as follows:

$$P_g(row|D_i = k) = P(row|D_i = k) \implies P(row) = \sum_{k \in D_i} P_g(row|D_i = k)P(D_i = k) \tag{2.17}$$

where k is a value in i^{th} categorical cloumns D_i. For the implementation of this solution, a conditional vector consisting of a mask vector that represents the address of the table value (column and corresponding row value) is required. This conditional vector does not guarantee the feed-forward pass obtains the correct value based on the mask vector M; instead, the suggested approach penalizes the conditional generator's loss by averaging the cross-entropy between the generated $\hat{M}_i$ and the expected conditional vector M_i over all instances of the batch. The generator loss can be expressed as follows:

$$\mathcal{L}_G = \mathbb{E}[H(M_i, \hat{M}_i)] \tag{2.18}$$

Where H(.) is cross-entropy between two values. As a result, the generator learns to replicate the masked value in the generated row during training. The conditional vector for a data table with N categorical columns is the direct sum of all mask vectors (M) across each column D_i, where for each value $c_{i,j}$:

$$M_i = \left\{ \begin{array}{ll} 1 & \text{if } j^{th}\text{value} \\ 0 & \text{the rest} \end{array} \right\}, \; cond = M_1 \oplus \ldots \oplus M_N \tag{2.19}$$

In fact, generator loss allows the generator to learn to produce the same classes as the given conditions. Mask vectors (M_i) are initialized with 0 for each categorical column (D_i) during the conditional generator procedure. Then, a column is chosen at random, and the Probability Mass Function (PMF) is applied to the column's range of categories. According to PMF, one category is then picked, and its value in the corresponding mask vector is changed to 1. Finally, the conditional vector is

formed, and the generator is able to generate a synthetic row for the given categorical column. Figure 2.9 represents a mask vector generation process for a data table with N_d categorical columns when generator is conditioned for j^{th} category of i^{th} categorical column.

It has been discussed previously that columns in a table may have a meaningful

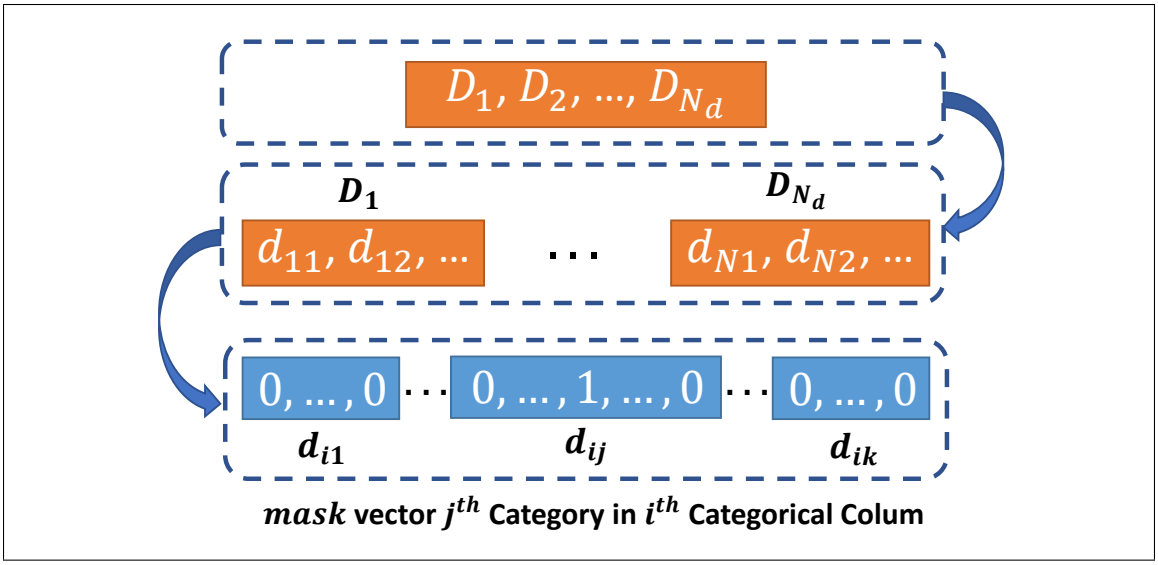

Figure 2.9: Following vectorization of categorical columns, all vectors will be initiated by 0, then j^{th} category from i^{th} column will be selected, and the value of the corresponding element will be changed to 1.

relationship with one another. CTAB-GAN [1] utilizes a classifier neural network using auxiliary classifier GAN (AC-GAN) [34] that is a conditional GAN type that requires the discriminator to predict the class label $c \sim p_c$ of generated data as well as the realness classifier. In AC-GAN, the generator generates a new sample using noise z and a class label c, while the discriminator provides both a probability distribution over sources $P(S|X)$ and a probability distribution over class labels $P(C|X)$. The objective function contains the following terms:

$$\mathcal{L}_S = \mathbb{E}_{x \sim p_{data}(x)}[\log P(S = real)] + \mathbb{E}_{x \sim p_z(z)}[\log P(S = fake)] \tag{2.20}$$

$$\mathcal{L}_C = \mathbb{E}_{x \sim p_{data}(x)}[\log P(C = c_{real})] + \mathbb{E}_{x \sim p_z(z)}[\log P(C = c_{fake})] \tag{2.21}$$

Where $\mathcal{L}_S$ is likelihood of predicting the correct source, $\mathcal{L}_C$ is likelihood of predicting the correct class, and c is a class label. Discriminator is trained to maximize $\mathcal{L}_C + \mathcal{L}_S$ and generator is trained to maximize $\mathcal{L}_C - \mathcal{L}_S$. These objective functions allow the training procedure to generate data according to a specific type of data, while the discriminator must predict the class label of the generated data and determine whether or not it is real. As a result of this, the classifier loss (Eq. 2.22) will be added to the generator in CTAB-GAN to increase the semantic integrity of generated records and penalizes generator where the combination of columns in a data row is semantically incorrect.

$$\mathcal{L}^G_{class} = \mathbb{E}_{z \sim p_z(z)}[|l(G(z)) - C(fe(G(z)))|] \tag{2.22}$$

where $l(.)$ returns the target label and $fe(.)$ returns the input features of a given row. As mentioned before, DATGAN [31] uses DAG to control the generation process based on semantic relationships and correlations between columns. According to the constructed DAG, each column and its sequence are represented by Long Short Term Memory (LSTM) cells. Therefore, by providing the generator with prior knowledge, DAG decreases the GAN's capacity to overfit noise in the training process and enables the GAN to produce more accurate data by using these noises more efficiently. Inputs and outputs of LSTM cells should be modified in accordance with the GAN architecture. Inputs can be expressed as follows:

$$i_t = a_t \oplus f_{t-1} \oplus z_t \tag{2.23}$$

where z_t is a tensor of Gaussian noise, which is the concatenation of the noise from the source nodes at each node of the DAG. f_{t-1} is the transformed output of previous tensor (h_{t-1}). For the purposes of determining which previous cell outputs are relevant to a node input, a_t represents a weighted average of all ancestor LSTM

outputs. Therefore, a_t and the z_t are defined based on all ancestors of the current node. Data input into DATGAN architecture (generated and real data) should be encoded into [-1,1] or [0,1] using techniques described in the "Data Transformation" section. Additionally, for categorical columns, generators produce probability over each class, making it easy for a discriminator to differentiate between real and created values. Therefore, DATGAN recommends using one-sided label smoothing for the default loss. It means the categorical 0,1 vectors are introduced with additive uniform noise and then rescaled to [0,1] bound vectors. Figure 2.10 illustrates the DATGAN process flow diagram, including the data transformer and label smoothing. In this algorithm, DAG is generated manually; therefore, semantic relationships

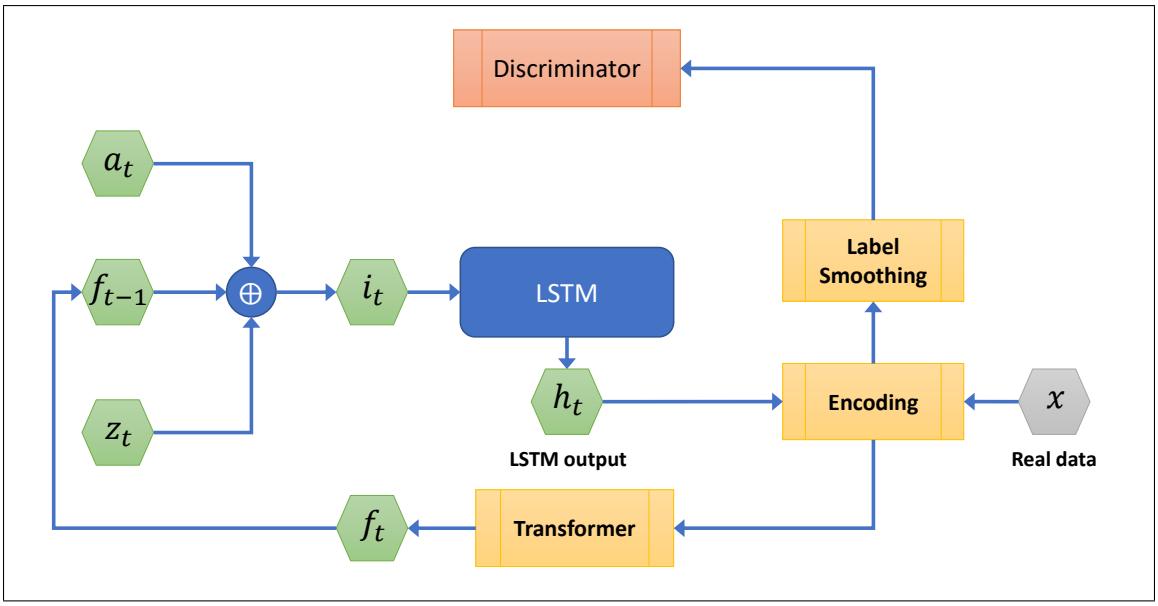

Figure 2.10: DATGAN process flow digram.

between variables should be injected as expert knowledge and cannot be detected by the model. However, tabular data cannot be considered sequential since the order of columns in a data table is generally random. Therefore, a DAG is used to create a specific sequence of columns.

Table 2.3: Comparative overview of GAN-based vs. traditional methods in tabular data generation.

Aspect	**GAN-Based Methods**	**Traditional Methods**
Data Complexity	Excelling in high-dimensional, complex data.	Suited for simpler, lower-dimensional data.
Realism	Generates highly realistic and detailed data.	Less capable in producing realistic data.
Computational Load	Higher, but necessary for complex model training.	Lower, but may compromise data complexity.
Ease of Use	Complex, but offers superior results for skilled users.	Simpler, but limited in advanced capabilities.
Versatility	Highly versatile in various domains and data types.	Limited versatility and application scope.
Control Over Data	Advanced techniques allow increased control.	More direct control, but at the expense of data quality.
Adaptability	Adapts well to new and evolving data patterns.	Less adaptive to changing data environments.
Innovation Potential	Continually evolving with cutting-edge research.	Lacks the rapid innovation seen in GAN-based methods.
Data Augmentation	Superior in generating novel data variations.	Basic augmentation capabilities.
Privacy Preservation	Can be tailored for privacy-preserving data generation.	Often lacks sophisticated privacy-preserving mechanisms.

2.3.4 Comparative Analysis of GAN-Based Methods

Table 2.3 presents a comparative analysis between GAN-based methods and traditional methods in synthetic data generation. It highlights GAN's superiority in handling complex, high-dimensional data and producing highly realistic outputs [35]. The table also emphasizes the versatility and adaptability of GAN-based methods

in various domains, showcasing their innovation potential and advanced capabilities in data augmentation and privacy preservation [36]. Conversely, traditional methods are noted for their simplicity and direct control over data, but they fall short in terms of complexity, realism, and adaptability.

2.3.5 Tabular GAN Evolution

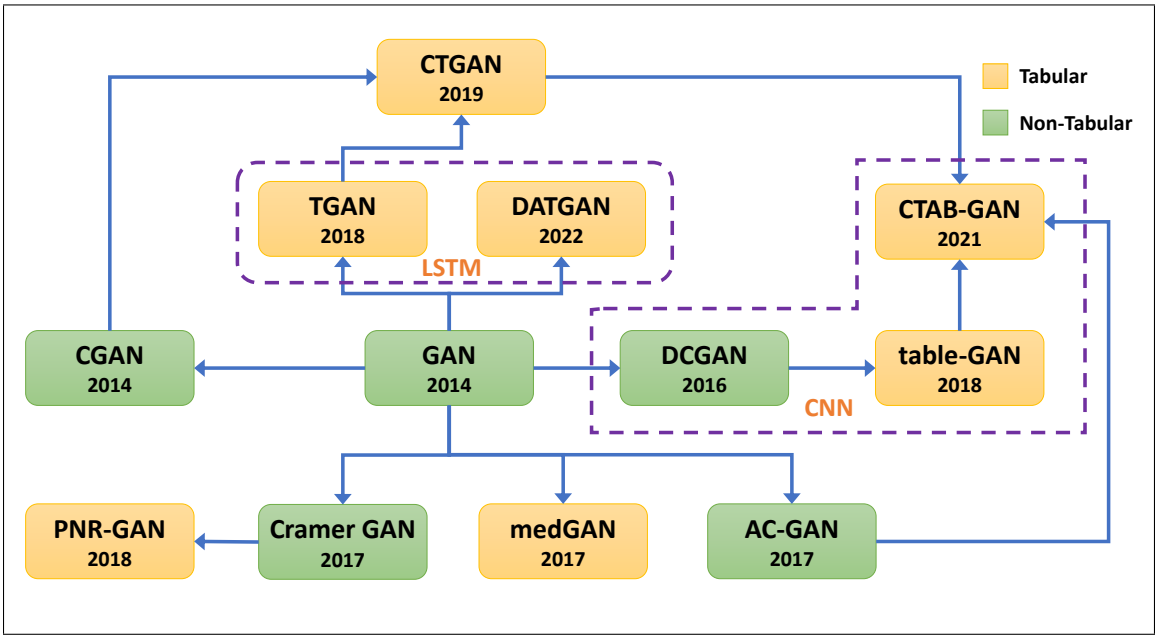

Figure 2.11: Tabular GAN-based generators evolution based on their relationship. Yellow boxes are tabular generators, and green boxes introduced for non-tabular data.

GAN has made significant progress in recent years, which has led to the development of novel variants that improve on previously introduced versions that had promising results prior to their introduction. Table 2.4 provides a summary of the variants of GAN that have been discussed in this paper. Also, figure 2.11 shows tabular GAN evolution, along with the year that they were introduced and their ancestors. As shown in this figure, table-GAN and CTAB-GAN utilize convolutional layers as part of their generator, however, CTAB-GAN makes use of a conditional version of generator built on CGAN and AC-GAN. CTGAN also utilizes the conditional version of GAN. With conditioned generators, realistic data can be generated based on the

constraints on the data table. On the other hand, TGAN and DATGAN use LSTM for memorizing the data relationships and correlations. Indeed, both conditional generators and LSTMs attempt to generate data based on a prior knowledge about the relationship between columns in a data table.

Table 2.4: Different tabular GAN architecture and capability.

Variant	Capability	Generator	Discriminator	Extra Loss Functions	Additional Networks
medGAN	Generate high-dimensional discrete columns Avoid mode collapse	FNN	FCN	MSE Cross-entropy	Autoencoder
PNR-GAN	Generate discrete columns Handling null values	cross-layer FCN	cross-layer FCN	Cramer loss	
table-GAN	Increase semantic Integrity	CNN	CNN	Information loss Classification loss	Classifier (MLP)
TGAN	Learn multimodal distribution. Generate mixed type variables.	LSTM	FCN	Cross-entropy	
CTGAN	Learn non-Gaussian and multimodal distribution. Address imbalance discrete column issue.	FCN	FCN	Wasserstein loss with gradient penalty	
CTAB-GAN	Generate discrete and mixed-type column Address imbalance discrete column issue. Learn long-tail distribution	CNN	CNN	Cross-entropy Information loss Classification loss	Classifier (MLP)
DATGAN	Increase semantic Integrity Increase representativity of imbalance class	LSTM	FCN	Wasserstein loss with gradient penalty	DAG

FNN: Feed Forward Neural Network

FCN: Fully Connected Neural Network

CNN: Convolutional Neural Network

MLP: Multi-Layer Perceptron

LSTM: Long Short-Term Memory

2.3.6 GAN vs. Diffusion Model for Tabular Data Generation

In the domain of synthetic data generation, particularly for tabular data, Generative Adversarial Networks (GANs) have demonstrated substantial efficacy. They adeptly handle complex relationships and heterogeneous data types, including nu-

merical, categorical, and ordinal variables. Studies reveal that GANs, through their adversarial training mechanism, can model intricate distributions and dependencies typical of tabular data. This capability significantly outperforms diffusion models, which often struggle with categorical and sparse data. This struggle stems from diffusion models' inherent design, which is better suited for continuous data modalities [37].

Diffusion models employ a process where each categorical feature is handled by a separate forward diffusion process. In this setup, noise components for all features are sampled independently. Moreover, diffusion tabular generators use multinomial diffusion to model categorical and binary features, and Gaussian diffusion for numerical ones [38]. This approach tends to ignore the correlations between categorical and numerical variables, potentially leading to synthesized data that does not accurately reflect the deterministic relationships found in the original data. As a result, diffusion models may fail to learn or even approximate these crucial relationships.

In essence, diffusion models operate by systematically degrading the training data through the successive addition of Gaussian noise, and then attempt to reconstruct the data by reversing this noising process. Specifically, a diffusion model is a latent variable model that employs a fixed Markov chain to map to the latent space. This describes the process of generating data points by simulating random walks from an initial state through a series of diffusion steps. In scenarios involving multimodal distributions, the random walk process may have to traverse through different modes of the distribution, a task that can pose significant challenges in accurately capturing the data's complex structure.

Conversely, GANs do not require the extensive dataset density nor complex preprocessing to handle issues like tabular data sparsity and high dimensionality, constraints that are less critical for GANs. Also, GANs are adaptable to various data densities and types without significant additional conditioning. Their application in tabu-

lar data synthesis has been successfully demonstrated across numerous benchmarks where they consistently generate higher fidelity data compared to diffusion models, which may still be experimenting with basic adaptations for tabular specifics [39]. Therefore, for synthetic data generation of tabular formats, GANs present a more robust and efficient solution than current diffusion model frameworks, confirming their superiority in handling the complex and varied demands of tabular data synthesis.

CHAPTER 3: METHODOLOGY

The CA-CTGAN architecture generates synthetic data that is representative of real-world data, with multiple types of data and additional contextual information. A detailed explanation of the proposed method and its structure is provided in this section.

3.1 Framework Overview

In this section, we introduce the Context-Aware Tabular Conditional GAN (CA-CTGAN) and underscore the methodology's novelty and its potential impact on the field of synthetic data generation. Building upon the CTGAN foundation, CA-TCGAN leverages a multifaceted approach to enhance synthetic data quality and controllability. We can break down the elements and operational path of the CA-CTGAN framework into the following items.

- **Contractive Autoencoder:** Initially, the raw data undergoes a transformation process to normalize and prepare it for input into the framework. The transformed data is then fed into a Contractive Autoencoder (CAE) to provide a semantically rich latent space and accelerates GAN training. Upon training the CAE, the model is saved, and using transfer learning, the latent space of this pretrained model is utilized to generate noise for the GAN generator. This approach diverges from traditional methods that employ a standard multivariate normal distribution (MVN) for noise, offering a more meaningful noise vector that reflects the semantic relationships between columns in the data.

- **Generator and Discriminator:** Following the noise generation, a fully connected generator commences the production of realistic synthetic data. This

data, alongside real data, is forwarded to both a discriminator, here referred to as a critic, and a classifier. The distinction between a critic and a traditional discriminator in GANs lies in the nature of their output and operational principle. While a discriminator classifies inputs into real or fake, a critic provides a continuous score that measures the authenticity of the input data, offering a more nuanced understanding and facilitating the generation of higher-quality synthetic data.

- **Classifier:** The classifier is trained on the generated data to predict class labels or other context-specific elements. This predictive capability enables the classifier to guide the generator towards producing data that is not only realistic but also contextually appropriate. The real data serves as a benchmark for evaluating the synthetic data's quality, with the classifier's loss being used to jointly train both the classifier and the generator. This feedback loop ensures that the generator's output continuously improves in quality and relevance.

The subsequent subsections, we will dissect each component of the CA-CTGAN framework in detail. This includes a deep dive into the workings of the Contractive Autoencoder, the generator-critic architecture, and the classifier's role in refining synthetic data generation. The overall interaction between CA-CTGAN framework components is shown in Figure 3.1.

3.2 Data Transformation

The Data Transformation process is a crucial initial step in preparing data for the CA-CTGAN framework, ensuring that the input is optimally formatted for processing by the network and it involves encoding continuous and categorical variables. The employed transformation techniques address the challenges inherent in dealing with real-world data, such as null values, long-tail distributions, multimodal distributions, and imbalanced categorical features.

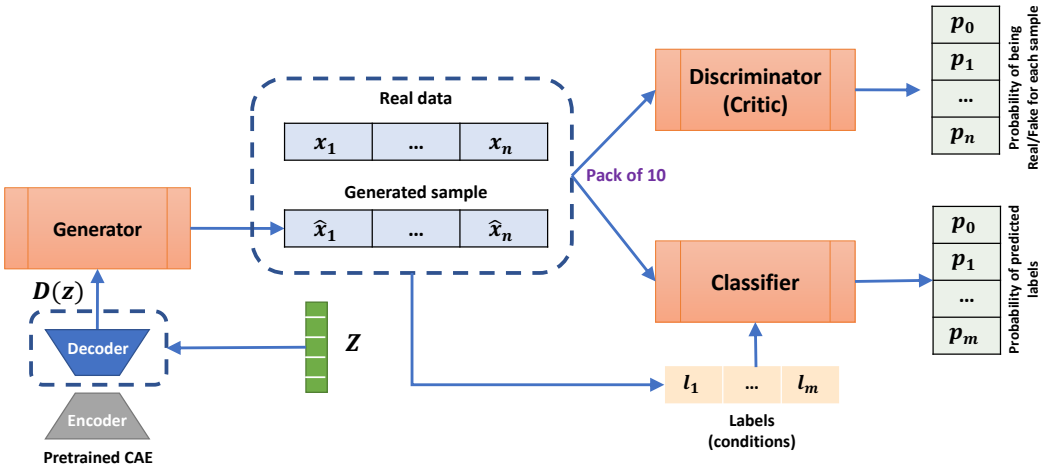

Figure 3.1: Components of CA-CTGAN.

3.2.1 Continuous Columns

For Continuous Columns, we employ Mode-specific Normalization [30] technique. Traditional normalization methods often assume a Gaussian distribution, which does not hold for many real-world datasets. Mode-specific Normalization, however, is specifically designed to manage non-Gaussian and multimodal distributions effectively.

This technique processes each column independently. Each value is represented by a one-hot vector identifying the mode and a scalar expressing the value within that mode. To achieve this end, we use Variational Gaussian Mixture (VGM) [40] which is a mixture of Gaussian distributions, where the model parameters are estimated based on variational inference. Variational inference is an optimization-based approach that approximates the true posterior distribution of the latent variables with a simpler distribution. Here, we describe encoding a multimodal distribution using a VGM step by step.

Mode Specification: Given data $X = x_1, x_2, ..., x_N$, where x_i represents the i^{th} data point, and assuming that the data is generated from K Gaussian distributions,

a Gaussian Mixture Model can be specified as:

$$\sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k) \tag{3.1}$$

where π_k is the mixing coefficient (prior probability) for the k^{th} Gaussian component, $N(.)$ is the Gaussian distribution with mean μ_k and covariance matrix Σ_k, and K is the number of components (modes).

The latent variable z_i denotes the assignment of the i^{th} data point x_i to a specific Gaussian component. It is a K-dimensional binary random variable which means one element related to the k is equal to 1 and the rest are 0. The prior distribution over z_i is given by a categorical distribution:

$$p(z_i) = Categorical(z_i|\pi) \tag{3.2}$$

where $\pi = \pi_1, ..., \pi_K$ is a vector of mixing coefficients.

Variational Inference: The goal of variational inference is to approximate the true posterior distribution $p(z|X)$ with a simpler distribution $q(z)$. In the case of VGM, we choose a factorized distribution for $q(z)$ as:

$$q(z) = \prod_i q_i(z_i) \tag{3.3}$$

where $q_i(z_i) = Categorical(z_i|\phi_i)$ and $\phi_i = \phi_i 1, ..., \phi_i K$ is a vector of variational parameters corresponding to data point i.

Evidence Lower BOund (ELBO): To estimate the parameters of the VGM (μ_k, Σ_k, π_k) and the variational parameters (ϕ_i), we maximize the Evidence Lower BOund (ELBO), which is a lower bound on the log-likelihood of the data. The ELBO can be expressed as:

$$L(X, \theta, \phi) = \sum_i E_q[\log p(x_i, z_i|\theta)] - E_q[\log q_i(z_i)] \tag{3.4}$$

where $\theta = \mu_k, \Sigma_k, \pi_k$ denotes the set of all model parameters.

Coordinate Ascent Variational Inference (CAVI): To maximize the ELBO, we can use the Coordinate Ascent Variational Inference (CAVI) algorithm, which iteratively updates the variational parameters ϕ_i and the model parameters θ. Algorithm 1 depicts the CAVI updates. By following this procedure, we can encode a multimodal

Algorithm 1 Coordinate Ascent Variational Inference for VGM

while not converged **do**
  for each data point i **do**
    Update $\phi_{ik} \propto \exp(\mathbb{E}_q[\log p(x_i, z_i = k|\theta)])$
    Normalize ϕ_i such that $\sum_k \phi_{ik} = 1$
  end for
  for each component k **do**
    $N_k = \sum_i \phi_{ik}$
    $\mu_k = \frac{1}{N_k} \sum_i \phi_{ik} x_i$
    $\Sigma_k = \frac{1}{N_k} \sum_i \phi_{ik} (x_i - \mu_k)(x_i - \mu_k)^T$
    $\pi_k = \frac{N_k}{N}$
  end for
end while

distribution using a Variational Gaussian Mixture model. The estimated parameters μ_k, Σ_k, and π_k can then be used to generate new samples for a specific mode.
In contrast to GMM, Variational Gaussian Mixture is more computationally demanding, but it is less likely to get stuck local optima and provides a measure of uncertainty in parameter estimates.

3.2.2 Categorical Data

A one-hot encoding is used to encode categorical data before feeding the Generator, and an integer encoding is used to encode ordinal data that maintains their inherent order. As opposed to one-hot encoding, which treats all categories independently, this approach maintains the ordinal relationship between them.
For feeding auxiliary classifier network, we use Entity Embedding [41] which is a technique that converts categorical variables into continuous representations by em-

bedding them in a continuous vector space. This approach is inspired by word embeddings [42] used in natural language processing, where words are mapped to vectors of continuous values, capturing semantic relationships between them. Entity embedding can be applied to categorical variables in a similar fashion, enabling the preservation of the relationships between categories while transforming them into a more suitable format for deep learning models.

To implement entity embedding, we can follow these steps:

Determine the size of the embedding vector (k): The size of the embedding vector is a hyperparameter that needs to be chosen based on the problem domain and the size of the categorical variable. A common heuristic is to choose k as the minimum of 50 and $(number\ of\ unique\ categories + 1)/2$.

Create an embedding layer: For each categorical variable, create an embedding layer in a neural network. The input to this layer is the integer-encoded version of the categorical variable, where each category is assigned a unique integer. The output of the embedding layer is a k-dimensional continuous vector representing the embedded category.

Train the neural network: While training the neural network, the categorical variables are fed into their respective embedding layers. The embedding layers learn to map the categorical variables to continuous representations by minimizing cross-entropy loss function.

Embedding matrix E of size (n, k) can be formulated as $E = [e_1, e_2, ..., e_n]$ where n represents the number of unique categories and e_i is the k-dimensional continuous vector representation of category i. During the training process, the categorical variable x_i is transformed into a continuous representation using the embedding matrix E.

By employing entity embedding, the categorical variables are transformed into continuous representations that can be effectively utilized in deep learning models. The

learned embeddings can capture the relationships between categories, leading to improved model performance and more meaningful generated data. However, Extracting the original categorical values from the embedded continuous representations is not straightforward, as the mapping is not one-to-one and the embeddings are learned in a continuous vector space. The goal of embedding is to capture the semantic relationships between categories, and during the learning process, these continuous representations are optimized to serve the main task, such as regression or classification. Therefore, we use embedding to train the Classifier network which will be presented in the next section.

3.2.3 Mixed-mode Data

In the section 2.3.1, we discussed the challenges of encoding mixed-type data, which consists of both categorical and continuous values or continuous values with missing entries. To address this issue, we employ the Mixed-Type Encoder introduced in the CTABGAN [1]. Mixture-type values are encoded as concatenated value-mode pairs, and continuous values are encoded using VGM (section 3.2.1). A mode indicator vector is used to encode the categorical component without normalization. As a result, mixed-type data is adequately represented, allowing for efficient processing and integration in the context of GAN-based data generation. Figure 3.2 shows the distribution over an arbitrary mixed-type column, with two modes for continuous (m_2, m_3) and two categorical parts (m_1, m_4) and illustrates how this algorithm transforms one row of mixed-mode data. Therefore the representation of a row become the concatenation of continuous and discrete columns (Eq. 2.10).

3.3 Design and Training process

As we mentioned before in section 3.1, the proposed CA-CTGAN architecture comprises four primary components: Contractive Autoencoder (CAE), Generative, Discriminator (Critic) and Classifier. We use CAE in Transfer Learning process and

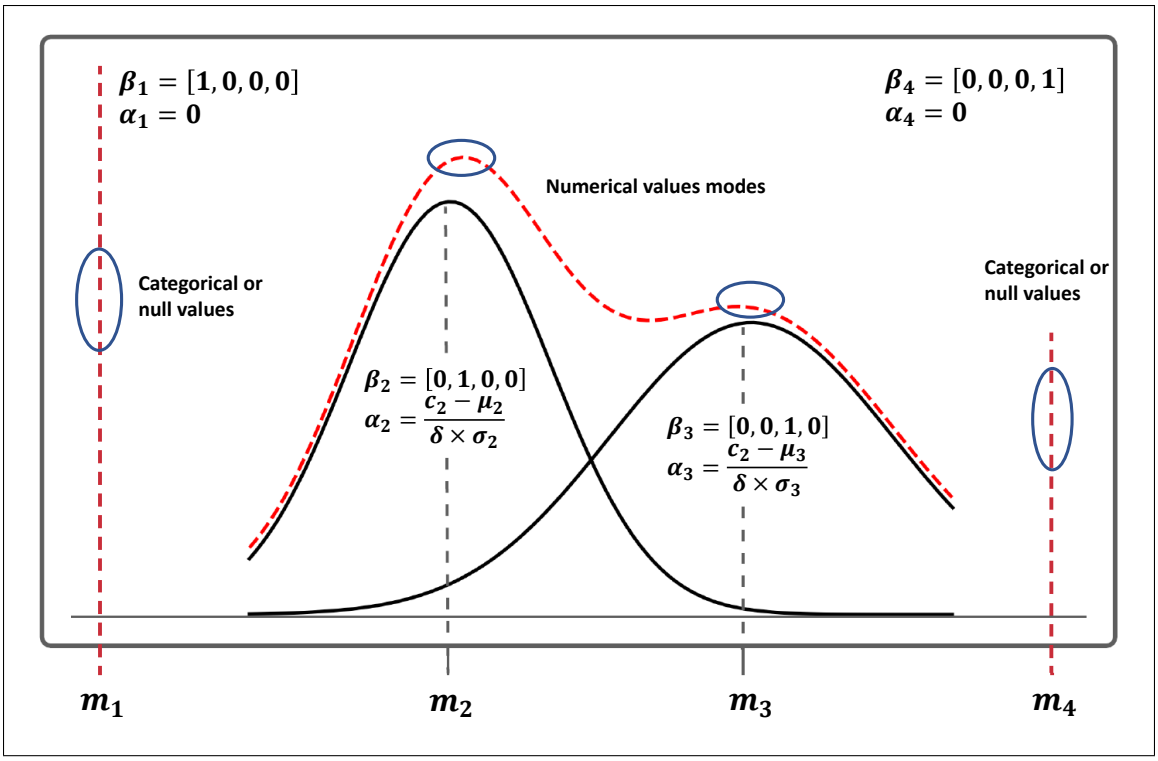

Figure 3.2: Distribution over a mixed-type column. m_1 and m_4 represent the categorical part or null values of this column, whereas m_2 and m_3 represent modes for numeric parts. The numeric parts are defined by Variational Gaussian Mixture (VGM) model. [1]

it acts as noise generator for generator and the output of generator along with the real data is used as input for discriminator and classifier. This section provides an in-depth exploration of the design and architecture of each model.

3.3.1 Contractive Autoencoder

After data preprocessing, CAE is trained to learn an efficient and meaningful representation of the input data. The encoder comprises two linear layers, transitioning the input transformed data from its original dimensionality to a hidden representation, and subsequently to a lower-dimensional latent space. Mirroring the encoding process, the decoder reconstructs the data from the latent space back to its original dimensionality which is matched with generator input. This is achieved through two

linear transformations, which progressively upsample the latent representation to the hidden state and then back to the input space. The sigmoid activation function is employed at each stage of both encoder and decoder to ensure the output values are appropriately scaled.

During each iteration of the training process, a batch of input data is fed into the CAE, and the output of the network is compared to the input data to compute the reconstruction loss. Also, the contractive loss, which encourages the model to learn a stable and invariant representation of the input data, is calculated by taking the Frobenius norm of the Jacobian of the encoder with respect to the input data (Eq. 2.5). This is achieved through automatic differentiation techniques. The reconstruction error, typically measured by the Mean Squared Error (MSE) between the input and its reconstruction. This process ensures that the most relevant features of the data are captured in a lower-dimensional space. However, when augmenting this objective with the Kullback-Leibler (KL) divergence loss [11], the model is encouraged not only to accurately reconstruct the input data but also to regularize the latent space representations. The KL divergence is a measure of how one probability distribution diverges from a second, expected probability distribution. This regularization effect helps in mitigating overfitting by preventing the model from learning to simply memorize the training data. The incorporation of KL divergence is especially pertinent for variational autoencoders (VAEs) [11], where it is a critical component that differentiates them from standard autoencoders. However, in the context of enhancing a traditional autoencoder for tabular data, the KL divergence loss can similarly impose a probabilistic structure on the latent space, leading to more meaningful and generalizable representations. Hence, the reconstruction loss can be represented as follows:

$$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{3.5}$$

$$\mathcal{L}_{KLD} = D_{KL}(P||Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)} \tag{3.6}$$

$$L = \mathcal{L}_{MSE} + \beta \times \mathcal{L}_{KLD} \tag{3.7}$$

Where β is a hyperparameter that balances the contribution of the KL divergence loss relative to the MSE loss. The total loss is the sum of the reconstruction loss and the contractive loss, weighted by a hyperparameter lambda that controls the trade-off between the two. Using backpropagation (Eq. 2.6), the gradients of the total loss with respect to the network parameters are computed, and the parameters are updated using the Adam optimizer. After the CAE model is satisfactorily trained, it can be employed to directly generate the input noise for the Generator in the CA-CTGAN. The input and output layer sizes of the CAE are designed to match the transformed data size, ensuring seamless integration within the generator. This alignment allows the CAE to directly process the transformed data, facilitating an efficient and effective noise generation process. This utilization of the pre-trained CAE allows the generator to benefit from the captured interdependencies between the columns in the data, providing a more meaningful starting point. As a result, this strategy contributes to a reduction in the convergence time for the CA-CTGAN and leads to an improvement in the quality of the generated data. Algorithm 2 shows the training process of CAE. In order to generate an input noise vector for the Generator using the pre-trained CAE, we follow a systematic approach that leverages the characteristics of the CAE architecture. The following steps outline the process for generating a sample from the latent space using the CAE:

1. **Transfer the pre-trained CAE:** Firstly, we transfer the weights of the pre-trained CAE, which has already captured the intricate relationships between the columns in the data.

2. **Sample from the latent space:** To generate a sample from the latent space,

Algorithm 2 Training Contractive Autoencoder

Input: Training data, $X \leftarrow concat(X, labels)$ with batch size B
Output: Trained model, $CAE \leftarrow$ Weights of trained model

Initialize the Contractive Autoencoder model with randomly generated weights
for n training iterations **do**
  for each batch of data X_{batch} in training data X, with batch size B **do**
    for each data sample x in batch X_{batch} **do**
      Pass input data x through the Contractive Autoencoder to obtain reconstructed data x' and encoded representation z
      Compute the reconstruction loss L_{recon} and the contractive loss L_{cont}
    end for
    Compute the average reconstruction loss $\bar{L}_{recon}$ and average contractive loss $\bar{L}_{cont}$ for the batch
    Calculate the gradients of the average total loss $\bar{L} = \bar{L}_{recon} + \bar{L}_{cont}$ with respect to the model parameters using backpropagation
    Update the model parameters using Adam optimizer
  end for
end for
return CAE

we randomly sample a vector from a Multivariate Normal Distribution (MVN). This sampled vector should have the same dimensionality as the latent space.

3. **Utilize the decoder:** The sampled vector from the latent space is then fed into the decoder part of the CAE. The decoder reconstructs a new data point in the original data space based on the latent representation.

Incorporating a Contractive Autoencoder (CAE) and adjusting the weight of the contractive loss, alongside MVN for sampling, have significantly improved the diversity and quality of data reconstruction from the latent space. This improvement can be attributed to the enhanced regularization and the sophisticated noise injection mechanism, which together foster a more robust and generalizable model.

The implementation of a Contractive Autoencoder (CAE) with a relatively high hyperparameter λ (0.3) for the contractive loss represents a strategic enhancement to the autoencoding framework, primarily aimed at augmenting the model's ability to

generate diverse and high-fidelity reconstructions from latent representations. The contractive loss effectively imposes a constraint on the sensitivity of the learned representations to small variations in the input data. By intensifying the weight of this contractive term, the model is encouraged to learn a latent space that is more invariant to minor perturbations in the input data. This regularization technique not only aids in mitigating overfitting by discouraging the memorization of training data but also promotes a smoother and more continuous latent space. Consequently, when noise is injected into this well-regularized latent space, the decoder can interpolate more effectively, leading to the generation of a richer diversity of plausible data points that maintain fidelity to the underlying data distribution. Furthermore, the adoption of a Multivariate Normal Distribution (MVN) for noise generation is a critical factor that synergizes with the CAE's enhanced latent space to improve reconstruction diversity. The choice of MVN allows for the injection of noise that is statistically coherent with the assumptions underpinning many natural data distributions. This compatibility ensures that the explorations conducted in the latent space via noise injection are meaningful and aligned with the geometry of the data manifold encoded by the autoencoder.

3.3.2 Generator

A significant challenge in the generator architecture is the inherent imbalance present within categorical columns of tabular data. With deterministic transformations, even mapping a matched distribution of real data in traning process, fail to address this imbalance effectively. Such approaches can lead to underrepresentation of minority categories during training, resulting in a generator that poorly approximates the real data distribution. This problem is analogous to the class imbalance issue encountered in discriminative modeling but is further complicated by the multi-column nature of tabular data and the necessity to maintain the integrity of the real data distribution. To overcome these challenges, the CA-CTGAN framework adopted

conditional generator [30] that aims to evenly sample across all categories of discrete attributes during training, thereby ensuring a balanced representation of the data. This generator is designed to learn the conditional distribution of data given specific attribute values, allowing for the reconstruction of the original, unaltered data distribution during evaluation (eq. 2.17).

The integration of a conditional generator within the GAN architecture necessitates addressing several key issues:

1. Condition Representation: It's crucial to develop a method for representing the condition (i.e., the specific value from a discrete attribute) and preparing the generator's input to include this condition.

2. Condition Preservation: Ensuring that the generated data preserves the specified condition is essential. This requires the generator to accurately embed the condition within the generated samples.

3. Learning the Real Data Conditional Distribution: The conditional generator must effectively learn the real data's conditional distribution. This capability is critical for the generator to not only balance the representation of categories during training but also to accurately reconstruct the original data distribution.

The generator is a feed-forward neural network and employs Leaky ReLU activation functions in their hidden layers. This activation function mitigates the issue of dying ReLU in the generator network, resulting in faster convergence and improved performance [43].

3.3.3 Discriminator

The discriminator, referred to as the critic. The primary function of the critic is to estimate the divergence between the conditional distribution of the generated data $\mathbb{P}_g(row|condition)$ and the conditional distribution of the real data $\mathbb{P}(row|condition)$.

This evaluation is critical for guiding the generator towards producing data that closely approximates the real data distribution under various conditional constraints. The effectiveness of the critic's assessment hinges on the appropriate sampling of the condition vector and the real training data. The condition vector, which represents specific attribute values for which data is generated, must be sampled in a manner that ensures a comprehensive exploration of the attribute space, including both common and rare categories. This balanced exploration is vital for the critic to accurately estimate the divergence across the entire distribution of attribute values, rather than focusing disproportionately on more frequent categories.

To achieve this balanced exploration, the CA-CTGAN framework employs a training-by-sampling strategy [30] where the condition vector and training data are sampled according to the log-frequency of each category. This approach mitigates the bias towards overrepresented categories by elevating the importance of rarer categories, ensuring that the model develops an even understanding of all possible values within discrete columns. By sampling condition vectors and training data based on category log-frequency, the framework promotes an equitable representation of the data's diversity.

The discriminator network takes in packs (m samples) of both real and fake data samples based on PacGan methodology [2]. Mode collapse, or the lack of divergence between the generated samples from the trained generator, is one of the major challenges associated with training Generative Adversarial Networks (GANs). Different approaches have been proposed, like minibatch discriminators, two-sample tests, moment matching, and inverse mappings of generators. The PacGAN framework is introduced to tackle this problem [2]. The key idea behind PacGAN is to modify the discriminator to process a pack of data samples instead of a single sample. Thus, instead of using a discriminator $D(x)$ to map one row to a label, it uses an augmented discriminator $D(x_1, x_2, ..., x_m)$ to map m samples, both from real data and from the

generator. This helps to capture the inter-sample relationships and discourage the generator from focusing on generating a single mode in the data distribution.

To do so, we concatenate m samples along a new dimension and modify the subsequent layers to process the concatenated samples, and ensure that the input labels are also adjusted accordingly to match the pack of samples. Figure 3.3 represents how PacGAN augments the input layer by $m = 2$ packs. The grid-patterned nodes show input nodes for the second sample.

The discriminator comprises packed samples that are represented as $(x_1, x_2, ..., x_m)$,

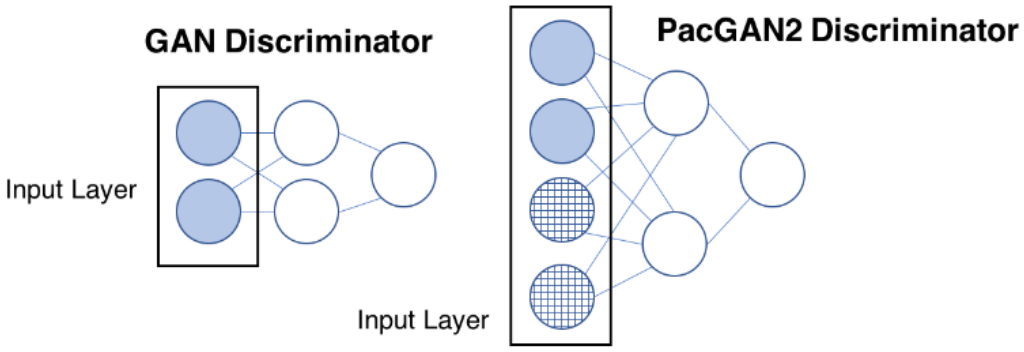

Figure 3.3: In the PacGAN model, the input layer is expanded by a factor of the packing degree (Here $m = 2$). The connections to the initial hidden layer are modified to ensure that the first two layers maintain full connectivity, consistent with the original architecture [2].

either for real data or generated data. The set of m independent samples originating from each class are regarded as a unified, high-dimensional feature $(x_1, x_2, ..., x_m)$. Therefore, the discriminator acquires the ability to classify a set of m samples that have been packed together. The process of packing the samples is believed to assist the discriminator in detecting mode collapse, as the absence of diversity is less likely in a group of samples as opposed to an individual sample.

3.3.4 Auxiliary Classifier

In parallel to the discriminator, the classifier network predicts the labels (context-specific elements) of each generated row, enabling the assessment of the semantic integrity of the generated records. Classifier's function goes beyond just creating data to evaluating if the created rows maintain the contextual subtleties of the original dataset.

A standout feature in classifier networks is the incorporation of Entity Embedding techniques [41] for encoding categorical columns. Entity Embeddings offer a dense, low-dimensional, and meaningful representation of categorical variables, transcending the limitations of traditional one-hot encoding by capturing and preserving the relational intricacies among categories. This technique not only reduces the dimensionality of the input space but also enhances the model's ability to learn complex patterns and relationships among categorical variables, which are often prevalent in tabular datasets. By integrating the embeddings with numerical inputs, the model creates a unified representation that is fed through a series of linear layers and normalization steps, enhancing the learning process through added layers of abstraction and complexity reduction.

The architecture of classifier embodies a sequence of linear layers, batch normalization, and dropout, orchestrated to refine the feature representation progressively, culminating in the prediction of labels. This label prediction is critical for assessing the semantic integrity of the generated rows, ensuring that the data can be generated for specific consition.

3.3.5 Loss Function

The complexity of GAN training stems from the simultaneous optimization of two models, the generator and the discriminator, that are in a constant competition with

each other. This leads to a delicate balance between the two models, where a minor imbalance can result in the failure of the optimization process [6]. In addition to mode collapse, one of the challenges in GAN training is the vanishing gradient problem, which occurs when the discriminator becomes too powerful in differentiating between real and fake data. In this scenario, the loss function reaches zero, resulting in no gradient for updating the generator during learning iterations. Conversely, if the discriminator performs poorly, the generator may not receive accurate feedback, leading to a suboptimal representation of the target distribution in the generated samples [44]. In order to tackle the aforementioned concerns and produce realistic samples that can effectively replicate the samples in desired labels, we propose the utilization of the following Loss functions.

Original Loss: The original loss functions for the generator and discriminator (Eq. 2.2) are adapted to accommodate the use of PacGAN in the discriminator. For the discriminator, as we disscussed in section 3.3.3 the objective is to correctly classify each packed sample as either real or generated. The loss function is formulated as a binary cross-entropy loss applied to the bundled samples. The discriminator loss, L_D, is defined as the sum of the losses for correctly classifying real data and generated data:

$$L^D_{Orig} = \mathbb{E}_x[\log(D(x_1, ..., x_m|y))] + \mathbb{E}_z[\log(1 - D(G(z_1, ..., z_m|y)))] \tag{3.8}$$

The generator's loss function L_G is defined using binary cross-entropy. The generator aims to deceive the discriminator into classifying generated data as real:

$$L^G_{Orig} = -\mathbb{E}_z[\log(D(G(z_1, ..., z_m|y)))] \tag{3.9}$$

where the generator tries to maximize the probability of the discriminator misclassifying the generated samples as real data.

Similarity Loss: The Wasserstein distance, utilized as a critical component in train-

ing the discriminator (referred to as a critic in the context of WGANs, to emphasize its evaluative rather than binary classificatory role), ensures that the discriminator accurately assesses how closely the generated data approximates the distribution of real data. Recognized for its effectiveness in measuring the discrepancy between two probability distributions, the Wasserstein distance significantly enhances the stability and quality of training in GANs. By incorporating the Wasserstein distance, along with a gradient penalty, into the critic's loss function, the critic is better equipped to guide the generator toward producing data that more closely mirrors the real data distribution. The generator, aiming to minimize the critic's evaluation of its outputs, is indirectly influenced by the Wasserstein distance to generate higher-quality data. As mentioned before, unlike traditional GANs where the discriminator operates on a binary output, the critic in WGANs (Eq. 3.10) provides a continuous value that more precisely estimates the Earth Mover's distance, or the minimum cost of transporting mass to transform the generated data distribution into the real data distribution. This can be conceptualized as the infimum (greatest lower bound) of transportation costs, making the critic's role pivotal in refining the generator's outputs through a more nuanced and effective training process.

$$\mathcal{W}(p_{data}, p_g) = \inf_{\gamma \sim \Pi(p_{data}, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\| x - y \|] \tag{3.10}$$

Where $\Pi(p_{data}, p_g)$ is all possible joint probability distributions between p_{data} and p_g, x is the starting point in data distribution and y is the destination in generated distribution. However, the calculation of the Wasserstein distance using Eq. 3.10 requires evaluating all possible joint distributions of samples from two distributions, which is intractable due to the large number of possible distributions. To address this issue, the [8] in WGANs proposed to use the Kantorovich-Rubinstein duality to compute an upper bound on the Wasserstein distance. The dual form of the

Wasserstein distance is given by:

$$L^D_{WGAN} = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_g}[f(x)] - \mathbb{E}_{x \sim p_{data}}[f(x)] \tag{3.11}$$

where $f(x)$ is Lipschitz continuous functions from the sample space to the real numbers, and the supremum is taken over all functions f. This approach offers a practical means of approximating the Wasserstein distance between two distributions. By optimizing the functions f , the model effectively estimates the Wasserstein distance between the real and generated data distributions. This estimation is then utilized to train the discriminator, or critic, ensuring it can accurately assess the discrepancy between real and generated distributions. The generator, in turn, is trained to minimize the critic's evaluations, indirectly guided by the Wasserstein distance to improve the quality of its output. The incorporation of the Wasserstein distance and gradient penalty in the critic's loss function plays a pivotal role in stabilizing the training process and enhancing the fidelity of the generated data, without directly applying the Wasserstein loss as a training criterion for the generator.

However, enforcing the 1-Lipschitz condition is not straightforward. Initially, weight clipping was used to enforce this condition, but it was found to lead to optimization issues and poor quality of generated samples. This led to the introduction of Gradient Penalty (GP) as a method to enforce the 1-Lipschitz condition more gently and effectively [33].

The Gradient Penalty adds an additional term to the loss function that penalizes the model if the gradient norm moves away from 1. This is done by sampling points along the straight line between pairs of real and generated data points and ensuring that the gradients of the critic's output with respect to these points have a norm of 1. The Gradient Penalty term is formulated as:

$$GP = \lambda(\|\nabla_{\hat{x}} f(\hat{x})\|_2 - 1)^2 \tag{3.12}$$

Where $\hat{x}$ represents the sampled points, λ is a penalty coefficient, and $\nabla_{\hat{x}} f(\hat{x})$ is the gradient of the critic's output with respect to $\hat{x}$. This term is added to the original Wasserstein Loss, leading to the final objective:

$$L^D_{WGAN} = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_g}[f(x)] - \mathbb{E}_{x \sim p_{data}}[g(y)] + GP \tag{3.13}$$

The generator's goal, on the other hand, is to produce data that minimizes the critic's ability to distinguish between real and generated samples. In the context of WGAN-GP, the generator is trained to minimize the following loss function:

$$L^G_{WGAN} = -\mathbb{E}_{x \sim p_g}[f(x)] \tag{3.14}$$

This means the generator aims to maximize the critic's score for its generated samples.

Classification Loss: During the Classifier training process, two distinct loss functions are employed to measure the discrepancy between the conditioned labels y and predicted labels $\hat{y}$. The first loss function L^C_{class} quantifies the divergence between the conditioned labels and the predicted labels obtained from the original data using the Classifier. This loss function evaluates the Classifier's ability to correctly classify the labels of real data samples.

$$L^C_{class} = \mathbb{E}_{x \sim p_{data}}[|y - C(x)|] \tag{3.15}$$

where y is the ground truth label and $C(.)$ returns the predicted labels for real data by classifier.

The second loss function L^G_{class} measures the discrepancy between the conditioned

labels and the predicted labels derived from the generated data using the Classifier. This loss function assesses the semantic integrity of the generated data samples by comparing the input labels used to generate the data with the predicted labels assigned by the Classifier. By minimizing both loss functions, the Classifier effectively learns to distinguish between the original and generated data while preserving the contextual relationships between input features and corresponding labels.

$$L_{class}^{G} = \mathbb{E}_{x \sim p_g}[|y - C(G(z))|] \tag{3.16}$$

where y is the ground truth label, $C(G(z))$ represents the predicted labels for generated data by classifier.

3.3.6 Training Process

During the training process of the proposed methodology, each epoch is carried out by processing the data in minibatches and components of the model are trained using specific loss functions to optimize their performance. The Discriminator is trained using $L_{orig}^{D} + L_{wgan}^{D}$, which is adapted to incorporate the PacGAN [2] and WGANGP [8]. This enables the Discriminator to efficiently distinguish between real and generated data samples while addressing potential issues of mode collapse. The Classifier network is trained using L_{class}^{C}, which measures the discrepancy between the conditioned labels and the predicted labels obtained from the original data. This loss function helps the Classifier to accurately predict context-element labels associated with the input data, ensuring the semantic integrity of the generated synthetic records. The Generator is trained using $L_{orig}^{G} + L_{wgan}^{G} + L_{class}^{G}$. The Class Loss measures the discrepancy between the conditioned labels and the predicted labels derived from the generated data.

By employing these specific loss functions during the training process, the proposed methodology ensures the effective generation of contextually relevant and semantically

meaningful synthetic data. Algorithm 3 shows the training process of CA-CTGAN and the figure 3.4 represents the training flow in CA-CTGAN architecture.

Algorithm 3 Training CA-CTGAN

Input: Training data, $X \leftarrow concat(features, labels)$ with batch size B
Output: Trained Generator, $G \leftarrow$ Generator of Trained Conditional GAN

Initialize Generator G, Discriminator D, and Classifier C
Preprocess the data, apply feature encoding
Initialize the input layer of generator with the transferred pre-trained Contractive Autoencoder
for n training epochs **do**
  for each batch of data X_{batch} in training data X **do**
    Sample X_{batch} of real data and their corresponding $Labels$
    Pack samples for Discriminator using PacGAN
    Update D using L^D_{orig} for packed real and fake data
    Update C using L^C_{class} for real data
    Generate minibatch of fake data and their corresponding labels using G
    Update G using $L^G_{orig} + L^G_{wass} + L^G_{class}$ for fake data
  end for
  Average the losses over the batch
  Calculate the gradients of the losses with respect to the model parameters using backpropagation
  Update the model parameters using Adam optimizer
end for
return Trained Model

Figure 3.1 represents the overall CA-CTGAN architecture, comprising the generator, discriminator, and classifier networks, effectively generates contextually relevant synthetic data that respects the semantic relationships between input features and corresponding spatial or temporal labels. By incorporating PACGAN, the framework addresses the mode collapse issue, making it a versatile solution for synthetic data generation.

3.4 Contribution and Novelty

In this section, we highlight the key contributions and innovative aspects of the proposed CA-CTGAN framework. These aspects showcase the significance of the research and the potential to advance synthetic data generation across various domains.

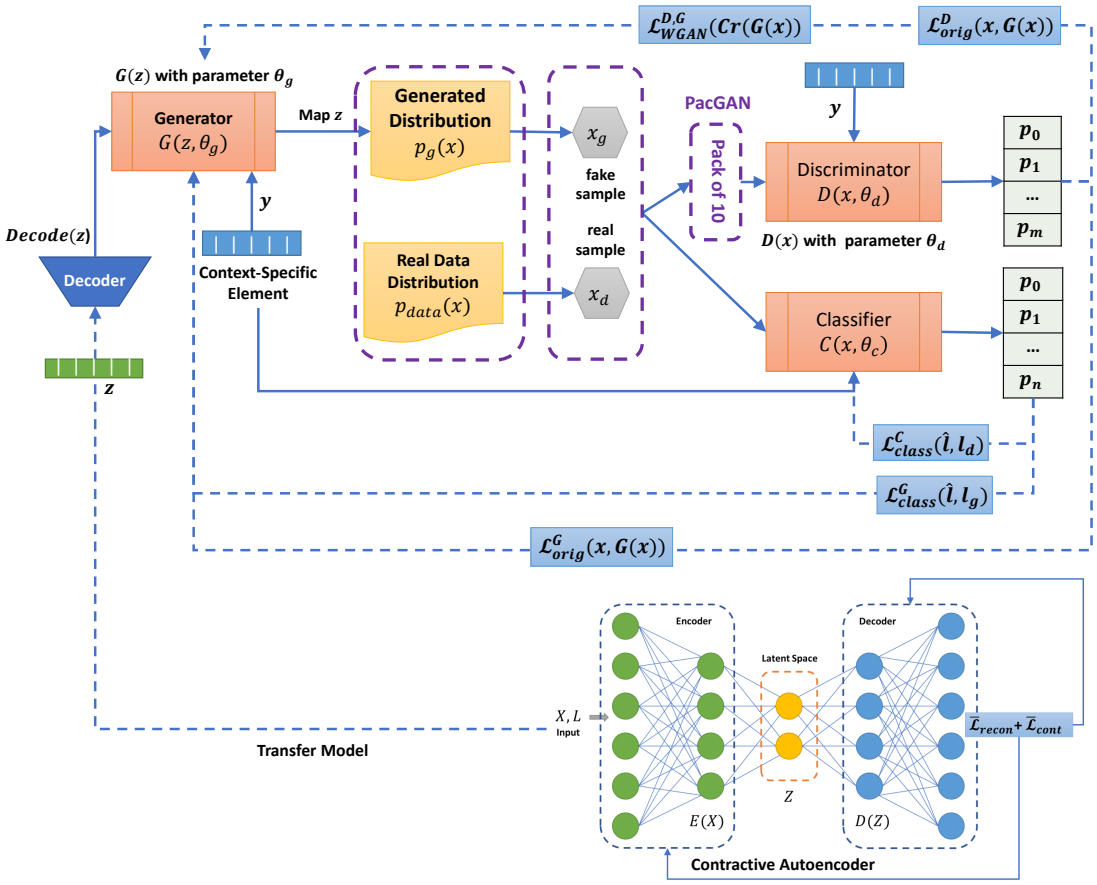

Figure 3.4: Detailed schematic representation of the CA-CTGAN training process and architecture.

The major contributions and novelties are as follows:

- Development of a novel Context-Aware Conditional Tabular GAN (CA-CTGAN) architecture that effectively synthesizes high-resolution tabular data while considering the contextual information for diverse experimental domains.
- Development of a new tabular data generation framework (CA-CTGAN) using conditional GAN that allows the framework to generate data specifically tailored with targeted synthetic data.
- Integration of Transfer Learning in the CA-CTGAN framework, enabling the generator to capture complex relationships between columns in the data while generating synthetic data.

- Development of an auxiliary multi-class Classifier network with entity embedding for controlling the generation process, enabling CA-CTGAN to produce data points at desired context-specific elements.

- Showcasing the potential applications of the CA-CTGAN approach across diverse fields, including including laboratory, field, natural, and clinical experiments, thereby contributing to the advancement of knowledge and technology in these domains.

This innovative application of the GAN-based tabular data generation not only maintains the semantic integrity of the generated data but also increases the applicability of the CA-CTGAN approach across a wide range of scientific fields and enhances the quality of the synthetic data to provide a strong foundation for further research and development in the area of synthetic data generation.

CHAPTER 4: EXPERIMENTAL STUDIES

This chapter provides a detailed overview of the datasets employed within this dissertation and outlines the experimental setup used for developing and evaluating the synthetic tabular data generation framework. The datasets are characterized in terms of their origin, size, features , and the presence of context-specific labels. Additionally, the chapter describes the hardware and software specifications, parameter settings, and evaluation metrics used throughout the experiments.

4.1 Datasets

We utilized nine real world datasets throughout the experiments and a detailed characterization of the each dataset is presented in the following section.

4.1.1 Adult Income Dataset

The Adult dataset [45] is a real-world dataset derived from the 1994 US Census. It contains approximately 48,842 instances with a mix of numerical and categorical features. The dataset is commonly used as a benchmark for classification tasks focused on predicting income level (above or below $50K annually). For effective utilization, the Adult dataset frequently requires preprocessing to handle missing values and encode categorical variables. Figures 4.1, 4.2 and 4.3 illustrate an analysis of the distribution of values in the dataset. For categorical features, violin plots highlight the relative frequency of different categories within each variable. Pair plot offers insights into the distribution of each numerical features as well as correlation between columns.

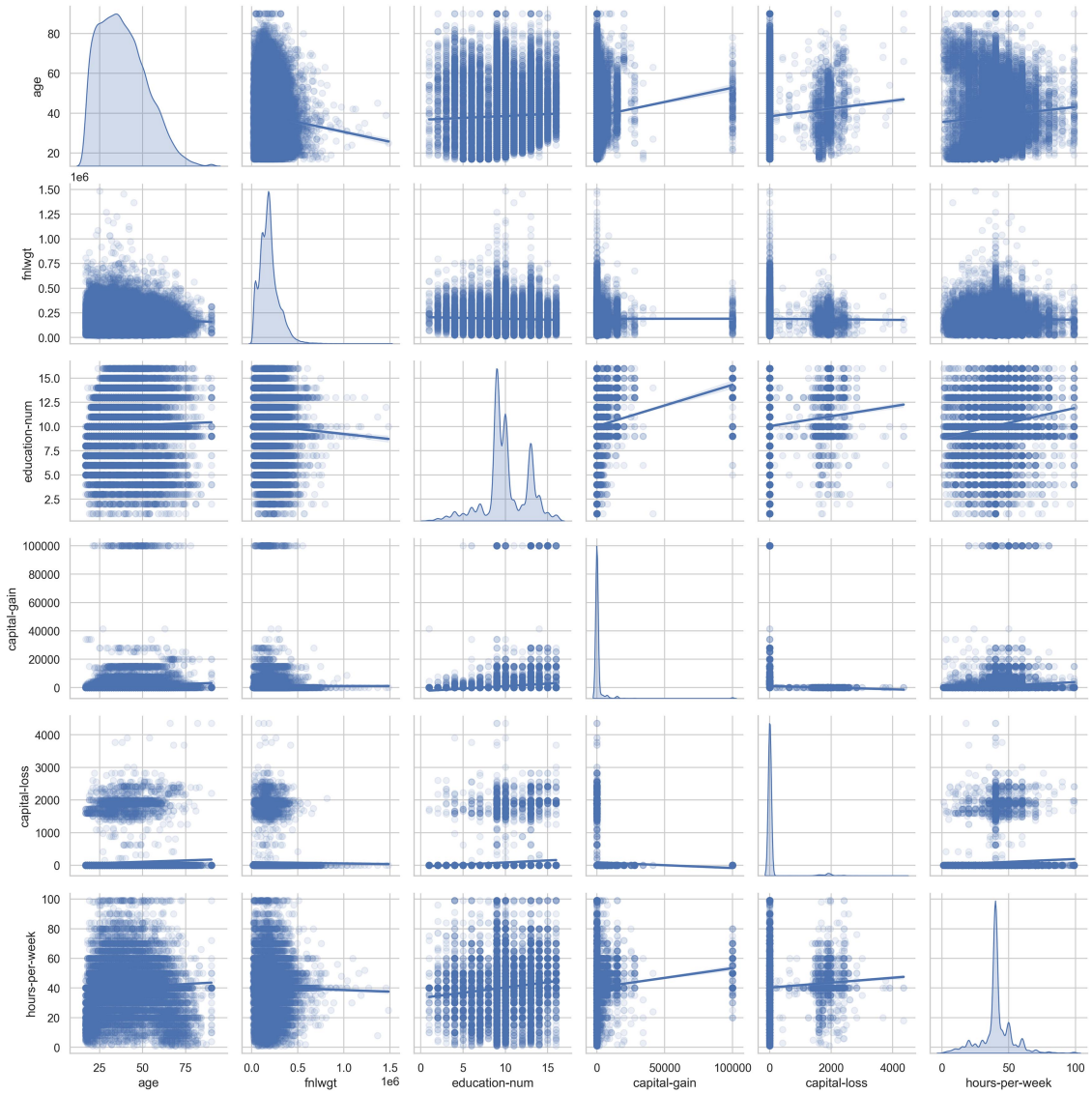

Figure 4.1: Distribution and correlation of numerical values in the Adult dataset

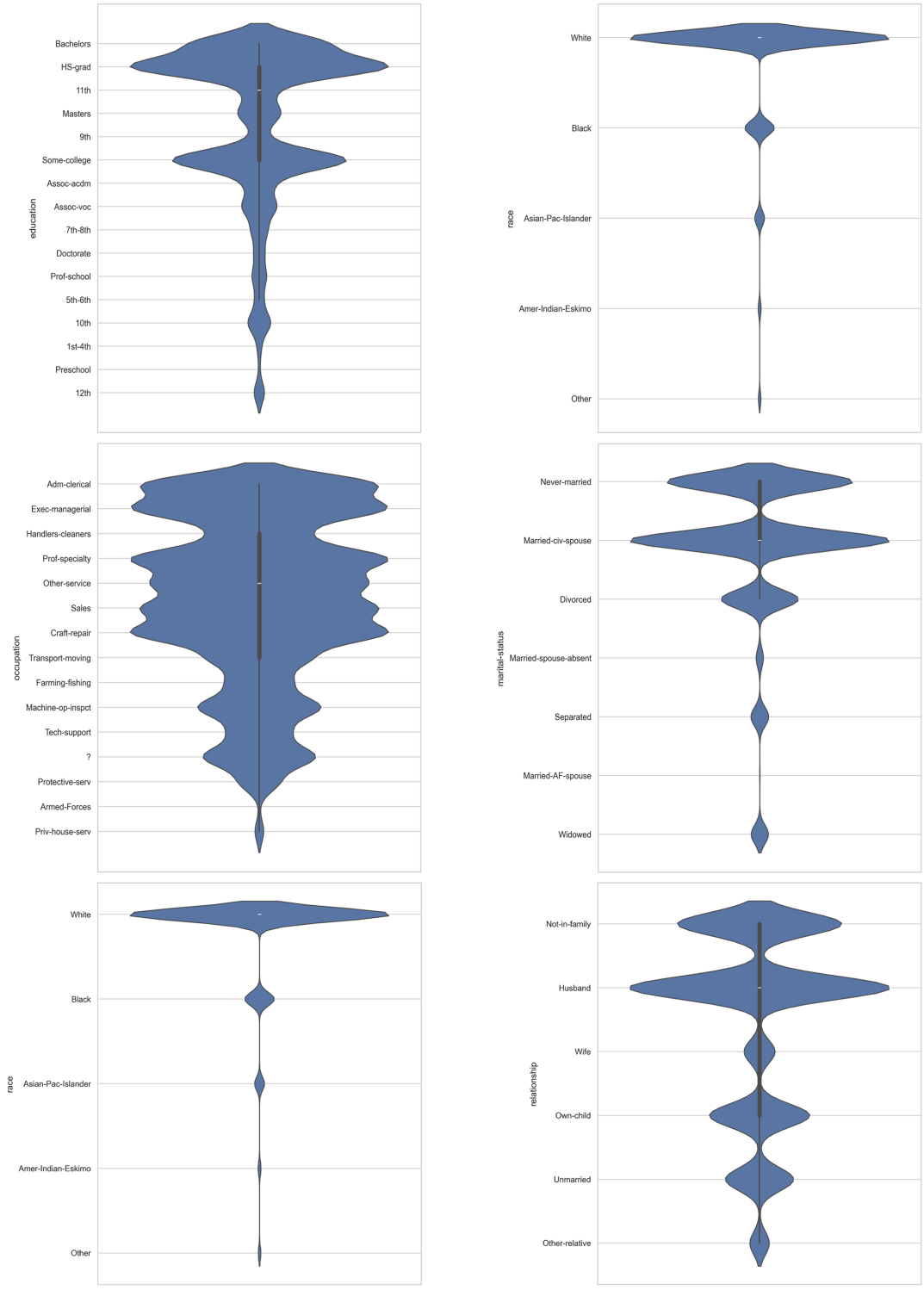

Figure 4.2: Frequency of different categories within each categorical column in the Adult dataset

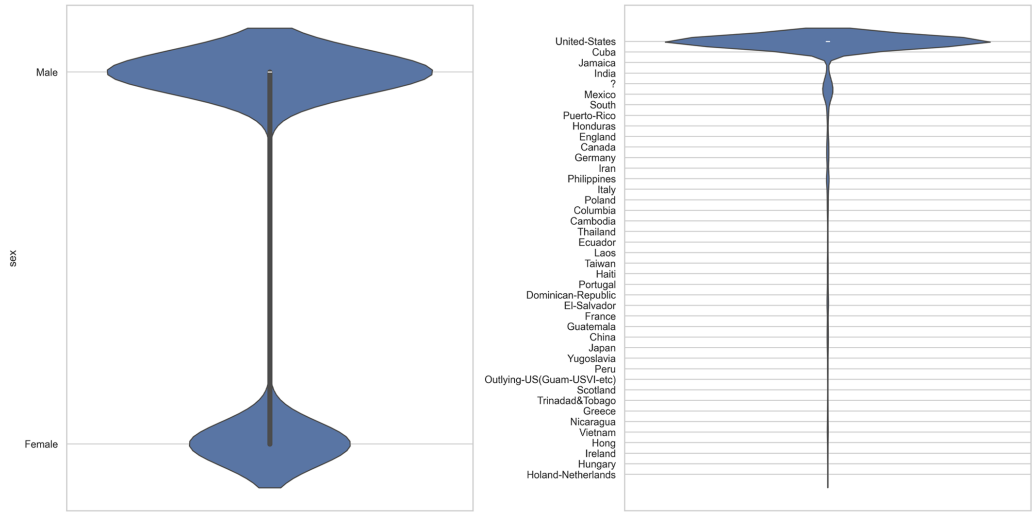

Figure 4.3: Frequency of different categories within each categorical column in the Adult dataset

4.1.2 Air Quality dataset

The Air Quality dataset [46], contains data collected from an air quality chemical multisensor device deployed in a polluted area within an Italian city. The dataset encompasses hourly measurements recorded over a one-year period, from March 2004 to February 2005. It includes sensor responses alongside ground-truth concentrations for various air pollutants, including carbon monoxide (CO), non-methane hydrocarbons, benzene, total nitrogen oxides (NOx), and nitrogen dioxide (NO2). This dataset provides valuable insights into air quality monitoring and pollution assessment using chemical sensor devices. Figure 4.4 shows a study of the distribution of numerical values in the dataset.

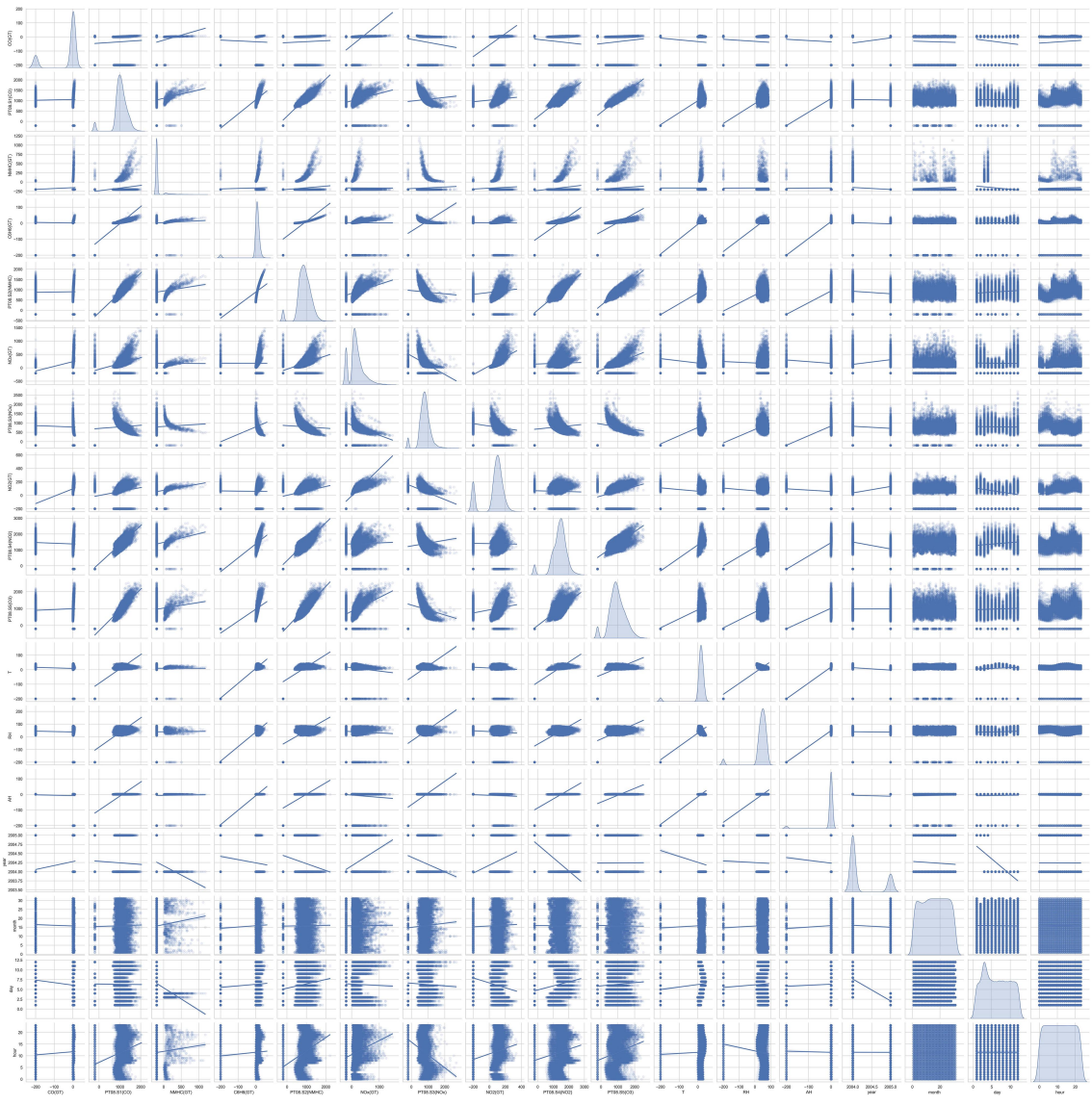

Figure 4.4: Distribution and correlation of numerical values in the Air Quality dataset

4.1.3 Apartment for Rent dataset

The Apartment for Rent Classified dataset [47], offers a collection of classified advertisements for rental apartments in the United States. It encompasses information for 10,000 distinct apartments, characterized by some features. These features include details typically found in rental listings, such as the number of bedrooms and bathrooms, along with the monthly rental price. This dataset presents valuable opportunities for applying machine learning techniques like classification, regression,

and clustering to analyze rental market trends and gain insights into factors influencing rental pricing. Figures 4.5 and 4.6 show an analysis of the distribution of a number of columns in the datasets.

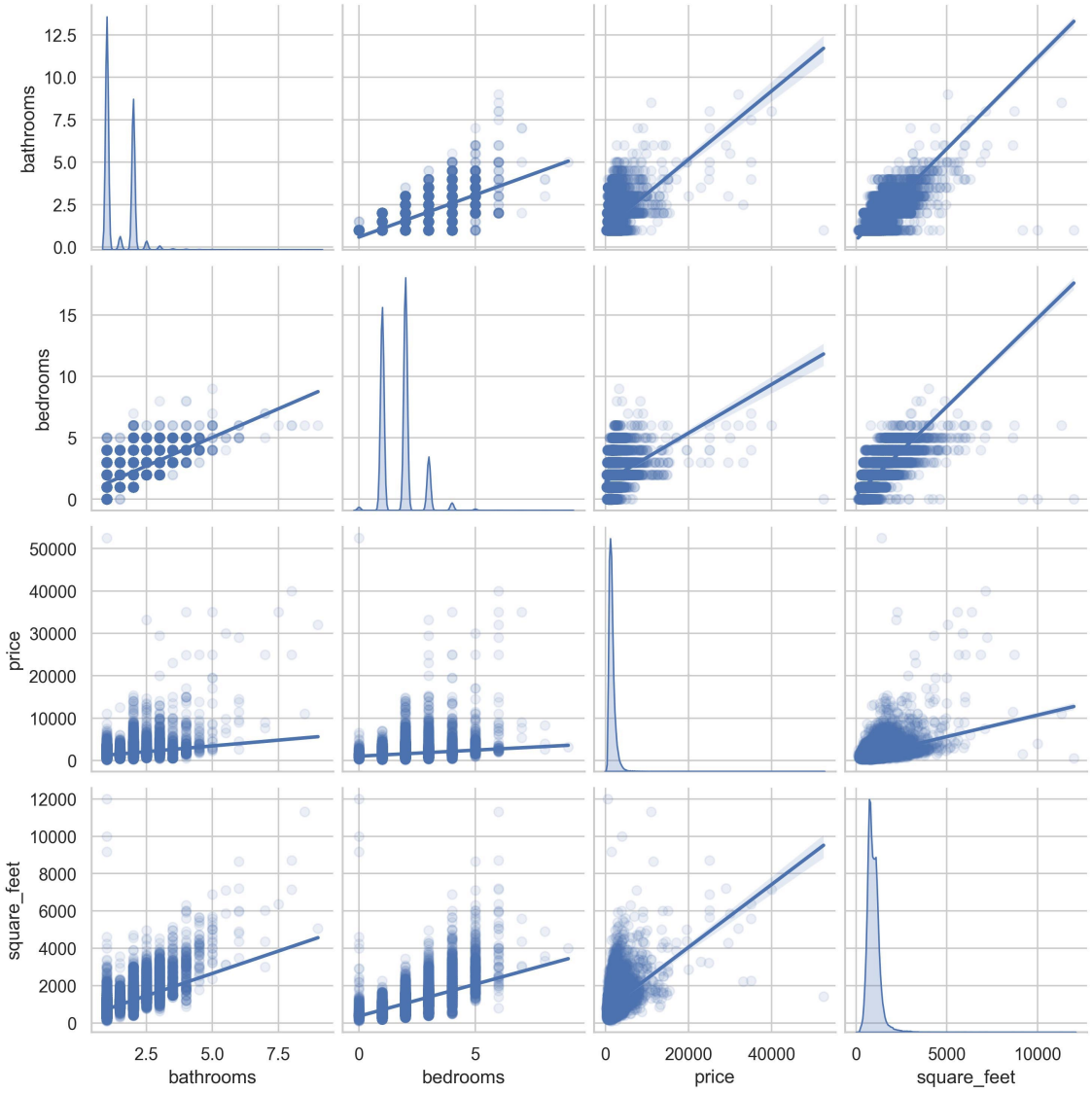

Figure 4.5: Distribution and correlation of numerical values in the Apartment dataset

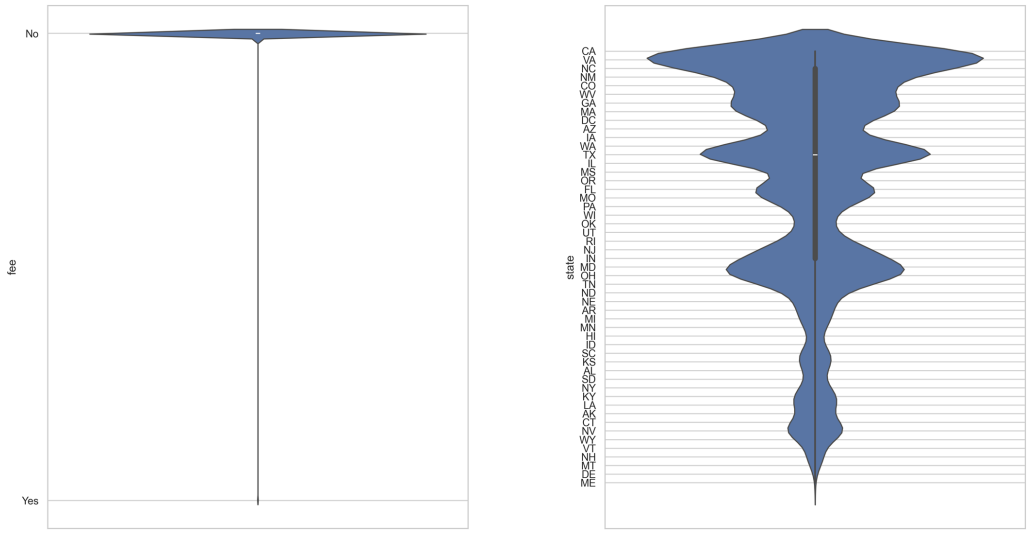

Figure 4.6: Frequency of different categories within each categorical column in the Apartment dataset

4.1.4 Bank Marketing dataset

The Bank Marketing dataset [48], provides information related to marketing campaigns conducted by a Portuguese bank to promote term deposits. It encompasses data for over 4,500 bank clients, including demographic details, contact information, and details regarding the marketing campaign they were part of. Most importantly, the dataset indicates whether each client subscribed to a term deposit or not, making it valuable for analyzing the effectiveness of marketing campaigns and identifying factors that influence customer decisions. As shown in figures 4.7 and 4.8, the datasets' distribution of values has been analyzed.

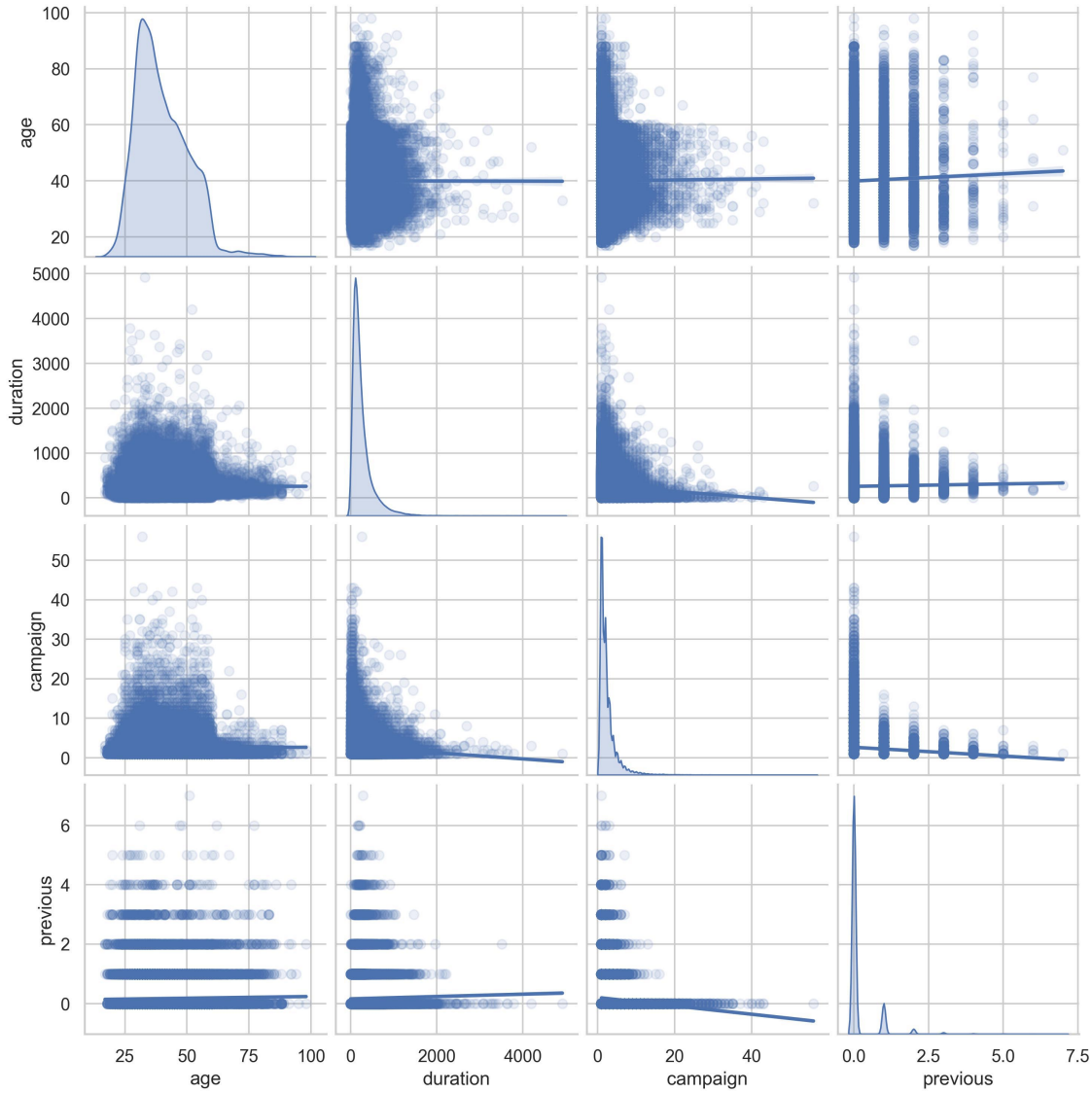

Figure 4.7: Distribution and correlation of numerical values in the Bank Marketing dataset

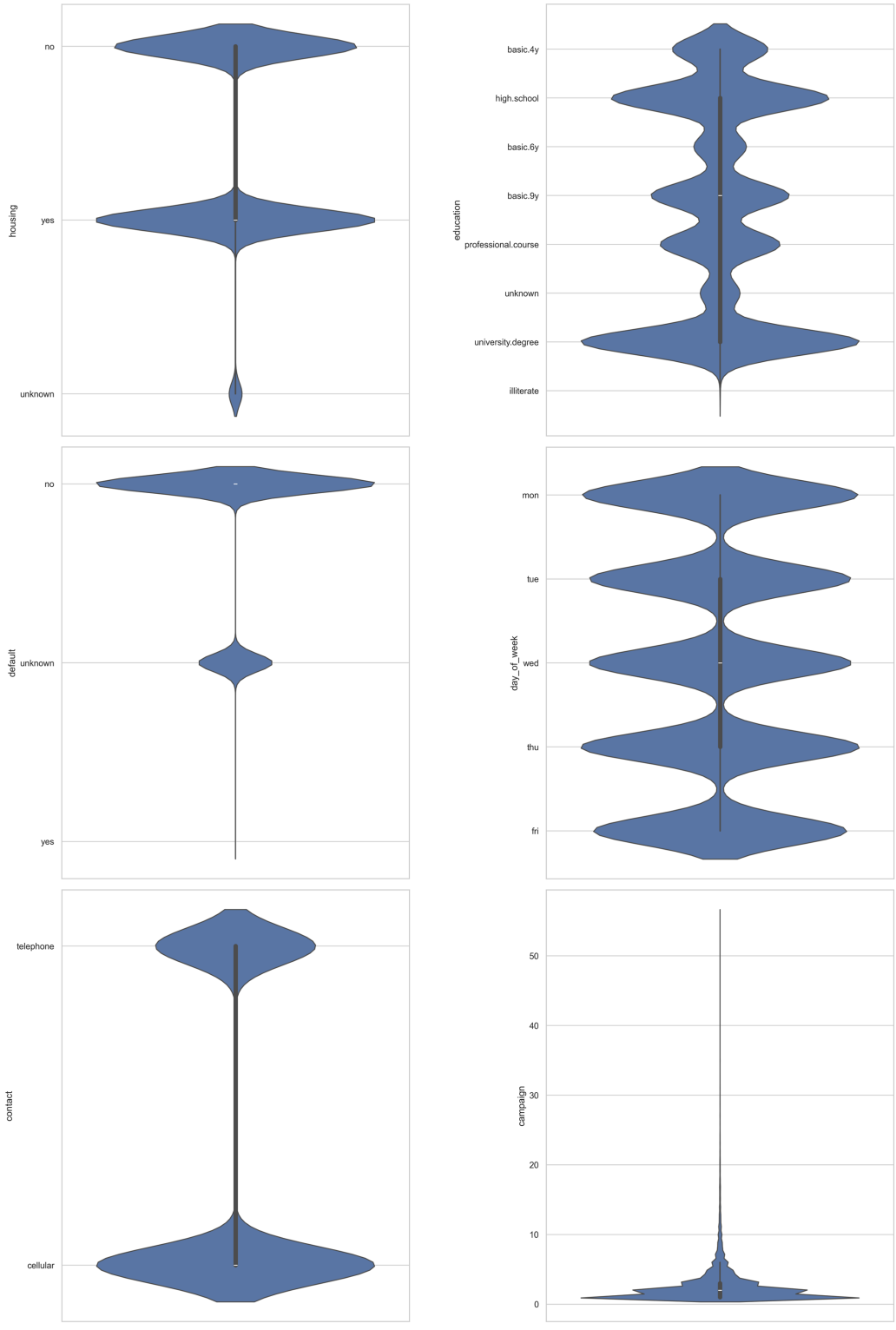

Figure 4.8: Frequency of different categories within each categorical column in the Bank Marketing dataset

4.1.5 Beijing PM2.5 dataset

The Beijing PM2.5 dataset [49] offers hourly measurements of PM2.5 concentration levels in Beijing, China, alongside corresponding meteorological data. The PM2.5 concentration data was collected by the US Embassy in Beijing, while the meteorological data originates from Beijing Capital International Airport. The dataset spans a five-year period, ranging from January 1st, 2010 to December 31st, 2014. In addition to PM2.5 concentrations, it encompasses various meteorological features like dew point, temperature, pressure, wind direction, and wind speed. This dataset provides valuable resources for researchers studying air quality, environmental science, and the impact of weather conditions on pollution levels. The analysis of the datasets' value distributions is shown in 4.9.

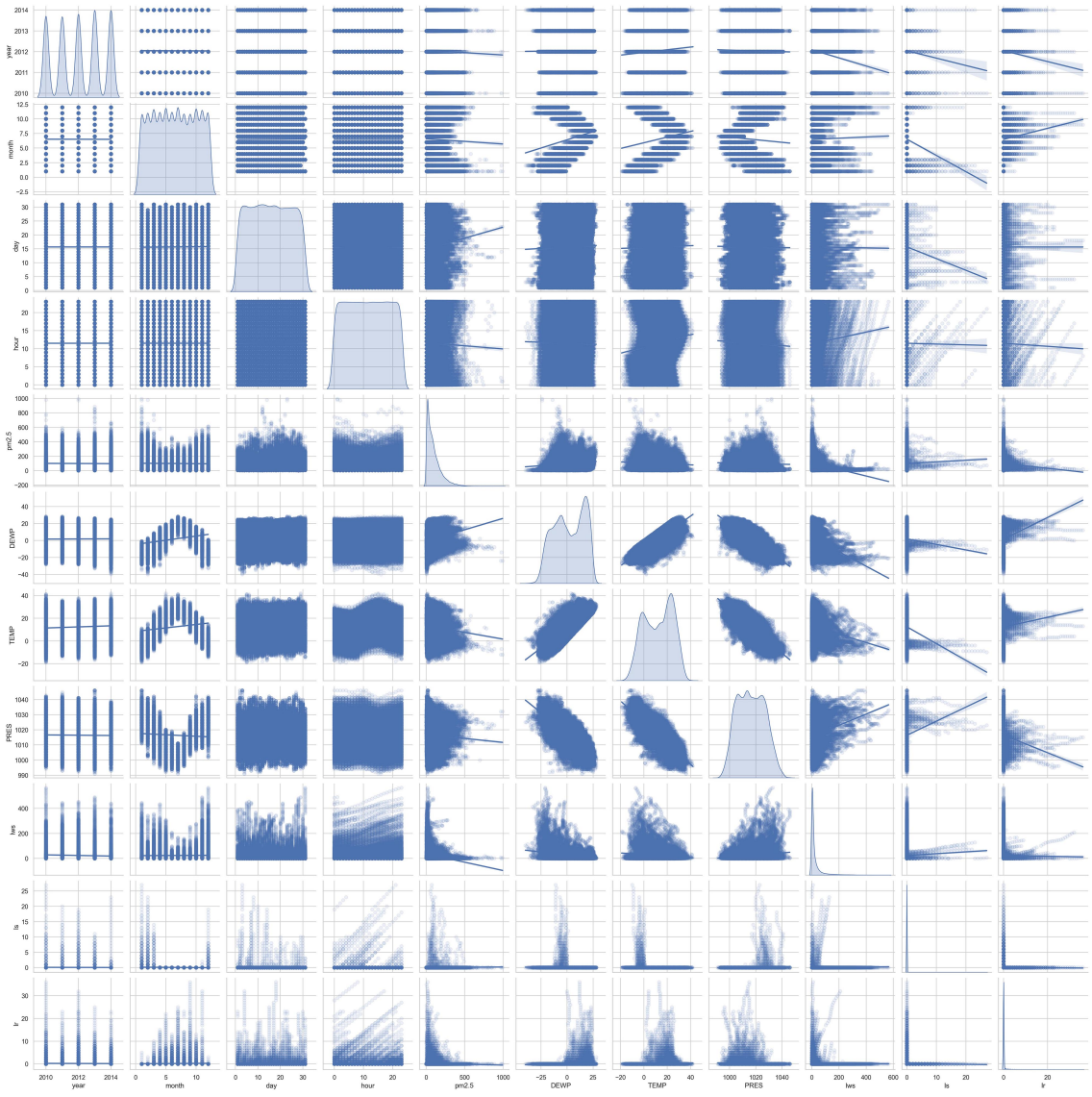

Figure 4.9: Distribution and correlation of numerical values in the Beijing PM2.5 datasets

4.1.6 Bike Sharing Dataset

The Bike Sharing dataset [50] offers a dataset containing information about bike rentals from the Capital bikeshare system, spanning the period from 2011 to 2012. This dataset encompasses various factors influencing ridership, including weather conditions and seasonality. It presents valuable resources for researchers studying traffic patterns, environmental concerns, and the health benefits associated with cycling. Figure 4.10 show a study of the distribution of numerical columns in the dataset.

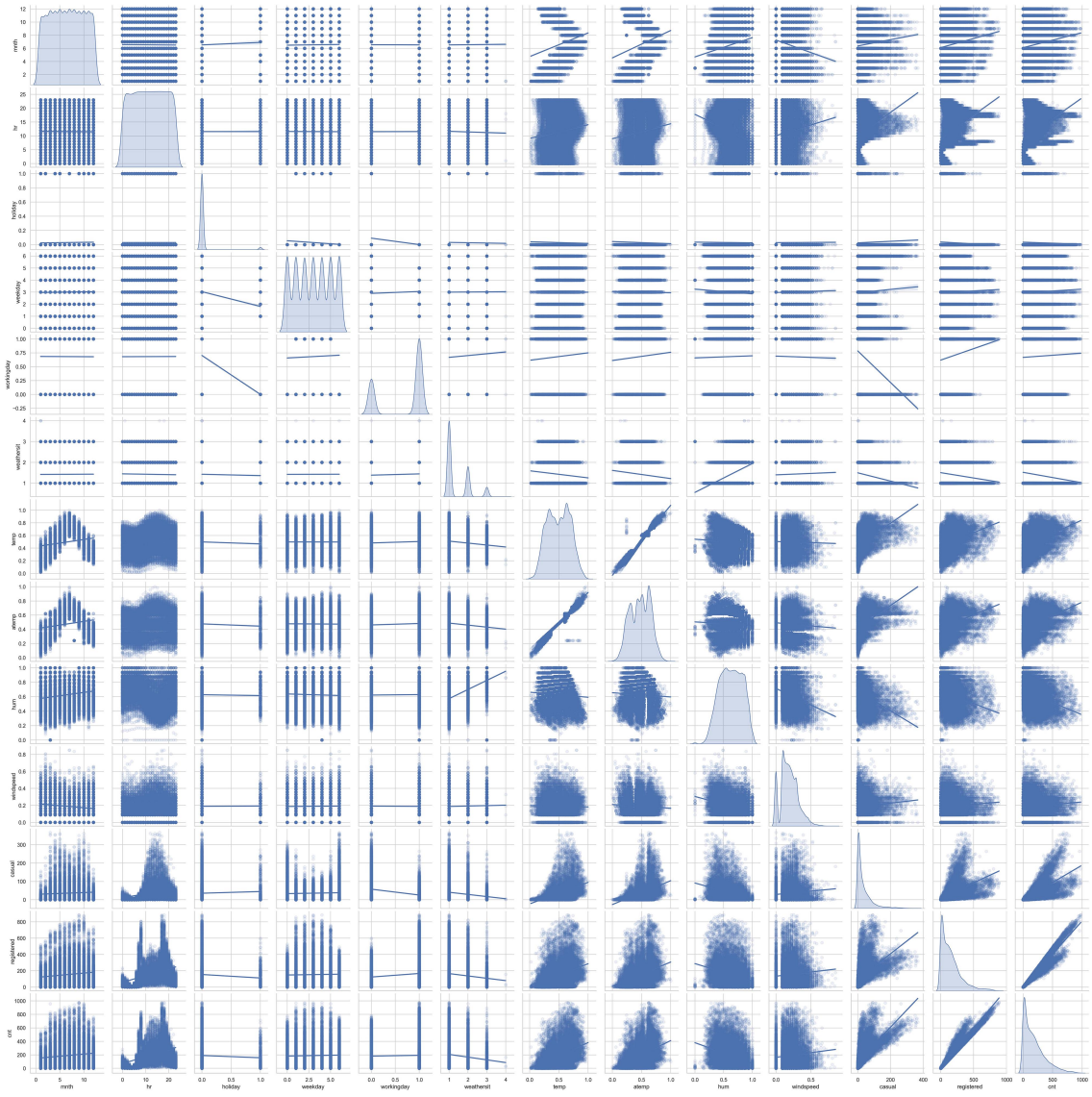

Figure 4.10: Distribution and correlation of numerical values in the Bike Sharing datasets

4.1.7 Individual Household Electric Power Consumption Dataset

The Individual Household Electric Power Consumption dataset [51] contains detailed measurements of electric power consumption within a single household. Collected over a period of almost four years with a one-minute sampling rate, the dataset includes various electrical quantities and timestamps. Additionally, it provides submetering values, offering insights into the power usage of specific appliances or areas within the house. Researchers can use this dataset for tasks like energy load fore-

casting, anomaly detection, and analyzing energy consumption patterns. As shown in figure 4.11, the datasets' distribution of values has been analyzed.

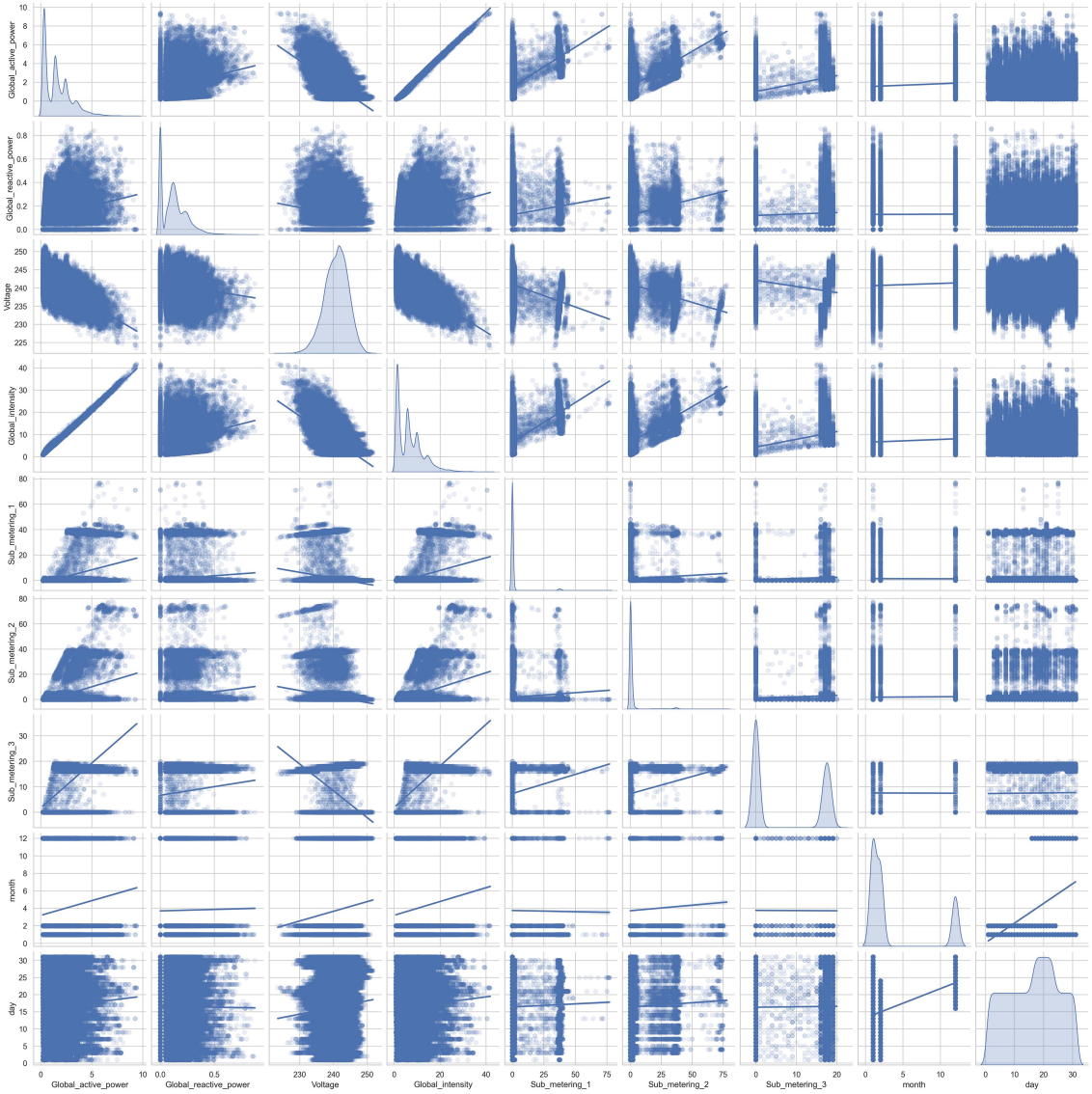

Figure 4.11: Distribution and correlation of numerical values in the Household Electric Power Consumption datasets

4.1.8 Metro Interstate Traffic Volume Dataset

The Metro Interstate Traffic Volume dataset [52] is a collection of hourly traffic volume data specifically for westbound I-94 in Minneapolis, Minnesota. It encompasses data ranging from 2012 to 2018, and includes additional features like weather conditions and holidays. It offers valuable resources for researchers studying traffic

flow, congestion prediction, and the impact of weather and holidays on traffic patterns. As shown in figures 4.12 and 4.13, the datasets' distribution of values has been analyzed.

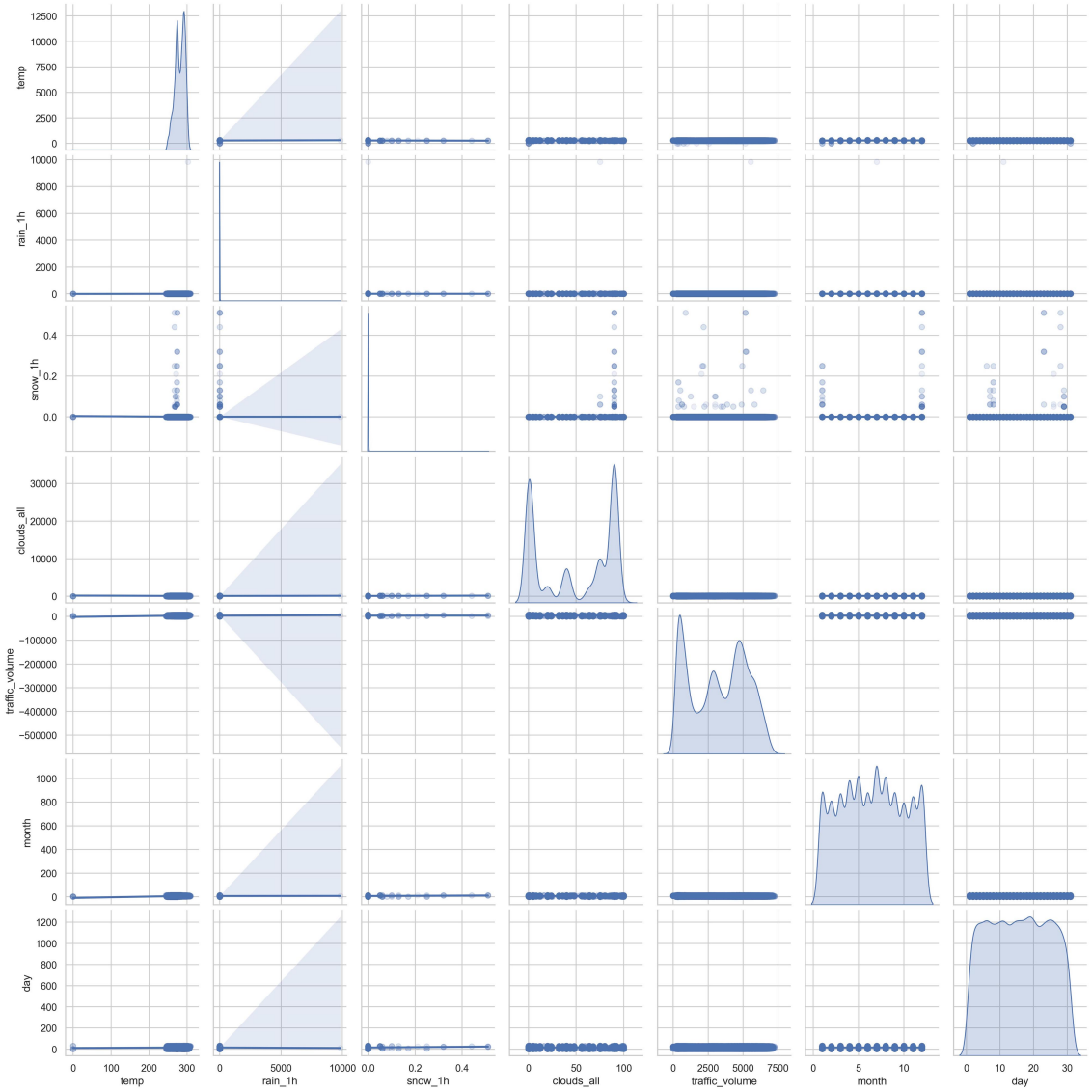

Figure 4.12: Distribution and correlation of numerical values in the Metro Interstate Traffic Volume datasets

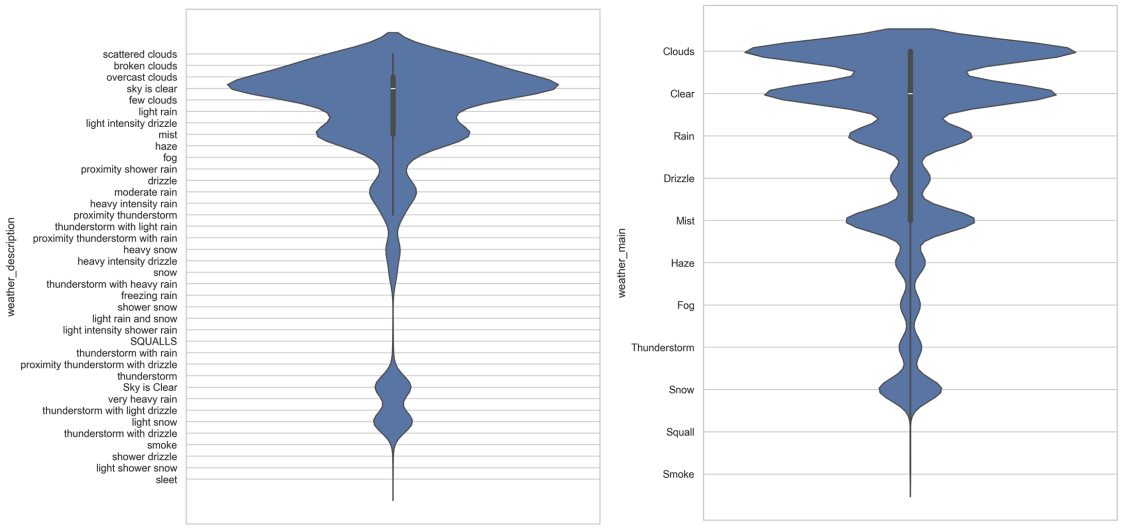

Figure 4.13: Frequency of different categories of categorical columns in the Metro Interstate Traffic Volume dataset

4.1.9 MetroPT-3 Dataset

The MetroPT-3 dataset [53] contains measurements collected from a compressor's Air Production Unit (APU) onboard a metro train. Data includes readings from various sensors monitoring pressure, temperature, motor current, and air intake valves. This dataset provides information in an operational context and is designed primarily for predictive maintenance. It can be used by researchers to develop machine learning models that can detect early signs of degradation or malfunction within metro train systems, thereby facilitating proactive maintenance strategies. Figure 4.14 shows a study of the distribution of values in the dataset.

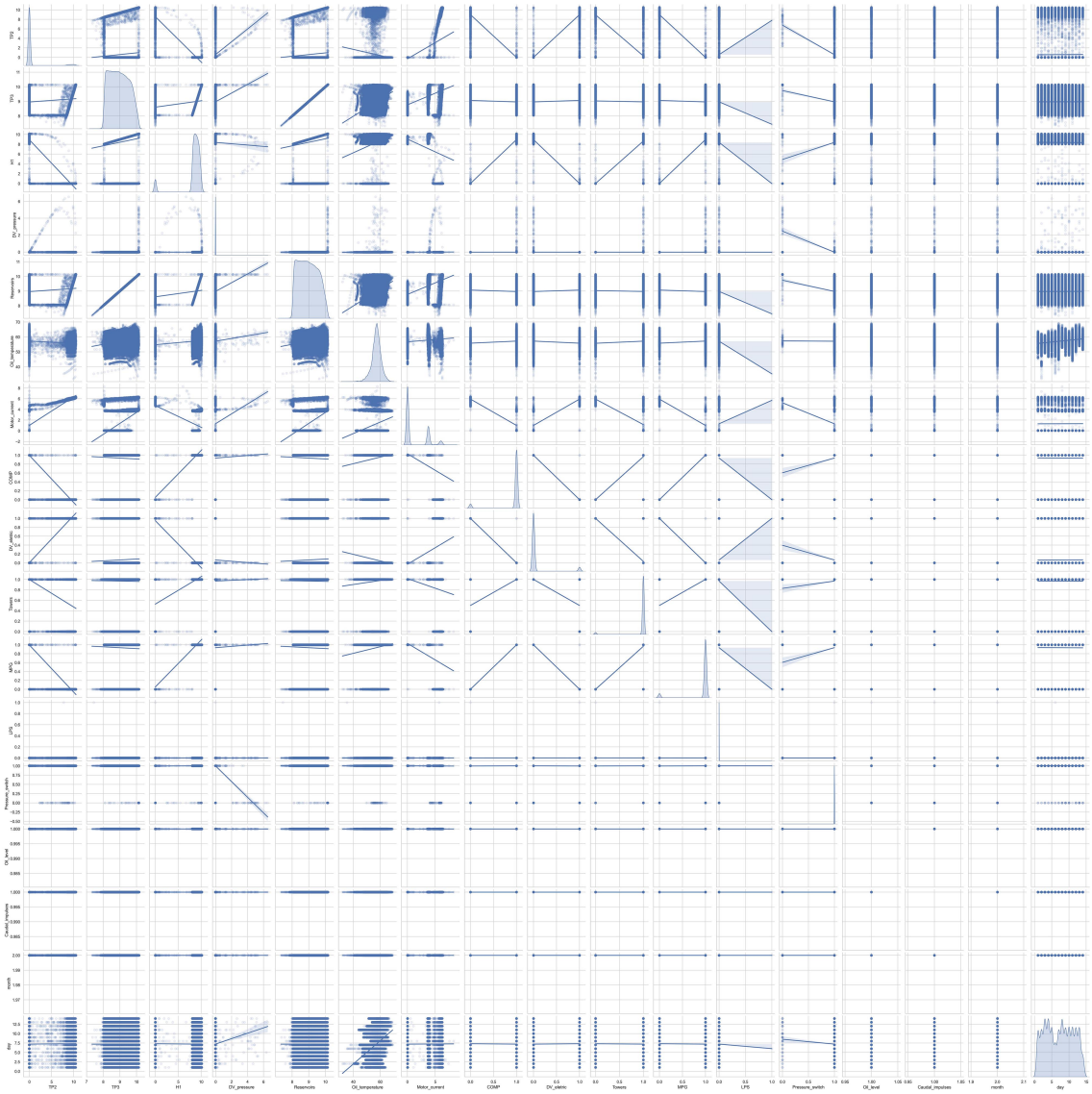

Figure 4.14: Distribution and correlation of numerical values in the MetroPT-3 datasets

Following preprocessing and necessary data cleaning procedures, the resulting datasets shapes are presented in Table 4.1.

Table 4.1: Datasets shape

Dataset	Rows	Categorical	Numerical	Context-specific
Household Power Consumption	2,049,280	2	7	2
MetroPT-3	1,516,948	10	7	2
Apartment for Rent	99,004	4	3	2
Metro Interstate Traffic	48,204	5	5	2
Beijing PM	41,757	5	7	2
Bank Marketing	41,188	12	3	1
Adult	32,561	8	6	1
Bike Sharing	17,379	6	7	2
Air Quality	9,357	0	13	2

Overall, the distinct patterns observed in the datasets underscore the importance of being aware of the context of the data when generating synthetic data using the CA-CTGAN model. By capturing and reproducing these patterns, the CA-CtGAN model can generate context-aware, high-quality synthetic data that accurately represents the complexities and intricacies of real datasets.

4.2 Baselines and Experimental Setup

In this study, our evaluation of CA-CTGAN's performance involves a comparison with three leading GAN-based models for generating tabular data: CTGAN [30], DATGAN [31], and CTAB-GAN [1]. Our choice to select these baseline is strategically informed by a nuanced understanding of the existing landscape in GAN-based tabular data generation, as detailed in the introduction and related work sections of this dissertation. These models are primarily designed with an emphasis on data privacy, implementing mechanisms to prevent the generation of data that directly mirrors real-world scenarios. This approach inherently includes parameters that allow for the adjustment of privacy levels to avoid producing data too closely resembling the

original datasets. In contrast, CA-CTGAN diverges significantly in its foundational premise and application. Unlike these baselines, CA-CTGAN prioritizes the generation of synthetic data that closely aligns with real-world conditions. This distinction is critical for our objectives; we aim to simulate real experiments as accurately as possible, especially under conditions not present in the original data, to unearth insights into scenarios that remain unexplored. Therefore, while the baseline models offer flexibility in managing privacy concerns through adjustable parameters, our approach involves minimizing these privacy constraints to enable a direct comparison.

To ensure a fair and consistent comparison, we employed the official implementations of these models, as provided by their original authors. Our experimental setup was powered by a machine equipped with 64 GB of memory, dual 16 GB GeForce RTX A4000 GPUs, and an Intel Xeon W-2255 CPU (3.0 GHz x 20 cores). We adhered to the recommended hyperparameters specified by each model's creators to optimize performance and maintain consistency across tests.

For the Contractive Autoencoder (CAE), critical for our CA-CTGAN framework, we allocated 10% of each dataset for training. This approach was designed to ensure a balanced representation of categories, particularly for datasets with categorical variables, by utilizing a log-frequency-based sampling method. This technique aims to minimize skewness in the representation of categories within each categorical column. Algorithm 4 shows the steps we implemented for log-frequency-based sampling. The remainder of the data was used for training the models. For evaluation, we generated 5000 rows from each dataset from scratch, and compared the synthetic datasets with the real ones using the metrics detailed in the subsequent section. This testing phase included scenarios designed to assess controlled generation capabilities, highlighting the context-based generation potential of CA-CTGAN. Unlike baseline models that may face challenges due to their sample rejection mechanisms potentially lowering the likelihood of generating data that meets specific conditions CA-CTGAN aims to

Algorithm 4 Log-Frequency Based Sampling of Categorical Data

Input: Dataset D with categories $C = \{c_1, c_2, \ldots, c_N\}$ and their frequencies $F = \{f_1, f_2, \ldots, f_N\}$
Output: Sampled dataset S using log-frequency weights
procedure LogFrequencySampling(D, F)
    Initialize an empty list L for log-transformed frequencies
    Initialize an empty list W for weights of categories
    for each category frequency f_i in F **do**
        Compute log-transformed frequency $l_i = \log(f_i + 1)$
        Append l_i to L
    end for
    Compute total log-transformed frequency $L_{total} = \sum_{i=1}^{N} L[i]$
    for each l_i in L **do**
        Compute weight $w_i = \frac{l_i}{L_{total}}$
        Append w_i to W
    end for
    Sample from D using weights W to create sampled dataset S
    return S
end procedure

overcome these limitations, providing more consistent results under targeted conditions.

By comparing CA-CTGAN with established baseline models across selected datasets, this study demonstrates CA-CTGAN's superior ability to generate realistic and contextually relevant synthetic data. This underscores its potential to effectively replace actual experiments in a variety of domains, marking a significant advancement in the field of synthetic data generation.

4.3 Evaluation Metrics

In generative models, evaluation methods cannot be generalized to other contexts; instead, they must be evaluated explicitly based on their application. In Generative Models optimization, Gaussian distributions are fitted to a mixture of Gaussian distributions by minimizing distance measures such as Maximum mean discrepancy (MMD) [54] and Jensen-Shannon divergence (JSD). Minimizing MMD or JSD results in the omission of some modes in a multimodal distribution. In addition, maximizing

average log-likelihood or minimizing KL-divergence can assign large probabilities to non-data regions. In image synthesizing applications, three common criteria are used to evaluate generative models: log-likelihood, Parzen window estimates, and visual fidelity of samples [55]. However, the evaluation of results for tabular data with complex data types and distribution would be quite different.

In order to measure accuracy, a generated data should first demonstrate that it is a good representation of real data. The SDMetrics Python library [3] introduces a set of metrics to measure the quality and privacy of synthetic data. However, considering data privacy is not the goal of this study and may reduce the quality of the data. These metrics are summarized and reformed to make them suitable for evaluating the generated data. In order to achieve this objective, the comparison of two real and generated datasets can be divided into the following categories:

4.3.1 Data Coverage

For discrete columns D_i, we must determine whether all categories in the real data are represented in the generated. To accomplish this goal, a score is calculated by dividing the number of unique categories in the generated data by the number of unique categories in the corresponding column of the actual data as follows:

$$coverage_{Di} = \left(\frac{N_{D_g}}{N_{D_{data}}} \right)_i . \tag{4.1}$$

where i is the column index, N_{D_g} is the number of unique categories in the generated data, and $N_{D_{data}}$ is the number of unique categories in the real data. When a column is scored 1, all of the unique categories in the actual data are present in the generated data, while a score of 0 indicates that no unique categories are present in the generated data. In the case of continuous columns, the coverage metric is used to measure whether a generated column covers the whole range of values that can be found in

the real column. The coverage score for continuous columns is calculated as follows:

$$coverage_{Ci} = 1 - \left[\max\left(\frac{\min(C_g) - \min(C_{data})}{\max(C_{data}) - \min(C_{data})}, 0\right) + \max\left(\frac{\max(C_{data}) - \max(C_g)}{\max(C_{data}) - \min(C_{data})}, 0\right)\right]. \tag{4.2}$$

where C_g is the generated value and C_{data} is the real value of column C_i. The goal of this metric is to determine how closely the min and max of the generated values match the actual min and max values. It is possible for Equation (4.2) to become negative if the range covered by the generated data is inadequate, and in such a situation, it returns a score of 0 since this is the lowest possible result.

4.3.2 Data Constraint

In order to measure how a continuous column adheres to a boundary of real data, boundary adherence is introduced. The frequency of generated values within the minimum and maximum ranges of the real column values is calculated using this metric.

$$adherence_{Ci} = \frac{N_{(min<x_i<max)}}{N_i}. \tag{4.3}$$

where N_i is the number of records in column C_i. A column with a score of 1 indicates all values adhere to the boundaries of real data, while a column with a score of 0 indicates that no values fall between the minimum and maximum and 1 indicates that values fall between the minimum and maximum of the real data.

For discrete columns, we measure how well the generated data stays true to the original categories, ensuring no fabrication of new categories. The process involves extracting the real column's unique category set C_r, then determining the count of synthetic data points s belonging to C_r. The final score represents the proportion of conforming synthetic data points to the total synthetic dataset as follows:

$$adherence_{Di} = \frac{|s, s \in C_r|}{|s|} \tag{4.4}$$

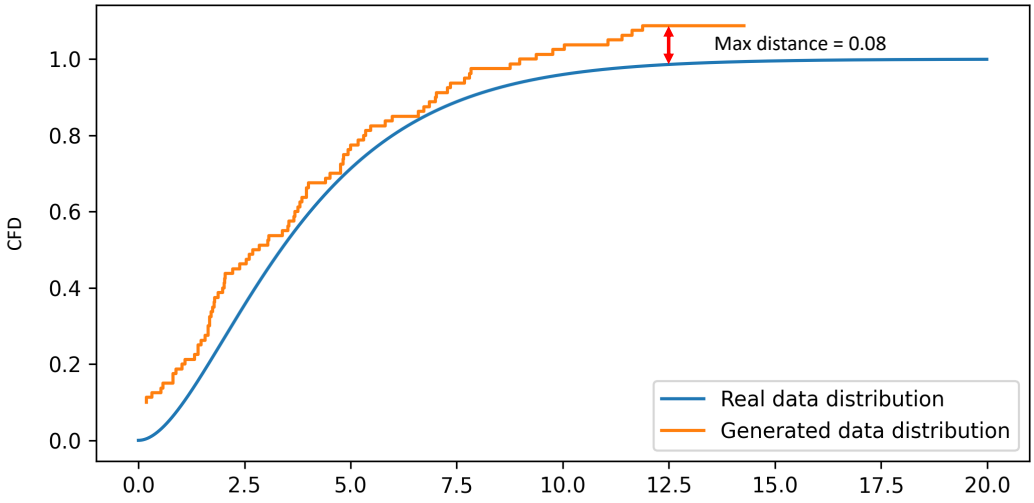

Figure 4.15: Distances are measured between 0 and 1, but the complement of this metric can also be considered. Therefore, a higher score indicates higher quality according to 1-(KS statistic distance) [3].

4.3.3 Data Similarity

The Synthetic Data Metrics (SDMetrics) library [3] introduces several metrics for measuring data similarity. In order to calculate the similarity between real and generated marginal distributions, two types of metrics are available: the Kolmogorov–Smirnov (KS) statistic for continuous columns and the Total Variation Distance (TVD) for discrete columns. Based on the KS statistic, we can determine how much the empirical distribution function of the generated data differs from the Cumulative Distribution Function (CDF) of the real data. This means that in this case, the KS statistic represents the maximum difference between the two generated and real CDFs, as illustrated in Figure 4.15. The KS statistic can be calculated using the following expression:

$$KS_{data,g} = \sup_{x} \left| F_{1,data}(x) - F_{2,g}(x) \right|. \tag{4.5}$$

where $F_{1,data}$ and $F_{2,g}$ are the cumulative distribution functions of the real and generated data, respectively, and *sup* is the supremum function.

Based on TVD, we can measures the maximum difference between the corresponding probabilities of each category across the generated and real column. In order to calculate the TVD statistic, we first computes the frequency of each category value in the real and generated columns and express them as probabilities. Once the frequencies are calculated, the TVD statistic compares the difference in probabilities using the following formula:

$$TVD_{data,g} = 1 - \delta(X, G) = 1 - \frac{1}{2} \sum_{x \in D_{data}} |X_{x,g} - G_{x,g}|. \tag{4.6}$$

where x and g refer to all possible categories in discrete column D, and X and G represent the frequencies for those categories for real and generated data, respectively. The similarity score is considered the complement of a TVD, so a higher score indicates a higher level of quality.

In addition, it is possible to measure the statistical similarity between a column of real data and a column of generated data using mean, median, and standard deviation using the following formula:

$$similarity = \max\left(1 - \frac{|f(x) - f(g)|}{|\max(x) - min(x)|}, 0\right). \tag{4.7}$$

where an arithmetic mean, median, or standard deviation is defined as f, and it returns a score between 0 and 1, where a high value represents a high degree of similarity.

In evaluation metrics, the similarity between real and synthetic discrete columns is also examined using the Chi-squared test. This statistical test serves as a rigorous method for comparing category frequencies by first normalizing the data from both real and generated datasets. By executing the Chi-squared test, we assess the null

hypothesis which states that the synthetic data and real data both originate from the same distribution. The core of this test lies in its output, the p-value, which acts as a measure of similarity: a higher p-value (approaching 1) suggests negligible differences between the synthetic and real datasets, thus supporting the hypothesis of common distributional origins. Conversely, a lower p-value (tending towards 0) signals significant discrepancies, leading to the rejection of the null hypothesis.

4.3.4 Data Relationship

For measuring the semantic relationship and correlation between columns within a dataset, the contingency can be applied to discrete columns using crosstabulation (also known as a contingency table). This score is a matrix representation of the multivariate frequency distribution of variables. First, two contingency tables should be created over the categories present in each column in order to compare a discrete column in the real data with the corresponding column in the generated data. Indeed, the created tables summarize the proportion of rows in real and generated data that have each combination of categories. After that, the total variation distance is used to calculate the difference between the contingency tables. In this case, the distance would be between 0 and 1, so subtracting 1 from the score would indicate a high degree of similarity. Below is a formula that summarizes the process.

$$contingency_{x,g} = 1 - \frac{1}{2} \sum_{x \in D_{data}} \sum_{g \in D_g} |X_{x,g} - G_{x,g}|. \tag{4.8}$$

where x and g refer to all possible categories in discrete column D, and X and G represent the frequencies for those categories for real and generated data, respectively. A score of 1 indicates the best contingency between real and generated data, and a score of 0 indicates the worst contingency. Also, a correlation similarity test can be applied to continuous columns by measuring the correlation between two numerical columns and computing the similarity between the real and generated data using

Pearson's and Spearman's rank coefficients. Initially, a correlation coefficient should be calculated between two continuous columns in the real data and their corresponding columns in the generated data. Then, after normalizing two correlation values, the following equation returns a similarity score.

$$correlation_{x,g} = 1 - \frac{|X_{x,g} - G_{x,g}|}{2}. \tag{4.9}$$

where x and g refer to all values in continuous column C, and X and G represent the distributions for real and generated data, respectively. In this score, the correlation between the columns is bounded between -1 and 1, with -1 representing the most negative correlation and 1 representing the most positive correlation between the real and generated columns.

4.3.5 ML Detection

This metric measures the difficulty of differentiating between real and generated data. By employing a machine learning classification model such as Logistic Regression and SVM, the model can better predict the nature of each row in the dataset. The final score, based on the average ROC AUC score, provides an indication of the model's performance in distinguishing between real and generated data. The score ranges between 0 and 1, where 1 indicates that the machine learning model cannot differentiate between real and generated data. If the model cannot reliably distinguish synthetic data from real data, this suggests that the synthetic data accurately captures the essential statistical properties and behaviors of the real dataset. High realism is crucial for ensuring that synthetic data can serve as a stand-in for real data in sensitive or inaccessible scenarios.

Also, one of the primary applications of synthetic data is to augment or replace real datasets in training machine learning models, particularly when data collection is limited or privacy concerns preclude the use of real data. The ML Detection metric

Algorithm 5 Algorithm for ML Detection

Input: Real data, Generated data
Output: Final score of ML model
procedure ML DETECTION( )
Augment data by combining all rows of real and synthetic data into a single table with an additional column to indicate whether each row is real or synthetic
for $i \leftarrow 1$ to Number of Cross-Validation Folds **do**
Split the augmented data into training and validation sets
Create ML classification model with specified parameters
Train the model on the training set to predict the extra column added in Step 1
Validate the model on the validation set
Calculate ROC AUC score for this fold
end for
Calculate the average ROC AUC score across all the cross-validation folds
return Final score: $1 - (\max(ROCAUC, 0.5) \times 2 - 1)$
end procedure

provides a quantifiable measure of whether synthetic data can be used interchangeably with real data without degrading the performance of machine learning models. The procedure of this metric is shown in algorithm 5.

4.3.6 ML Efficiency

ML Efficiency determine the utility of synthetic data in training machine learning models for predictive analytics. The metric varies depending on the algorithm used for computation and the nature of the prediction task (binary or multiclass classification) [3].

We employ several algorithms, each offering unique strengths and perspectives on the predictive capabilities of the synthetic data and finally we chose the AdaBoost Classifier. To this aim, model is trained exclusively on synthetic data, generating a model calibrated to predict the outcomes of a target variable.

The procedure for this metric is methodical: the selected machine learning algorithm is first trained on the synthetic dataset to develop a predictive model. This model

is then tasked with making predictions on a real dataset, which serves as the testing ground to validate the model's accuracy. The performance of the model is quantified using Accuracy and F1 score.

This ML Efficacy metric is crucial, as it provides direct insights into the practical applicability of synthetic data in predictive modeling, reflecting the CA-CTGAN model's potential to serve as a surrogate for real data in the training of reliable machine learning models.

CHAPTER 5: RESULT AND ANALYSIS

In this chapter, we delve into the outcomes of our experiments, starting with the performance results from CA-CTGAN models, which have been trained across 11 datasets and assessed using introduced evaluation metrics in section 4.3.6 . Following this initial presentation, we engage in a detailed comparative analysis, positioning the CA-CTGAN model against a selection of established baseline models across the datasets. The following sections detail the findings for each dataset, offering insights into the strengths and potential limitations of CA-CTGAN in the context of synthetic tabular data generation.

Throughout this analytical journey, we have embarked on an extensive exploration, evaluating all columns within the datasets and generating a comprehensive suite of reports and visualizations. Due to the breadth and depth of this evaluation, it is impractical to incorporate the entirety of these visual and analytical outputs within this single chapter. Therefore, we will highlight the most critical findings and takeaways based on the experiments conducted. Readers interested in a more detailed examination are encouraged to explore the official source code [56].

5.1 Model Performance

5.1.1 Data Coverage

In our analysis for evaluating the Data Coverage (section 4.3.1), the CA-CTGAN model demonstrated exceptional performance, achieving near-perfect scores for most columns like Voltage column in Power Consumption dataset as shown in figure 5.1. Category Coverage score for discrete columns and Range Coverage score for continuous columns are represented in tables 5.1, 5.2 and 5.3. These metric measures whether

a synthetic column covers all the possible categories and full range of values that are present in a real column respectively.

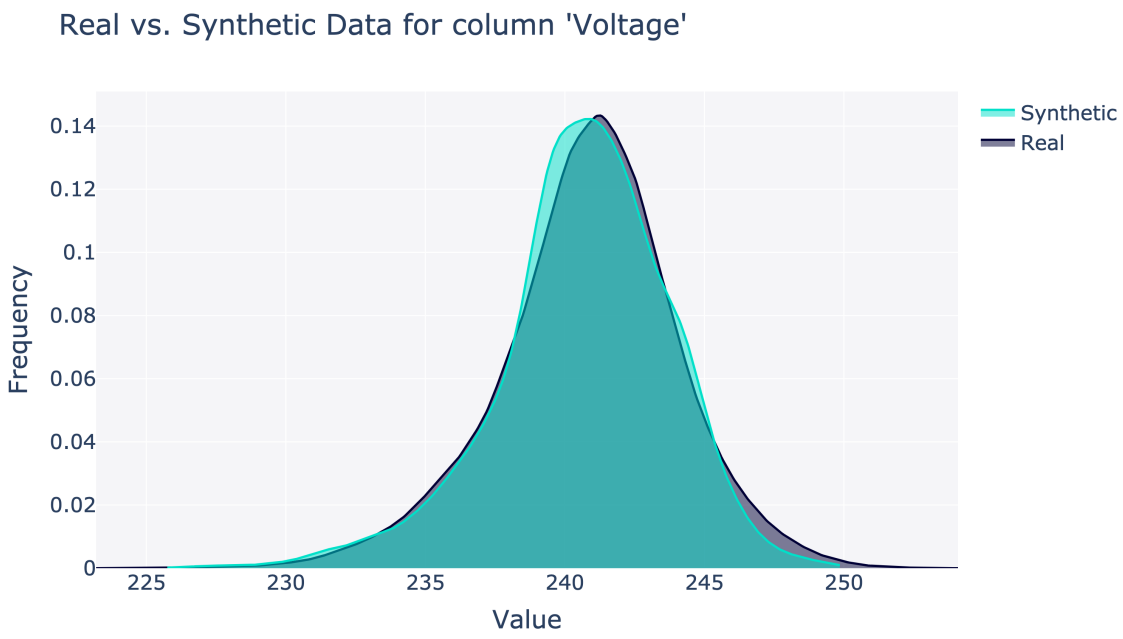

Figure 5.1: Data distribution for Voltage column in Power Consumption dataset.

The Data Coverage metric's performance is inherently influenced by the underlying distribution of the dataset, including the diversity of categories in discrete columns and the presence of outliers in continuous columns. Our findings underscore that while CA-CTGAN exhibits exceptional capability in generating synthetic data that covers the breadth of categories and valu es found in real-world datasets, its efficacy can vary based on specific data characteristics.

In columns with a broad yet imbalanced distribution of categories, such as 'country' in adult dataset and 'holiday' in Metro Interstate Traffic dataset (Figure 5.2), the model demonstrates high data coverage, effectively replicating minority categories with significant accuracy as shown in Figure 5.3.

Table 5.1: Data Coverage score for Adult, Bank Marketing, and Metro PT-3 datasets.

Adult		Bank Marketing		Metro PT-3	
Column	Score	Column	Score	Column	Score
workclass	1.00	job	1.00	COMP	1.00
education	1.00	marital	1.00	DV_eletric	1.00
marital-status	1.00	education	1.00	Towers	1.00
occupation	0.93	default	1.00	MPG	1.00
relationship	1.00	housing	1.00	LPS	1.00
race	1.00	loan	1.00	Pressure_switch	1.00
sex	1.00	contact	1.00	Oil_level	1.00
native-country	0.90	month	1.00	Caudal_impulses	1.00
income	1.00	day_of_week	1.00	TP2	0.99
age	1.00	campaign	0.50	TP3	0.36
fnlwgt	0.68	poutcome	1.00	H1	0.99
education-num	1.00	y	1.00	DV_pressure	0.74
capital-gain	0.56	age	0.95	Reservoirs	0.87
capital-loss	0.54	duration	0.78	Oil_temperature	0.78
hours-per-week	0.93	previous	0.83	Motor_current	0.67

Table 5.2: Data Coverage score for Air Quality, Bike Sharing, and Metro Interstate Traffic.

Air Quality		Bike Sharing		Metro InterstateTraffic	
Column	Score	Column	Score	Column	Score
CO(GT)	0.98	holiday	1.00	holiday	0.92
PT08.S1(CO)	0.84	weekday	1.00	weather_main	1.00
NMHC(GT)	0.76	workingday	1.00	weather_description	1.00
C6H6(GT)	0.93	weathersit	1.00	temp	0.80
PT08.S2(NMHC)	0.80	temp	1.00	clouds_all	0.95
NOx(GT)	0.86	atemp	0.89	traffic_volume	1.00
PT08.S3(NOx)	0.90	hum	0.94		
NO2(GT)	0.97	windspeed	0.73		
PT08.S4(NO2)	0.98	casual	0.85		
PT08.S5(O3)	0.74	registered	0.98		
T	0.79	cnt	0.99		
RH	0.93				
AH	0.92				

Table 5.3: Data Coverage score for Apartment Rent, Power Consumption and Beijing PM datasets.

Apartment Rent		Power Consumption		Beijing PM	
Column	Score	Column	Score	Column	Score
bedrooms	0.80	Global_active_power	0.93	cbwd	1.00
fee	1.00	Global_reactive_power	0.84	pm2.5	0.66
cityname	0.79	Voltage	0.78	DEWP	0.91
state	0.94	Global_intensity	0.89	TEMP	0.89
bathrooms	0.86	Sub_metering_1	0.67	PRES	0.96
price	0.79	Sub_metering_2	0.72	Iws	0.78
square_feet	0.84	Sub_metering_3	0.95		

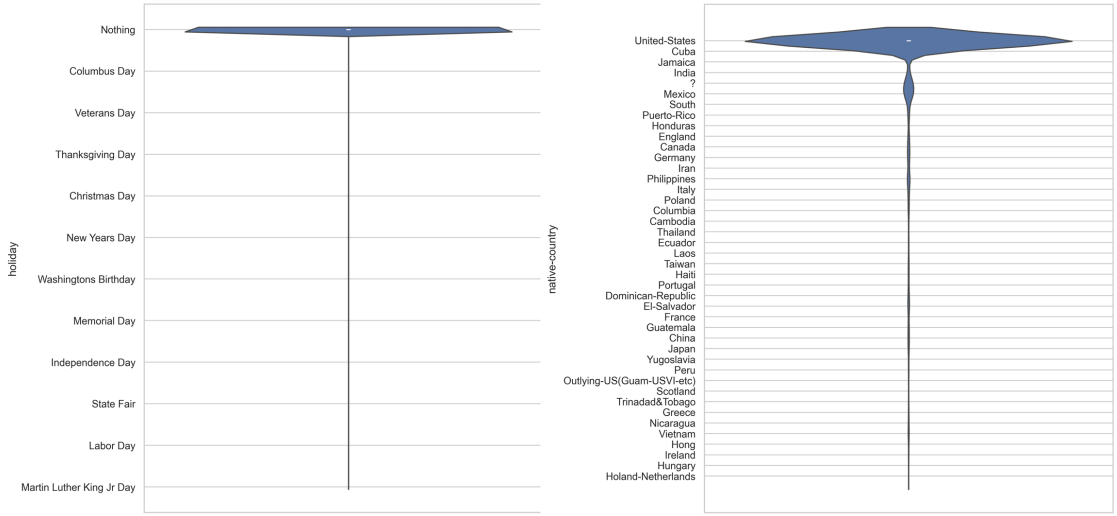

Figure 5.2: Frequency of categories in native-country column in Adult dataset and holiday column in Metro Interstate Traffic dataset.

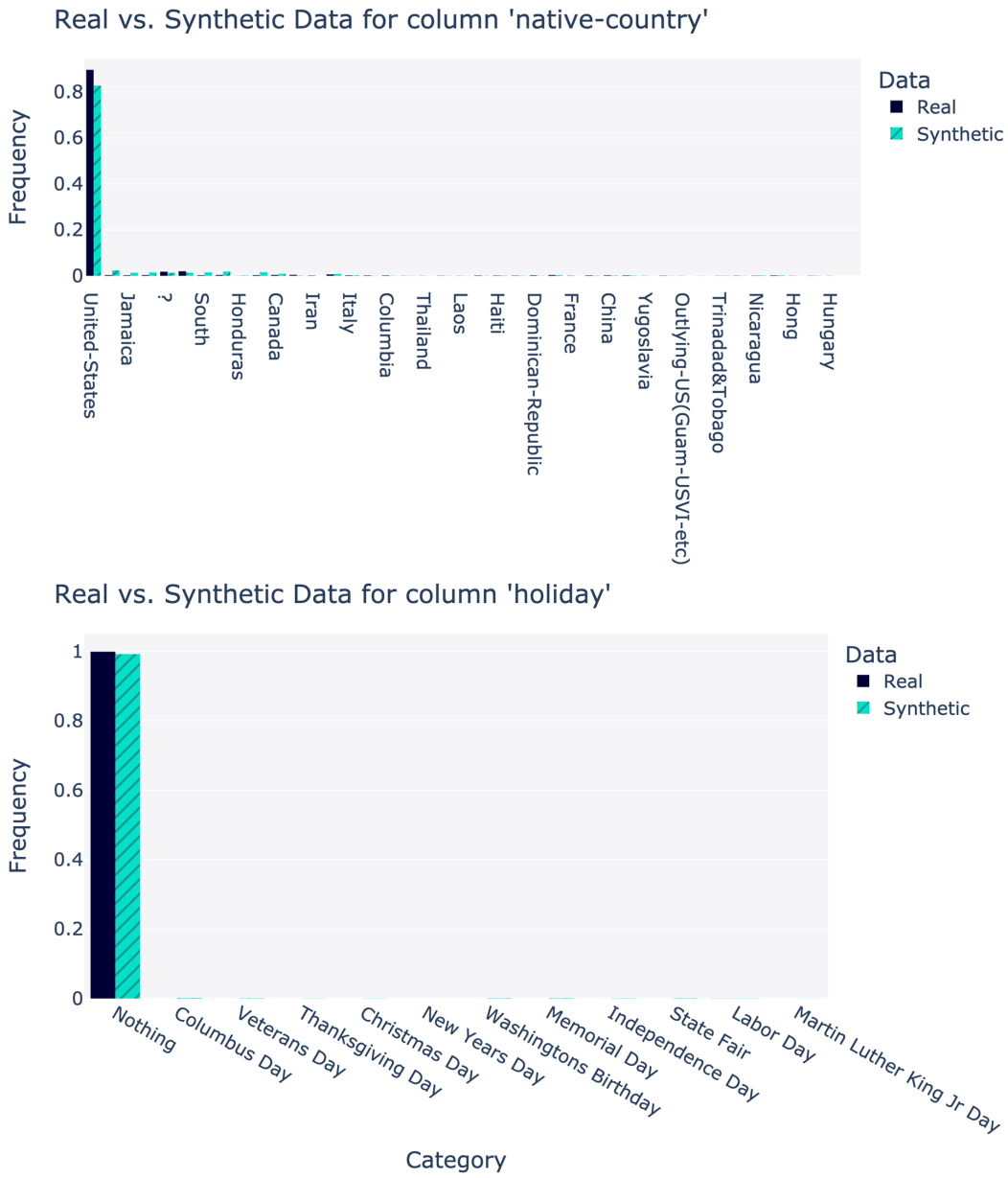

Figure 5.3: Comparison of data category coverage for native-country column in Adult and holiday column in Metro Interstate Traffic dataset.

Despite the fact that the 'capital-loss' column in the adult dataset displayed a significantly lower score of 0.54 in the presence of significant outliers, generated data perfectly captured both the skewness and the secondary mode the data distribution (Figure 5.4).

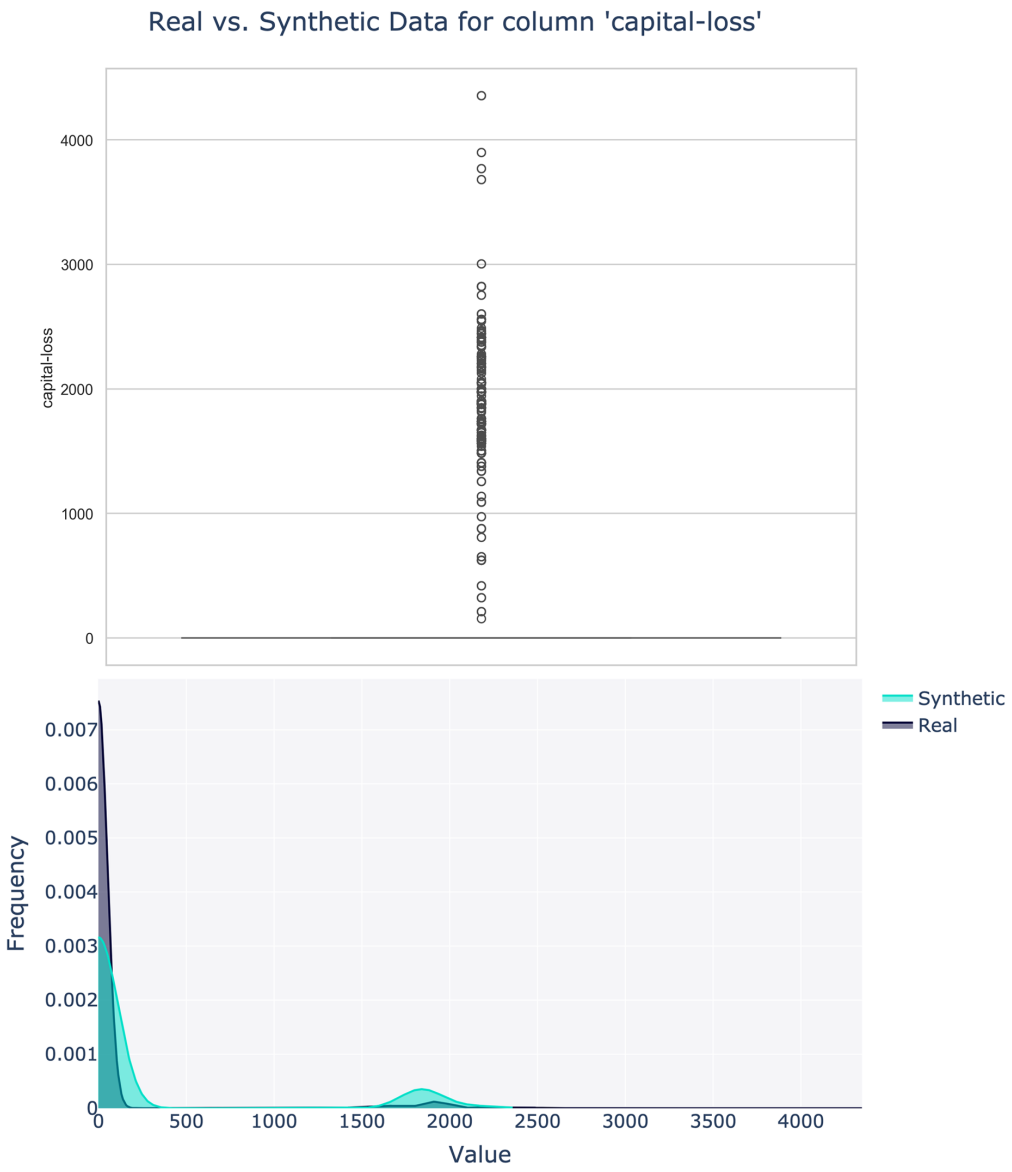

Figure 5.4: Data distribution and comparison of data category coverage for capital-loss column in Adult dataset.

This exact replication of the column distribution, including the less visible mode on the right, shows how well the model understands and can reproduce complex data distributions. The ability to mirror such detailed aspects of the original dataset highlights CA-CTGAN's advanced generative capabilities, ensuring that even the subtler features within a dataset are not overlooked but are instead faithfully represented in the synthetic output.

However, the failure to reflect the significant outliers highlights a restriction in which the produced data, although falling within the overall range, occasionally overlooks extreme outlier values, a phenomenon often resulting from errors in real-world datasets. This variation highlights a critical insight: the Data Coverage metric is modulated by the nature of the dataset itself, particularly the distribution of data and the extremity of its values. Such details show the importance of considering dataset-specific characteristics when evaluating generative models' ability to produce comprehensive and accurate synthetic datasets.

5.1.2 Data Constraint

Within our analysis, the Data Constraint Metric plays a pivotal role in quantifying the fidelity of generated data relative to original datasets, specifically focusing on boundary adherence for continuous columns and category adherence for discrete columns. Our comprehensive evaluation across nine datasets reveals remarkable adherence scores for nearly all discrete columns and the majority of continuous columns as shown in tables 5.4, 5.5 and 5.6. This results underscore the CA-CTGAN model's capability to closely mimic the original data structures.

In figure 5.5 the 'IWS' column from the Beijing PM dataset serves as an exemplary case for continuous columns, achieving a score of 0.92. This high score reflects the model's proficiency in encompassing the range of this skewed distribution, affirming its capacity to generate data that not only adheres to the real data's boundaries but also reflects its underlying distributional characteristics.

Table 5.4: Data Constraint metric: category and range adherence score for Adult, Bank Marketing and Metro PT datasets.

Adult		Bank Marketing		Metro PT	
Column	Score	Column	Score	Column	Score
age	0.998	age	1.000	TP2	1.000
workclass	1.000	job	1.000	TP3	1.000
fnlwgt	0.999	marital	1.000	H1	0.980
education	1.000	education	1.000	DV_pressure	0.999
education-num	1.000	default	1.000	Reservoirs	0.998
marital-status	1.000	housing	1.000	Oil_temperature	1.000
occupation	1.000	loan	1.000	Motor_current	1.000
relationship	1.000	contact	1.000	COMP	1.000
race	1.000	day_of_week	1.000	DV_eletric	1.000
sex	1.000	duration	0.996	Towers	1.000
capital-gain	0.841	campaign	1.000	MPG	1.000
capital-loss	0.896	previous	1.000	LPS	1.000
hours-per-week	1.000	poutcome	1.000	Pressure_switch	1.000
native-country	1.000	y	1.000	Oil_level	1.000
income	1.000			Caudal_impulses	1.000

Table 5.5: Data Constraint metric: category and range adherence score for Air Quality, Bike Sharing and Metro Interstate Traffic datasets.

Air Quality		Bike Sharing		Metro Traffic	
Column	Score	Column	Score	Column	Score
CO(GT)	0.849	holiday	1.000	holiday	1.000
PT08.S1(CO)	1.000	weekday	1.000	temp	1.000
NMHC(GT)	0.909	workingday	1.000	rain_1h	0.851
C6H6(GT)	1.000	weathersit	1.000	snow_1h	0.686
PT08.S2(NMHC)	1.000	temp	1.000	clouds_all	1.000
NOx(GT)	0.876	atemp	1.000	weather_main	1.000
PT08.S3(NOx)	1.000	hum	0.984	weather_description	1.000
NO2(GT)	0.890	windspeed	0.863	traffic_volume	1.000
PT08.S4(NO2)	1.000	casual	0.948		
PT08.S5(O3)	1.000	registered	0.966		
T	1.000	cnt	0.976		
RH	1.000				
AH	0.999				

Table 5.6: Data Constraint metric: category and range adherence score for Apartment Rent, Power Consumption and Beijing PM datasets.

Apartment Rent		Power Consumption		Beijing PM	
Column	Score	Column	Score	Column	Score
bathrooms	0.718	Global_active_power	0.997	pm2.5	1.000
bedrooms	1.000	Global_reactive_power	0.833	DEWP	1.000
fee	1.000	Voltage	1.000	TEMP	1.000
price	1.000	Global_intensity	0.998	PRES	1.000
square_feet	1.000	Sub_metering_1	0.345	cbwd	1.000
cityname	1.000	Sub_metering_2	0.863	Iws	0.923
state	1.000	Sub_metering_3	0.744		

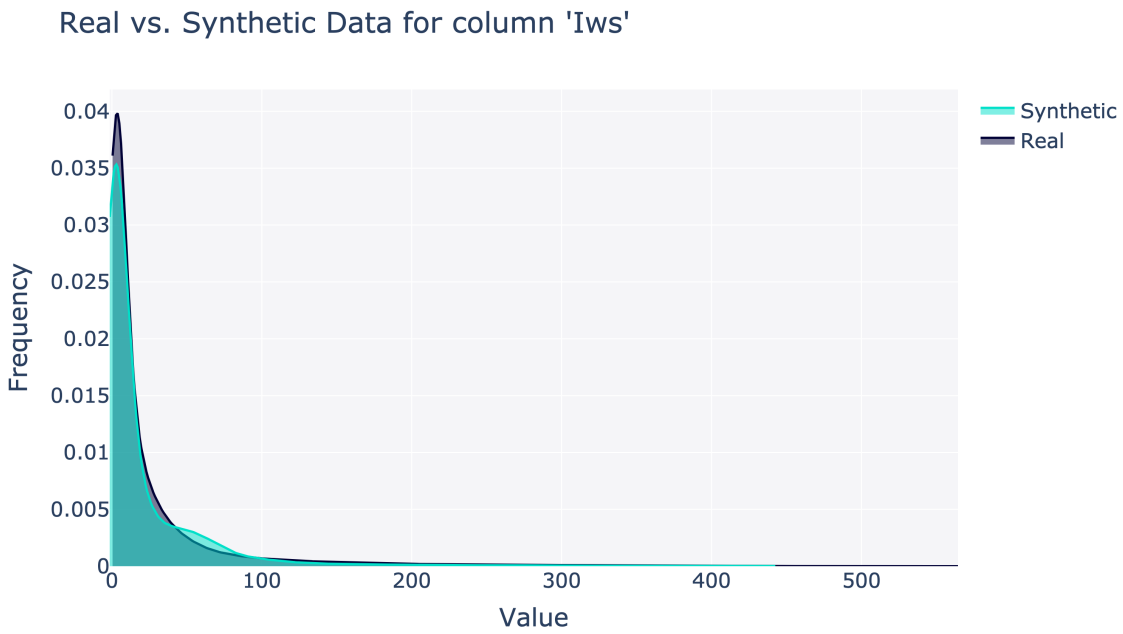

Figure 5.5: Range adherence for IWS column in Beijing PM dataset.

On the other hand, in figure 5.6 the 'NOx(GT)' column within the Air Quality dataset achieved a score of 0.87. Despite slight deviations from the range, the model successfully captured the column's bi-modal distribution, illustrating the nuanced understanding CA-CTGAN possesses in replicating complex data distributions.

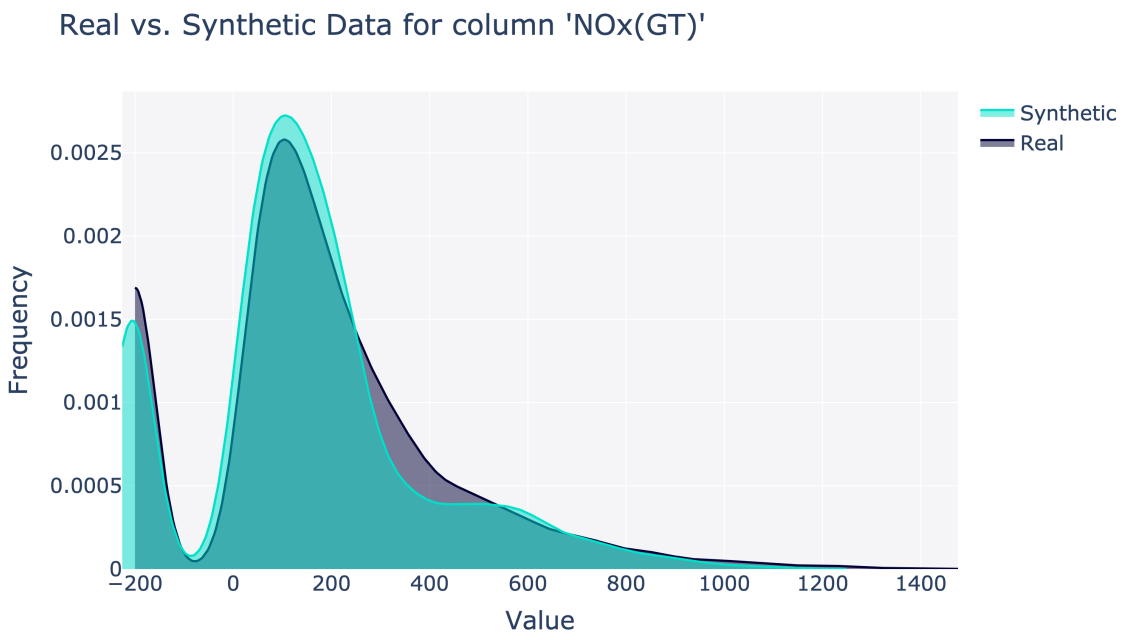

Figure 5.6: Range adherence for NOx(GT) column in Air Quality dataset.

These findings illuminate the strengths of the CA-CTGAN model in maintaining the range of original data and high fidelity to the original data's structural and distributional properties. Through percise adherence to both the discrete and continuous aspects of the datasets, CA-CTGAN demonstrates its effectiveness as a tool for generating synthetic data that retains the essential qualities of its real-world counterparts.

5.1.3 Data Similarity

To quantitatively evaluate the degree of similarity between the marginal distributions of real and generated data, our methodology incorporates two robust metrics: the Kolmogorov-Smirnov (KS) statistic for continuous columns and the Total Variation Distance (TVD) for discrete columns (Section 4.3.6).

Our analysis, as detailed in Tables 5.7, 5.8, and 5.9, along with visual representations in Figures 5.7 to 5.9, demonstrates notably strong conformity between the real and synthetic distributions across both discrete and continuous variables.

Table 5.7: Similarity score of real and generated column for KS and TVD statistic for Adult, Bank Marketing and Metro PT datasets.

Adult			Bank Marketing			Metro PT		
Column	Metric	Score	Column		Score	Column	Metric	Score
age	KS	0.952	age	KS	0.887	TP2	KS	0.760
workclass	TV	0.932	job	TV	0.840	TP3	KS	0.937
fnlwgt	KS	0.882	marital	TV	0.908	H1	KS	0.842
education	TV	0.933	education	TV	0.905	DV_pressure	KS	0.758
education-num	KS	0.950	default	TV	0.921	Reservoirs	KS	0.939
marital-status	TV	0.903	housing	TV	0.908	Oil_temperature	KS	0.921
occupation	TV	0.877	loan	TV	0.987	Motor_current	KS	0.801
relationship	TV	0.903	contact	TV	0.958	COMP	TV	0.819
race	TV	0.969	day_of_week	TV	0.865	DV_eletric	TV	0.851
sex	TV	0.926	duration	KS	0.850	Towers	TV	0.895
capital-gain	KS	0.720	campaign	TV	0.884	MPG	TV	0.814
capital-loss	KS	0.705	previous	KS	0.954	LPS	TV	0.954
hrs-per-week	KS	0.933	poutcome	TV	0.929	Pressure_switch	TV	0.954
native-country	TV	0.906	y	TV	0.947	Oil_level	TV	0.949
income	TV	0.965				Caudal_impulses	TV	0.937

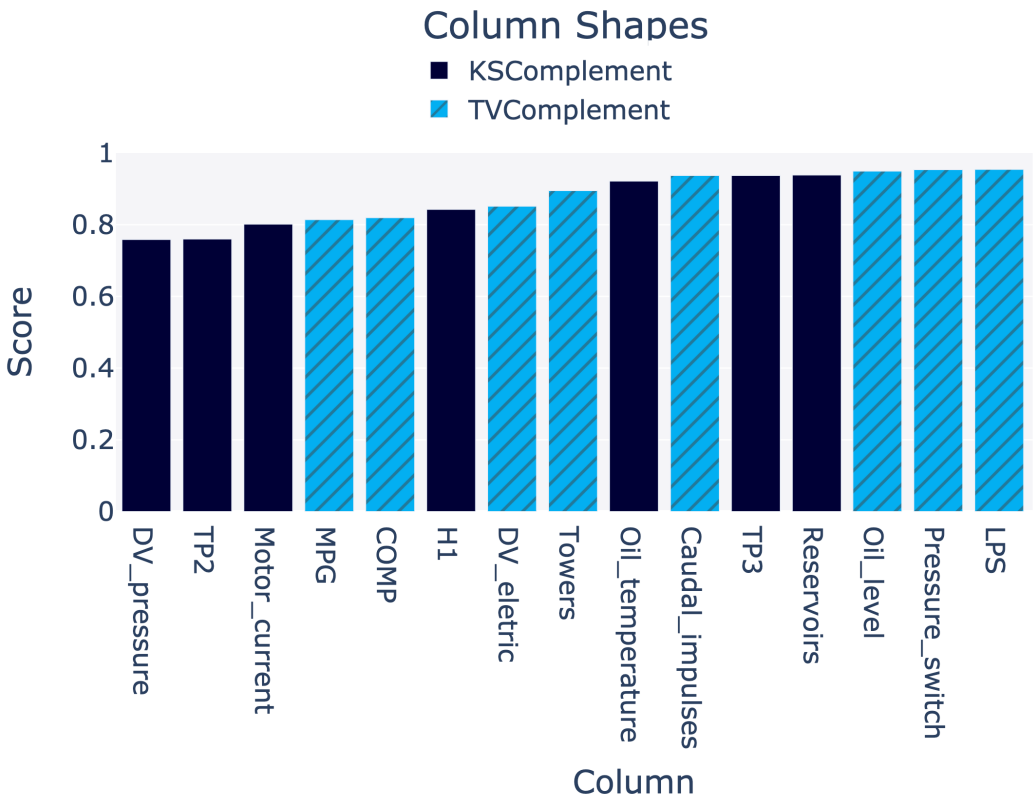

Figure 5.7: Column shape score for Metro PT dataset.

Table 5.8: Similarity score of real and generated column for KS and TVD statistic for Air Quality, Bike Sharing and Metro Interstate Traffic datasets.

Air Quality			Bike Sharing			Metro Traffic		
Column	Metric	Score	Column	Metric	Score	Column	Metric	Score
CO(GT)	KS	0.849	holiday	TV	0.966	holiday	TV	0.993
PT08.S1(CO)	KS	0.912	weekday	TV	0.889	temp	KS	0.954
NMHC(GT)	KS	0.689	workingday	TV	0.947	clouds_all	KS	0.926
C6H6(GT)	KS	0.906	weathersit	TV	0.963	weather_main	TV	0.900
NMHC	KS	0.923	temp	KS	0.950	weather_des	TV	0.821
NOx(GT)	KS	0.876	atemp	KS	0.948	traffic_volume	KS	0.869
PT08.S3(NOx)	KS	0.898	hum	KS	0.886			
NO2(GT)	KS	0.890	windspeed	KS	0.863			
NO2	KS	0.923	casual	KS	0.948			
O3	KS	0.958	registered	KS	0.960			
T	KS	0.950	cnt	KS	0.972			
RH	KS	0.872						
AH	KS	0.829						

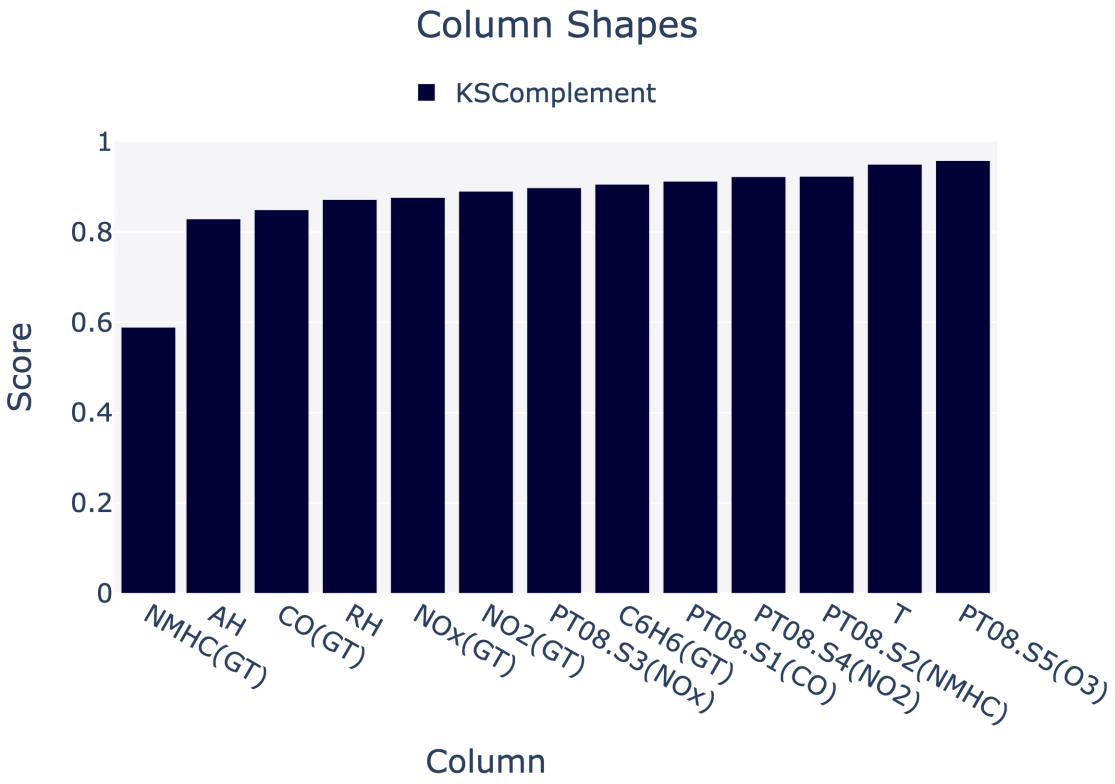

Figure 5.8: Column shape score for Air Quality dataset.

This success is attributed to the precision of the CA-CTGAN model in approximating the underlying distributional characteristics of the original data.

Table 5.9: Similarity score of real and generated column for KS and TVD statistic for Apartment Rent, Power Consumption and Beijing PM datasets.

Apartment Rent			Power Consumption			Beijing PM		
Column	Metric	Score	Column	Metric	Score	Column	Metric	Score
bathrooms	KS	0.716	G_active_power	KS	0.981	pm2.5	KS	0.882
bedrooms	TV	0.911	G_reactive_power	KS	0.833	DEWP	KS	0.963
fee	TV	0.792	Voltage	KS	0.972	TEMP	KS	0.961
price	KS	0.952	Global_intensity	KS	0.953	PRES	KS	0.904
square_feet	KS	0.945	Sub_metering_1	KS	0.645	cbwd	TV	0.916
cityname	TV	0.846	Sub_metering_2	KS	0.736	Iws	KS	0.839
state	TV	0.942	Sub_metering_3	KS	0.844	Is	KS	0.991
						Ir	KS	0.975

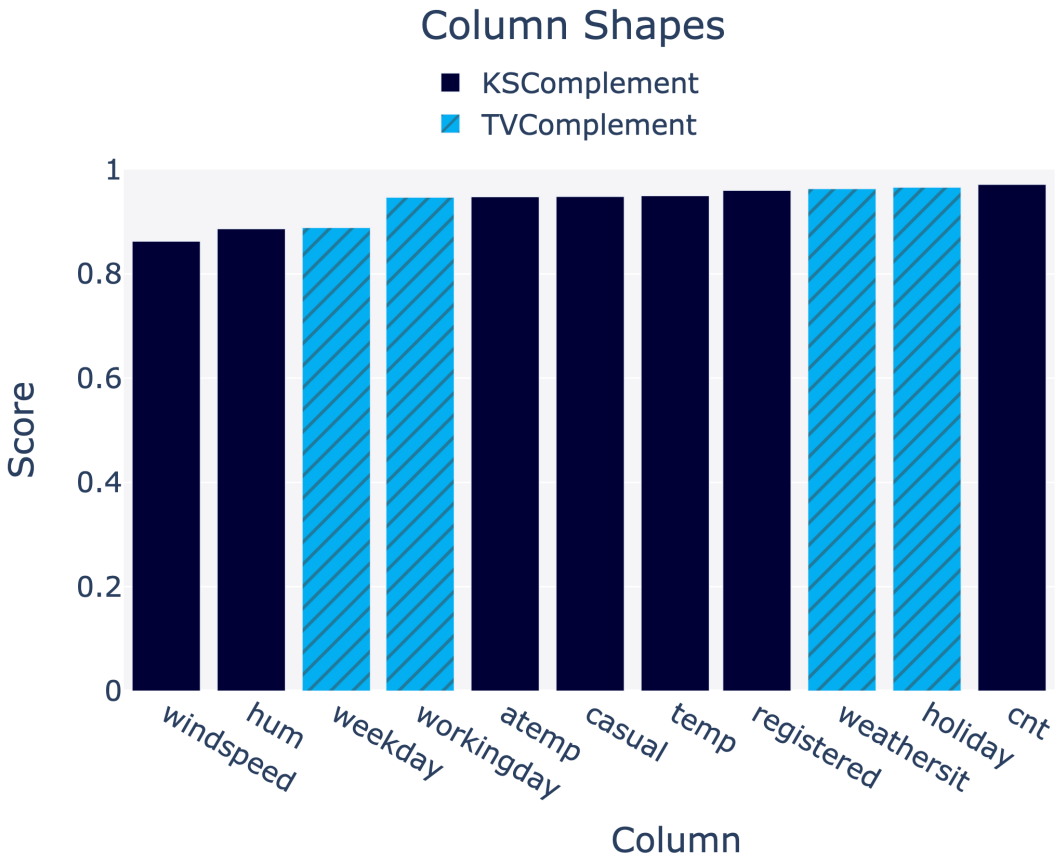

Figure 5.9: Column shape score for Bike Sharing dataset.

The Chi-squared test represents a critical component of our methodology for assessing the fidelity of synthetic data within discrete columns. This statistical test stands out for its rigorous approach to comparing the frequency of categories across real and generated datasets. The results presented in tables 5.10 and 5.11 demonstrate the

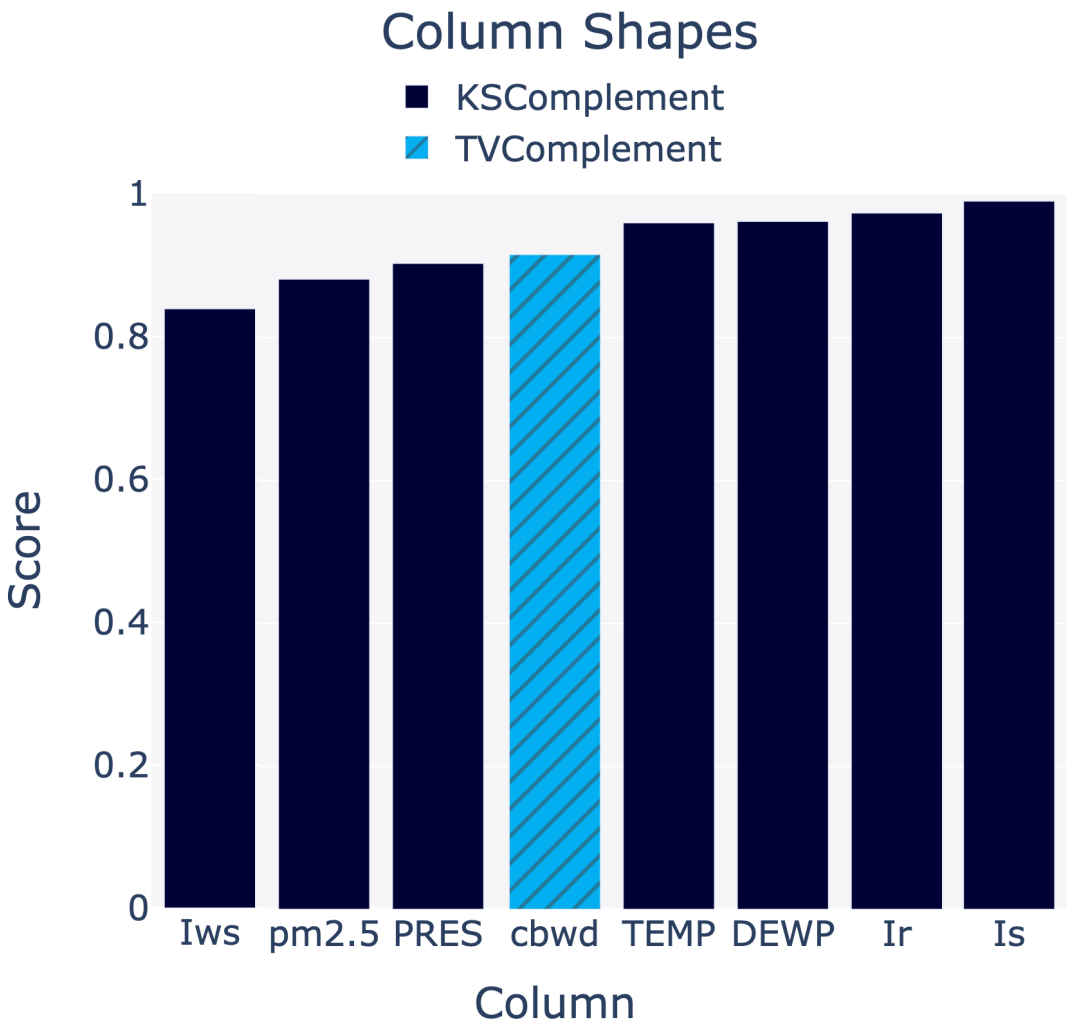

Figure 5.10: Column shape score for Beijing PM dataset.

Table 5.10: Shape similarity for discrete columns using Chi-squared test for Adult, Bank Marketing and Metro PT.

Adult		Bank Marketing		Metro PT	
Column	Score	Column	Score	Column	Score
workclass	1	job	1	COMP	0.825
education	1	marital	0.988	DV_eletric	0.885
marital-status	1	education	1	Towers	0.898
occupation	1	default	0.973	MPG	0.818
relationship	1	housing	0.982	LPS	0.631
race	1	loan	0.999	Pressure_switch	0.814
sex	0.875	contact	0.931	Oil_level	0.924
native-country	1	day_of_week	0.999	Caudal_impulses	0.796
income	0.935	campaign	1		
		poutcome	0.977		
		y	0.868		

Table 5.11: Shape similarity for discrete columns using Chi-squared test for Bike Sharing, Metro Traffic and Beijing PM.

Bike Sharing		Metro Traffic		Apartment Rent		Beijing PM	
Column	Score	Column	Score	Column	Score	Column	Score
holiday	0.839	holiday	1	bedrooms	1	cbwd	0.998
weekday	1	weather_main	1	cityname	1		
workingday	0.909	weather_desc	1	state	1		
weathersit	0.717						

utilization of the Chi-squared metric on columns that are suitable for analysis, specifically columns that do not contain any null values. The Chi-squared test, through its nuanced measure of category frequency alignment, offers compelling evidence of the model's capability to replicate the intricate categorical landscapes of original datasets accurately. This precision ensures that the synthetic data maintains the essential statistical relationships inherent in the real data, affirming its utility for a wide array of analytical applications.

Another fundamental approach to evaluate the similarity between real and synthetic datasets for continuous columns involves statistical analysis using central tendency and variability metrics including mean, median, and standard deviation. These measures provide insights into the central point around which the data values cluster,

Table 5.12: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Metro PT and Bike Sharing datasets.

Metro PT			Bike Sharing		
Column	Score	Metric	Column	Score	Metric
TP2	0.891	mean	temp	0.995	mean
	1.000	median		0.991	median
	0.933	std		0.998	std
TP3	0.994	mean	atemp	1.000	mean
	0.991	median		0.982	median
	0.993	std		0.998	std
H1	0.864	mean	hum	0.989	mean
	0.975	median		1.000	median
	0.915	std		0.979	std
DV_pressure	0.993	mean	windspeed	0.988	mean
	1.000	median		0.966	median
	0.988	std		0.994	std
Reservoirs	0.993	mean	casual	0.988	mean
	0.993	median		0.992	median
	0.991	std		0.985	std
Oil_temperature	0.985	mean	registered	0.997	mean
	0.981	median		0.986	median
	0.990	std		0.976	std
Motor_current	0.925	mean	cnt	0.989	mean
	0.605	median		0.994	median
	0.980	std		0.980	std

as well as the spread of the data points. Tables 5.12, 5.13, 5.14 and 5.15 serve as a comprehensive reference for these statistical metrics, laying out the comparison across all continuous columns within the datasets studied.

By aligning the mean values of the synthetic data with those of the real data, we assess the model's ability to generate data with an equivalent average value. Similarly, by comparing the median, we can determine the model's accuracy in capturing the midpoint of the data distribution, which is particularly informative for skewed distributions. Lastly, the standard deviation comparison reveals how closely the model replicates the range and dispersion of the real data values.

An alignment of these statistical metrics between the real and generated datasets is

Table 5.13: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Power Consumption, Beijing PM, and Adult datasets.

Power Consumption			Beijing PM			Adult		
Column	Score	Metric	Column	Score	Metric	Column	Score	Metric
G_active_pow	0.998	mean	pm2.5	0.992	mean	age	0.981	mean
	0.999	median		0.975	median		0.986	median
	0.995	std		0.998	std		0.988	std
G_reactive_pow	1.000	mean	DEWP	0.999	mean	fnlwgt	0.989	mean
	0.995	median		0.985	median		0.988	median
	0.997	std		0.991	std		0.999	std
Voltage	0.994	mean	TEMP	1.000	mean	education-num	0.996	mean
	0.995	median		0.995	median		1.000	median
	0.996	std		0.998	std		0.976	std
Global_intensity	0.999	mean	PRES	0.993	mean	capital-gain	0.997	mean
	0.997	median		0.981	median		1.000	median
	0.998	std		0.988	std		0.950	std
Sub_metering_1	0.999	mean	Iws	0.993	mean	capital-loss	0.968	mean
	1.000	median		0.996	median		1.000	median
	0.998	std		0.986	std		0.952	std
Sub_metering_2	1.000	mean	Is	0.998	mean	hours-week	0.999	mean
	1.000	median		1.000	median		1.000	median
	0.993	std		0.971	std		0.989	std
Sub_metering_3	0.995	mean	Ir	0.996	mean			
	1.000	median		1.000	median			
	0.999	std		0.977	std			

indicative of the model's capacity to replicate the overall distribution as well as the specific characteristics of the data distribution.

These results substantiate the efficacy of the CA-CTGAN model in producing synthetic data that retains the statistical essence of its real counterparts.
Through the application of similarity metrics, we affirm the model's capacity to replicate the intricate distributional properties of diverse datasets, thereby ensuring that the generated data can serve as a reliable surrogate for the original data in various analytical contexts.

5.1.4 Data Relationship

In assessing the CA-CTGAN model's capability to preserve the semantic relationships and correlations between columns within a dataset, we employed two pivotal metrics: contingency for discrete columns and correlation similarity for continuous

Table 5.14: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Metro Interstate Traffic, Bank Marketing, and Apartment Rent datasets.

Metro Traffic			Bank Marketing			Apartment Rent		
Column	Score	Metric	Column	Score	Metric	Column	Score	Metric
temp	0.998	mean	age	0.992	mean	bathrooms	0.999	mean
	0.995	median		0.975	median		1.000	median
	0.997	std		0.991	std		0.999	std
rain_1h	1.000	mean	duration	0.991	mean	price	0.998	mean
	1.000	median		0.991	median		0.999	median
	0.996	std		0.995	std		0.998	std
snow_1h	1.000	mean	previous	0.994	mean	square_feet	0.996	mean
	1.000	median		1.000	median		0.998	median
	0.984	std		0.999	std		0.994	std
clouds_all	0.991	mean						
	0.760	median						
	0.980	std						
traffic_volume	0.949	mean						
	0.888	median						
	0.987	std						

Table 5.15: Statistical similarity between numerical columns of real data and generated data using mean, median, and standard deviation for Air Quality dataset.

Air Quality								
Column	Score	Metric	Column	Score	Metric	Column	Score	Metric
CO(GT)	0.978	mean	NOx(GT)	0.988	mean	O3	0.997	mean
	1.000	median		0.992	median		0.999	median
	0.982	std		0.989	std		0.969	std
CO	0.996	mean	NOx	0.962	mean	T	0.966	mean
	0.981	median		0.992	median		0.995	median
	0.938	std		0.998	std		0.861	std
NMHC(GT)	0.985	mean	NO2(GT)	0.983	mean	RH	0.955	mean
	0.999	median		0.985	median		0.982	median
	0.975	std		0.994	std		0.876	std
C6H6(GT)	0.968	mean	NO2	0.979	mean	AH	0.961	mean
	0.994	median		0.980	median		0.999	median
	0.874	std		0.966	std		0.809	std
NMHC	0.981	mean						
	0.991	median						
	0.967	std						

columns. Leveraging these methodologies, our analysis reveals that the CA-CTGAN model adeptly generates semantically related samples that not only faithfully maintain the statistical relationships between columns but also adhere to the intricate semantic relationship between columns. This section will present detailed results accompanied by visualizations to elucidate the extent to which our model achieves semantic congruence between the generated and real datasets, highlighting its efficacy in capturing and reproducing the complex, multidimensional structure of real-world data. Through this evaluation, we demonstrate CA-CTGAN's unparalleled ability to generate synthetic data that preserves the essential semantic fabric of the original datasets, thereby affirming its potential as a transformative tool in the realm of data synthesis.

Figure 5.11 illustrates the intricate relationship between pairs of discrete and numerical columns, providing compelling evidence of the CA-CTGAN model's capability. A notable example of this is observed in the age and relationship columns. Here, the model successfully identifies and reflects the semantic linkage that typically associates the 'own-child' category with a younger age range, showcasing its understanding of the data's underlying patterns. Similarly, in examining the relationship between work class and education level, the model reveals a distinct pattern where individuals categorized as 'without pay' or 'never worked' are associated with lower education levels.

Furthermore, Figure 5.12 illustrates the contingency between two distinct columns and the intended label of the dataset, which is perfectly aligned with the actual data. Figures 5.13 and 5.14 depict the comparison between real and synthetic data distributions for the numerical columns in the Air Quality dataset. The scatter plot showcases the model's proficiency in replicating the distribution and relationship inherent between these two variables. It is particularly noteworthy that the model capably echoes the central trend and the majority of the data range present in the real dataset. Despite this fidelity, the plot also highlights a challenge faced by the

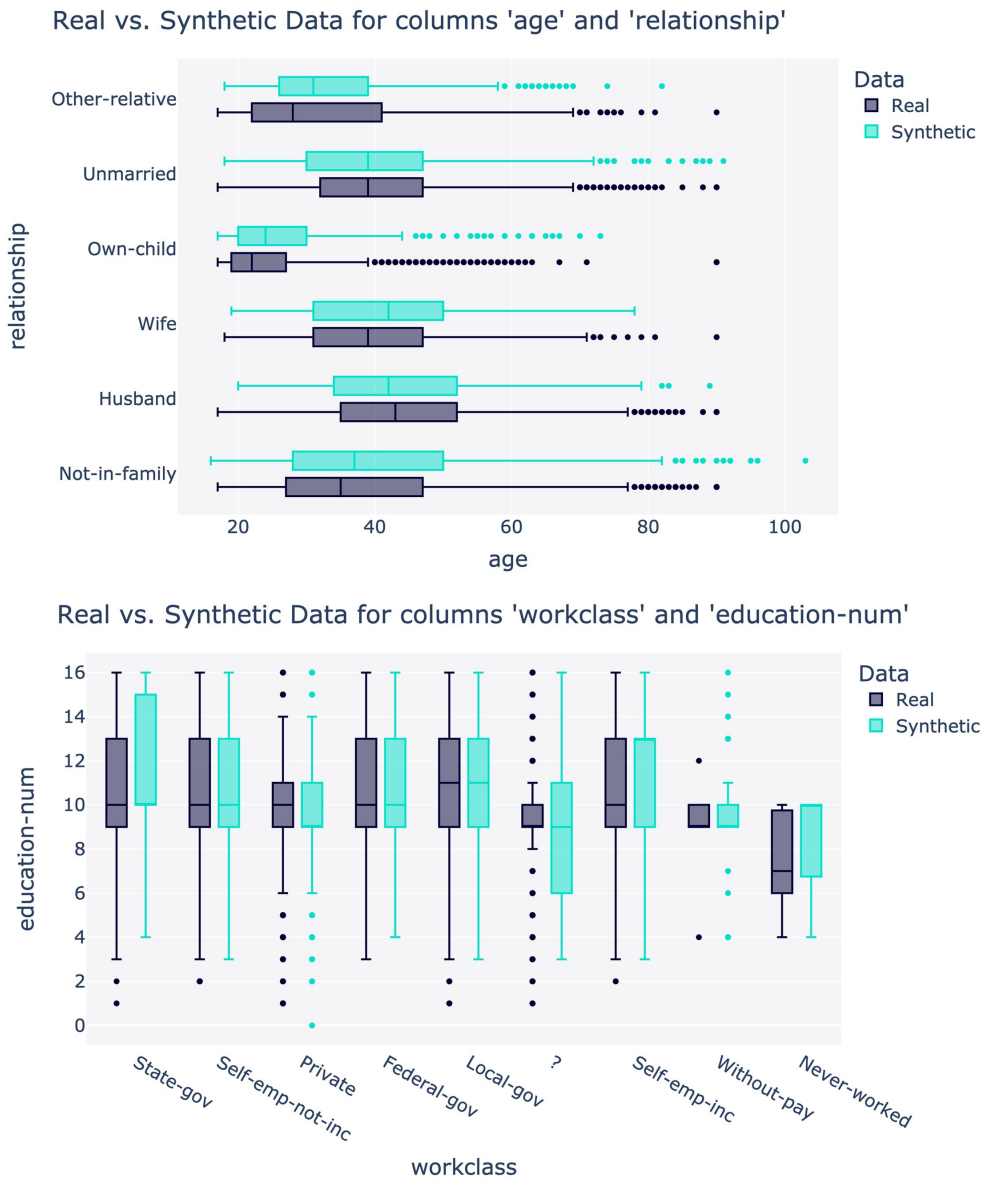

Figure 5.11: Relationship between a pair of discrete and numerical Columns in the Adult dataset.

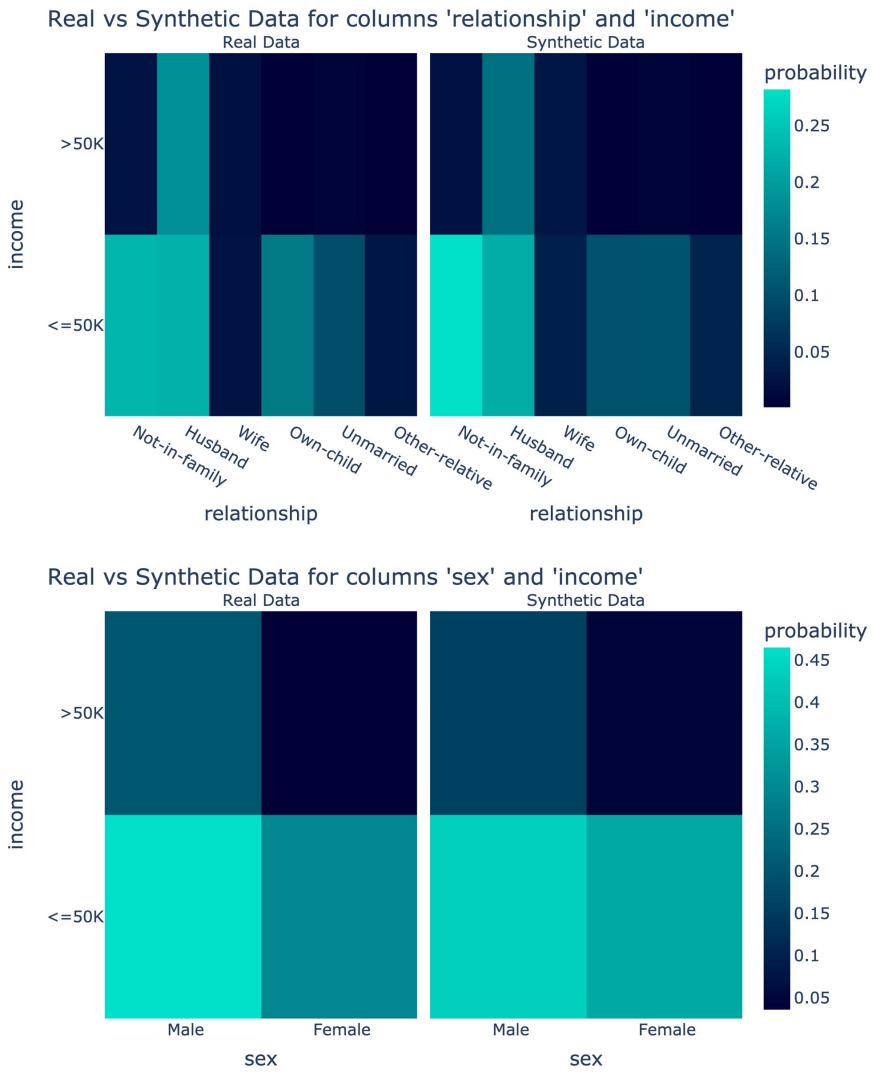

Figure 5.12: Relationship between a pair of discrete columns in the Adult dataset.

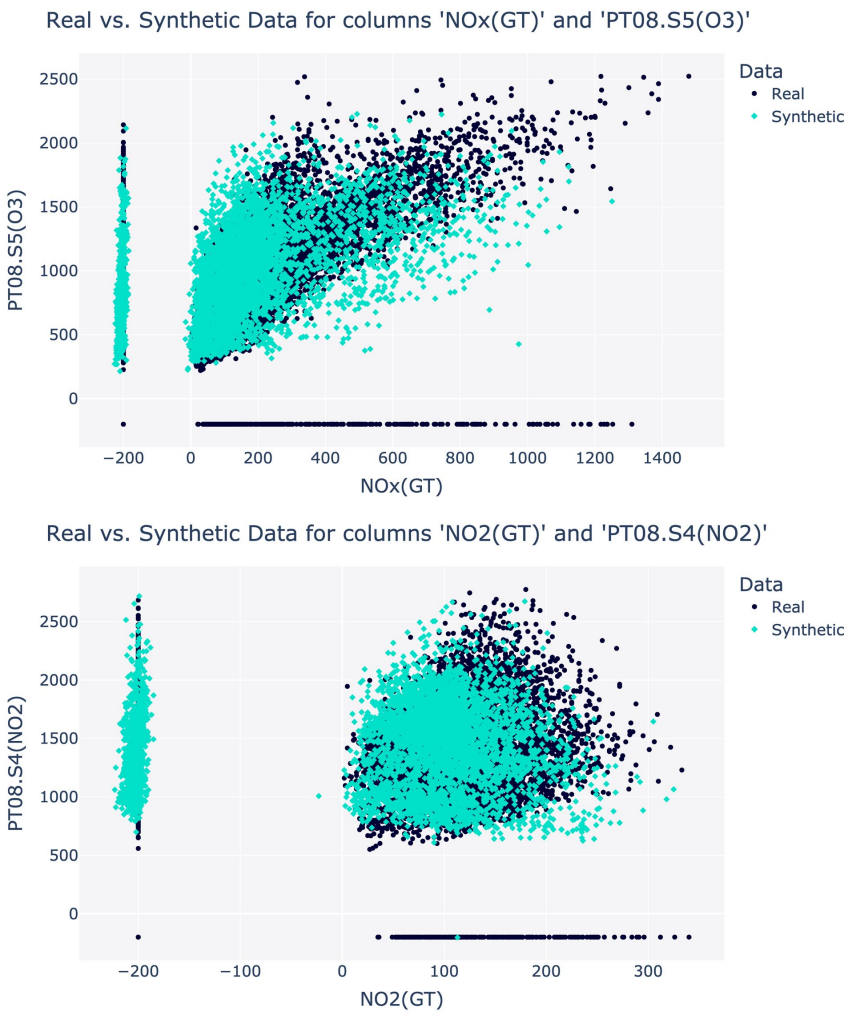

Figure 5.13: Relationship between a pair of continuous columns (NOx/NO2 and PT08) in the Air Quality dataset.

Figure 5.14: Relationship between a pair of continuous columns RH/T and C6H6/CO in the Air Quality dataset.

model: capturing the relationship of strong outliers. These outliers are visually represented as points that stray from the dense cluster of data. While the main body of the synthetic data aligns well with the real data, indicating a robust replication of the distribution, the synthetic data points do not extend to the extreme values occupied by the real outliers.

In Figure 5.15, the Apartment Rent dataset serves as another case study for the CA-CTGAN model's handling of intricate relationships involving categorical and numerical columns. The 'city' column, with its extensive range of categories, is portrayed

alongside a numerical attribute, allowing us to observe how well the model maintains the semantic linkages between location and associated numerical values.

The plot reveals that the CA-CTGAN model effectively captures the relational dynamics between the 'city' categorical variable and corresponding numerical columns. Also, the relationship between 'Square feet' and 'Price', two numerical columns is delineated, testing the model's awareness of subtleties like how smaller apartments can command higher prices depending on the city. Nevertheless, we observe that the model exhibits restraint in generating data points in areas of the distribution where small apartments have disproportionately high prices. Given that such instances were scarce within the real dataset, the model identifies them as outliers and consequently does not reproduce these in the synthetic dataset.

The CA-CTGAN model introduces a powerful feature of generating data conditioned on specific context elements, showcased in Figures 5.16 and 5.17. Figure 5.16 illuminates the relationship between the 'contact' column and the target label 'y' within the Bank Marketing dataset. The visualization encapsulates how the model preserves the contextual relevance between the method of contact and the outcome of the marketing campaign, which is a pivotal aspect for analyses focused on marketing efficiency and strategy optimization. The synthetic data not only upholds the distribution patterns found within the real data but also respects the underlying semantic linkage between the communication method and the customer's response.

In Figure 5.17, the model's performance is further exemplified through its handling of the 'Temp' continuous column and the 'cbwd' categorical column in relation to 'Day' and 'Month', which serve as contextual targets for data generation in this dataset. The visual representation demonstrates that the CA-CTGAN model successfully mirrors the real data's patterns, capturing the temperature's variation with categorical weather directions across different times. Notably, the generated data adheres to realistic and contextually bound value ranges, neither exceeding nor falling short of the

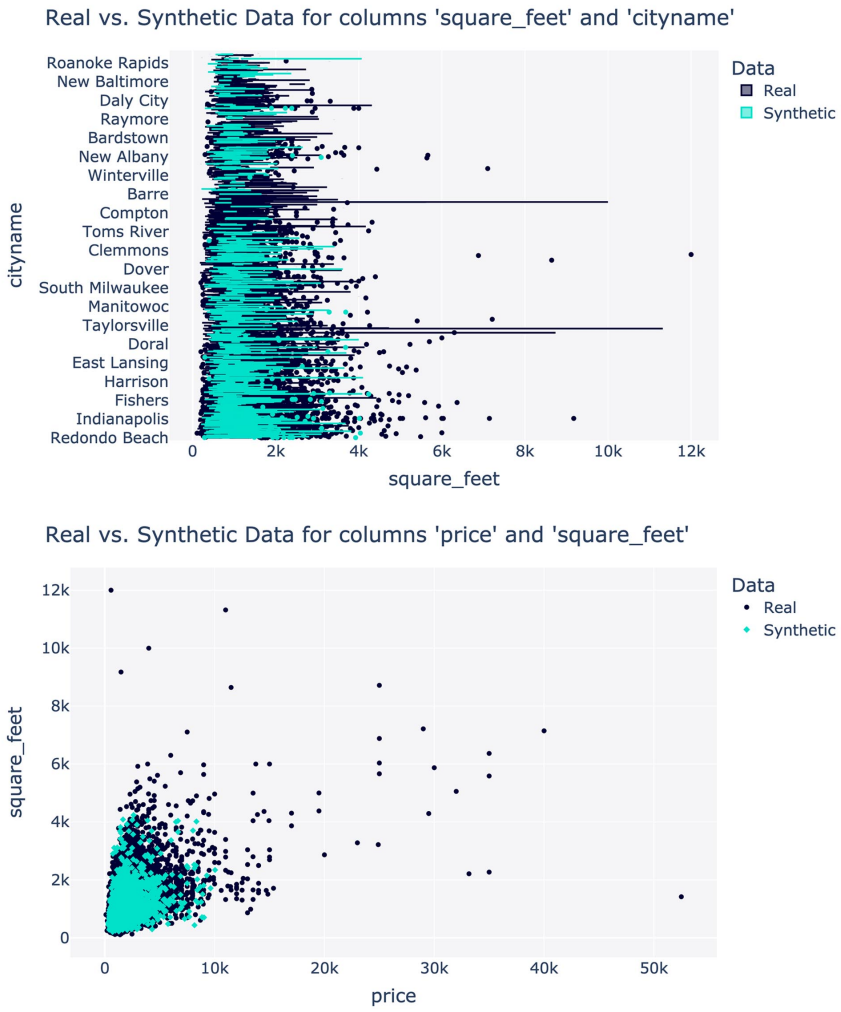

Figure 5.15: Relationship between a pair of continuous (sure feet and price) and a pair of discrete and continuous columns (square feet and cityname) in the apartment dataset.

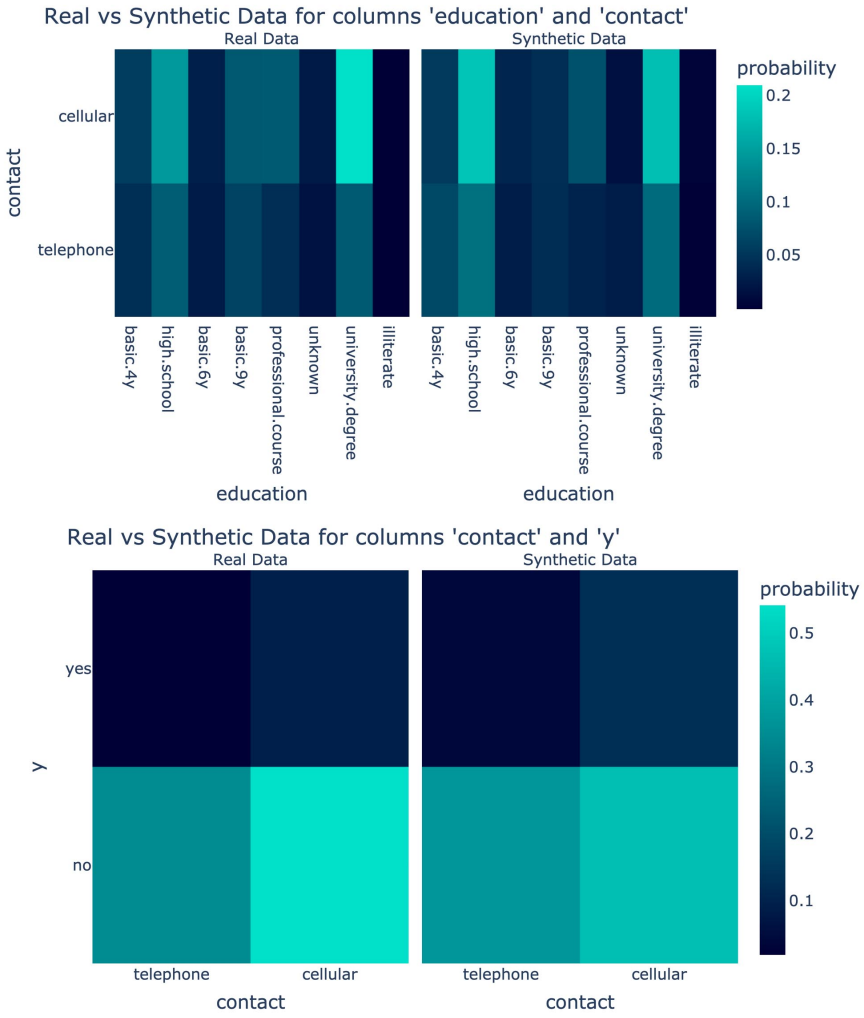

Figure 5.16: Relationship between a pair of discrete columns and target label in the Bank Marketing dataset.

expected limits.

The CA-CTGAN model's adeptness extends to the nuanced realm of handling high-precision numerical data types, including decimal and float values, even down to very small magnitudes. Figure 5.18 shows this capability, presenting a compelling visualization of the model's proficiency in maintaining meaningful relationships between columns with small decimal numbers and other numerical columns. This proficiency is of particular importance in fields where precision is critical, such as financial modeling, scientific computation, and engineering analyses. In these domains, even minor

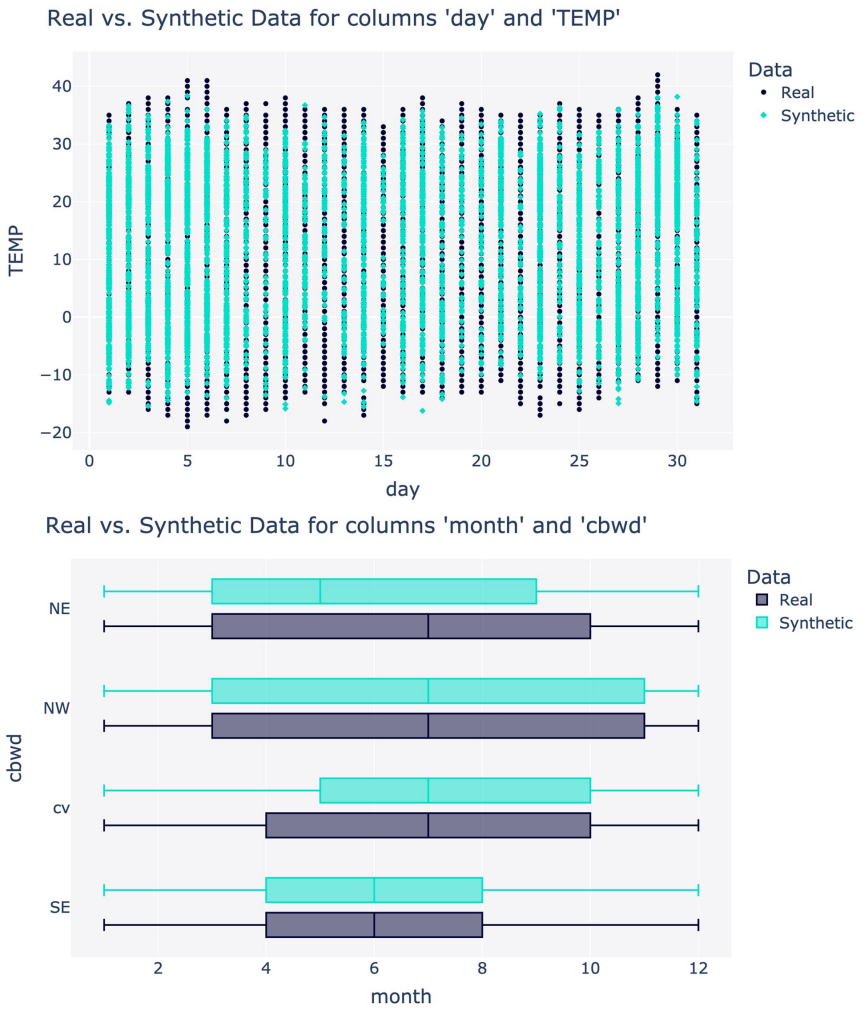

Figure 5.17: Relationship between a pair of continuous and discrete columns and target label in the Beijing PM dataset.

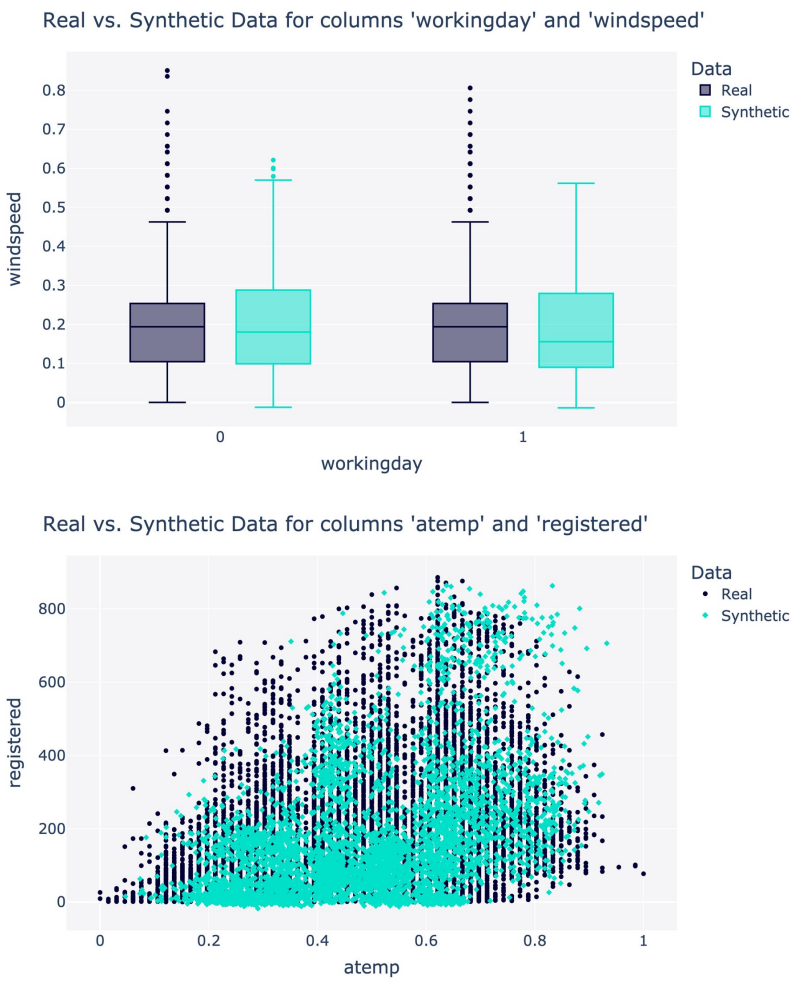

Figure 5.18: Relationship between a pair of continuous and discrete columns and a high-precision small number in Bike Sharing dataset.

discrepancies in decimal points can lead to significantly different outcomes. The ability of CA-CTGAN to accurately generate data with high decimal precision ensures that the synthetic data is not only statistically representative but also practically applicable in scenarios where precision cannot be compromised.

5.1.5 Machine Learning Performance

Table 5.16 encapsulates the performance of two distinct but interrelated metrics (ML Detection and ML Efficiency) which provide a holistic view of our synthetic data

utility in machine learning contexts.

ML Detection serves as a measure of the indistinguishability between real and synthetic data. To measure this, we employ a machine learning classifier, specifically, Logistic Regression. This classifier is tasked with identifying whether each row of data is real or synthetic. The classifier's effectiveness at this task is quantified using the Receiver Operating Characteristic (ROC) Area Under the Curve (AUC) score. The effectiveness of this differentiation is quantified using the average ROC AUC score, with a score closer to 1 indicating that the classifier is unable to distinguish between the two, thus suggesting a higher level of realism in the synthetic data.

This outcome implies that the synthetic data closely resembles the real data, effectively 'deceiving' the classifier into making errors or being uncertain in its classification tasks. Thus, a higher score here denotes a greater degree of realism in the synthetic data, as it mirrors the real data well enough to confuse the classification model.

This metric is pivotal as it directly assesses the indistinguishability of the synthetic data from the real data, which is a primary goal in the creation of synthetic datasets. The results of this metric, presented in our tables, provide a clear indication of how similar the generated data is to the original datasets, underscoring the effectiveness of our synthetic data generation methods.

Conversely, ML Efficiency focuses on the synthetic data efficacy in predictive tasks. Specifically, it evaluates the performance of a synthetic dataset when used to train a machine learning model, with the AdaBoost classifier being employed for this metric. The table records both Accuracy and F1-Score, which collectively reflect the trained model's precision and robustness when making predictions on a real dataset.

The scores in the table reflect a satisfactory balance between both metrics for all datasets considered.

Therefore, the ML Detection scores suggest that the synthetic data produced by CA-CTGAN is closely aligned with real data to the extent that it poses a challenge

Table 5.16: ML Detection (Logistic Regression) and ML Efficiency (AdaBoost) score for all datasets.

Dataset	Detection Score	Accuracy	F1
Adult	0.752	0.897	0.702
Bank Marketing	0.634	0.791	0.621
Metro PT	0.859	0.862	0.67
Air Quality	0.599	0.732	0.452
Bike Sharing	0.901	0.856	0.736
Metro Traffic	0.718	0.799	0.549
Apartment Rent	0.723	0.766	0.576
Power Consumption	0.906	0.883	0.623
Beijing PM	0.859	0.816	0.686

for machine learning models to differentiate. On the other hand, the ML Efficiency metrics, as evidenced by Accuracy and F1-Scores, demonstrate that synthetic data maintains a high level of practicality, enabling machine learning models to achieve reasonable performance on real-world tasks. Together, these metrics attest to the quality and applicability of the synthetic data generated by the CA-CTGAN model.

5.2 Comparative Study

In this section we present an empirical analysis that benchmarks the performance of the Context-Aware Conditional Tabular Generative Adversarial Network (CA-CTGAN) against three established GAN-based models for synthetic tabular data generation: CTGAN, DATGAN, and CTAB-GAN. This comparison utilizes an array of metrics explained in section 4.3.6, each providing a unique lens through which to assess the capabilities of the generative models.

To facilitate a fair and comprehensive comparison, we have calculated the average scores across all columns within our datasets. This approach accounts for potential limitations exhibited by some models, particularly when handling data types with high precision, such as float or tiny values, as well as their varying capacities to manage null values. By employing the average score method, we can avoid the potential bias that may result from individual columns, which could unevenly impact the over-

Table 5.17: A comparative analysis of CA-CTAGN and baseline methods was conducted on five datasets.

Dataset	Model	Coverage	Adherence	KS/TVD	Statistic	Contingency
Adult	CTGAN	0.860	0.979	0.888	0.989	0.822
	CTABGAN	0.872	0.959	0.864	**0.991**	0.852
	DATGAN	0.826	0.946	0.850	0.975	0.797
	CA-CTGAN	**0.903**	**0.982**	**0.898**	0.987	**0.860**
Bank Marketing	CTGAN	**0.954**	0.963	0.907	0.993	0.833
	CTABGAN	0.864	0.968	0.885	**0.995**	0.830
	DATGAN	0.790	0.892	0.810	0.960	0.763
	CA-CTGAN	0.938	**1.000**	**0.910**	0.992	**0.857**
Metro PT	CTGAN	0.858	0.975	**0.887**	0.955	0.778
	CTABGAN	0.859	0.966	0.855	**0.971**	0.742
	DATGAN	0.669	0.752	0.871	0.931	0.427
	CA-CTGAN	**0.894**	**0.998**	0.876	**0.971**	**0.825**
Bike Sharing	CTGAN	0.872	0.965	0.862	**0.990**	0.818
	CTABGAN	0.856	0.954	0.850	0.973	0.817
	DATGAN	0.888	0.954	0.885	0.968	0.705
	CA-CTGAN	**0.943**	**0.976**	**0.936**	0.989	**0.898**
Apartment Rent	CTGAN	**0.861**	0.956	0.874	0.953	**0.791**
	CTABGAN	0.778	0.954	0.817	**0.972**	0.734
	DATGAN	0.636	0.879	0.775	0.915	0.690
	CA-CTGAN	0.860	**0.960**	**0.885**	0.958	0.775

all evaluation due to these constraints.

Table 5.17 offer a side-by-side comparison, making it evident that CA-CTGAN not only outperforms other methods, but also significantly surpasses then in performance. This improvement is consistently observed across all metrics, confirming that the enhancements integrated into CA-CTGAN effectively address the shortcomings encountered in CTGAN and other peer models.

The results indicate that the CA-CTGAN framework, which is specifically designed to be aware of the context, demonstrates a notable capability to generate data that is statistically consistent with the original data and maintains its contextual integrity. This characteristic confers a significant advantage over the alternative models.

Figure 5.19 and 5.20 showcase the distribution frequencies of real and generated data for the 'education' column in the Adult dataset and 'LPS' column in the MetroPT dataset, respectively. In (a) CTGAN tends to overrepresent minor categories, diverging from the true data distribution seen in the Adult dataset. Conversely, (b)

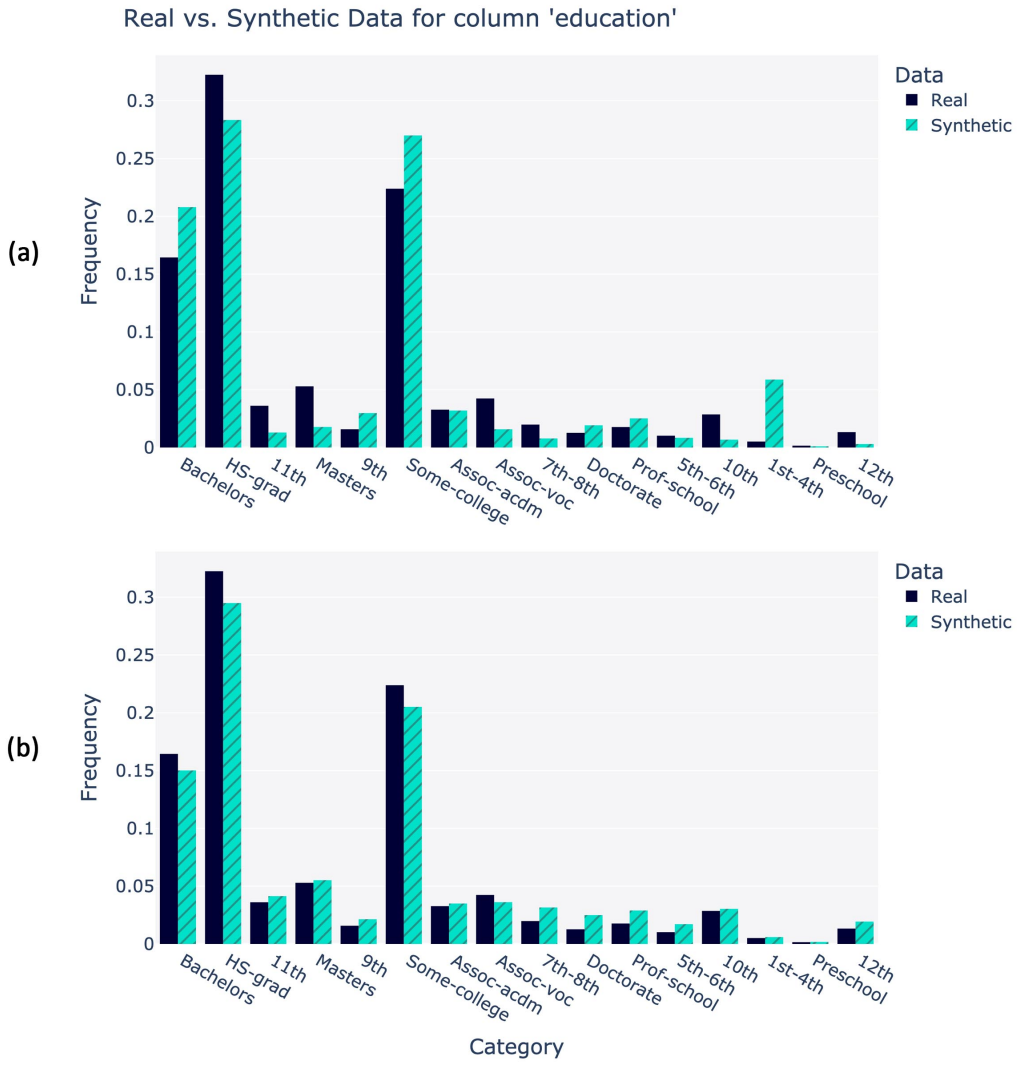

Figure 5.19: Frequency distribution comparison for 'Education' in Adult Dataset. (a) CTAGN (b) CA-CTGAN.

illustrates CA-CTGAN's ability to generate a distribution that closely mirrors the actual frequency of categories, confirming its enhanced capability for realistic data synthesis.

Figure 5.21 presents a heatmap comparison illustrating the relationship between the number of bedrooms and 'fee' in the Apartment dataset. The real data heatmap in (a) is set against those generated by CTABGAN and CA-CTGAN in (b) and (c), respectively. CA-CTGAN not only replicates the distribution patterns but also retains the contextual nuances of the dataset, demonstrating its superior performance

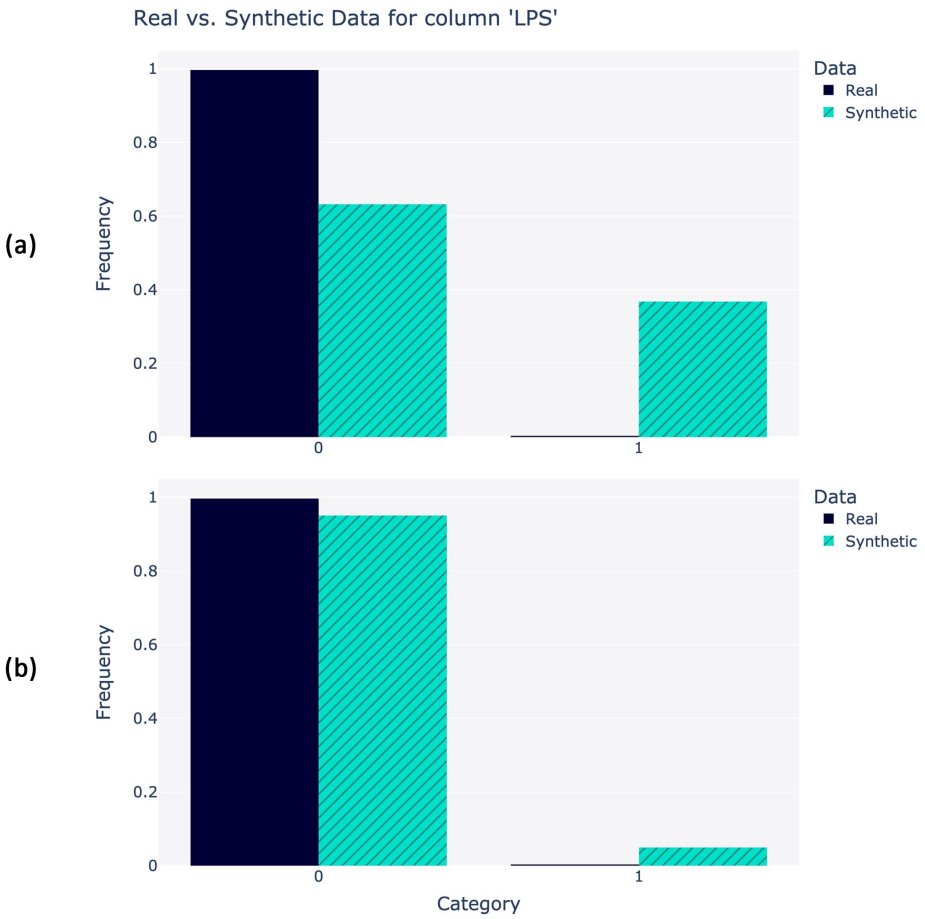

Figure 5.20: Frequency distribution comparison for 'LPS' in MetroPT dataset. (a) CTAGN (b) CA-CTGAN

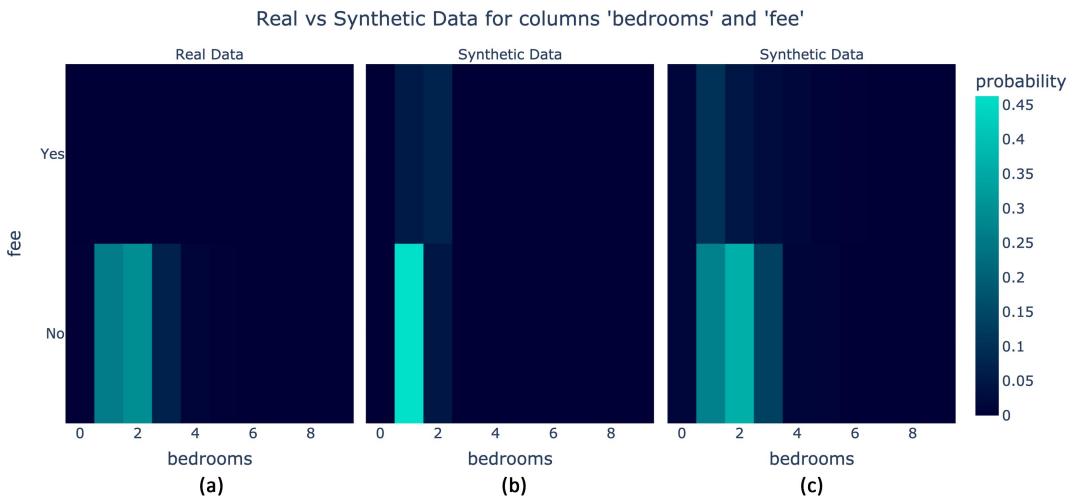

Figure 5.21: Heatmap of bedrooms vs. fee relationship in Apartment dataset: A real and generated data comparison. (a) Real Data (b) CTABGAN (c) CA-CTGAN

in context-aware data generation.

In Figure 5.22, the relationship between 'price' and 'square_feet' in the Apartment Rent dataset is examined through heatmaps of CTABGAN and CA-CTGAN against the real dataset. While both models capture the correlation between the columns, CA-CTGAN in (b) offers a more expansive coverage of the distribution range, showcasing its robustness in modeling a more nuanced and realistic dataset.

The proficiency of CA-CTGAN in capturing complex data distributions is further highlighted in Figures 5.23 and 5.24. Comparing the 'previous' column in the Bank Marketing dataset and 'atemp' column in the Bike Sharing dataset, CA-CTGAN in (b) accurately reflects the multimodal distribution of the real data. DATGAN, in (a), however, struggles with mode representation, particularly in skewed distributions, underlining the advanced capabilities of CA-CTGAN in data generation tasks.

Figure 5.25 offers a heatmap analysis of the 'LPS' and 'COMP' columns in the Metro PT dataset. Both CTABGAN and CA-CTGAN heatmaps are compared to discern the model's capacity to preserve data relationships. CA-CTGAN in (c) demonstrates a discernible advantage, more effectively capturing minor category relationships, thereby reinforcing its aptitude for detailed and contextually aware synthetic

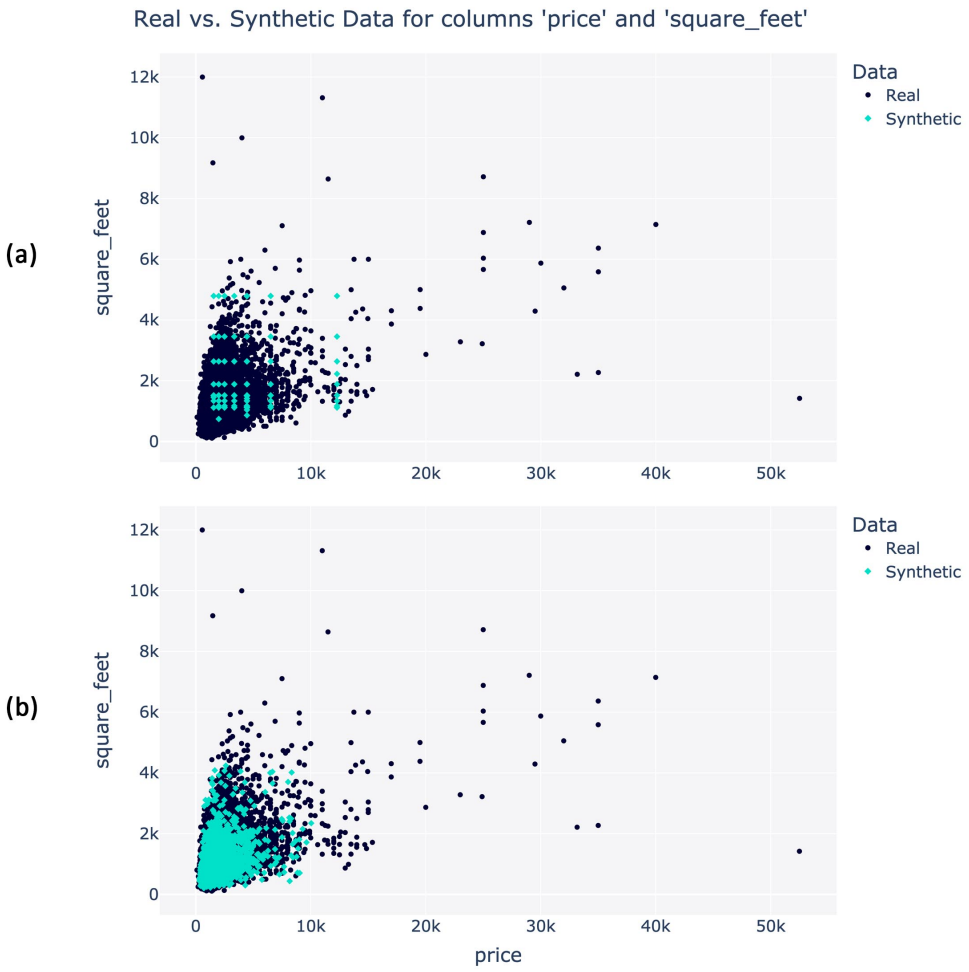

Figure 5.22: Comparative analysis of 'Price' and 'Square Feet' Relationship in Apartment Rent dataset using pair plot. (a) CTABGAN (b) CA-CTGAN

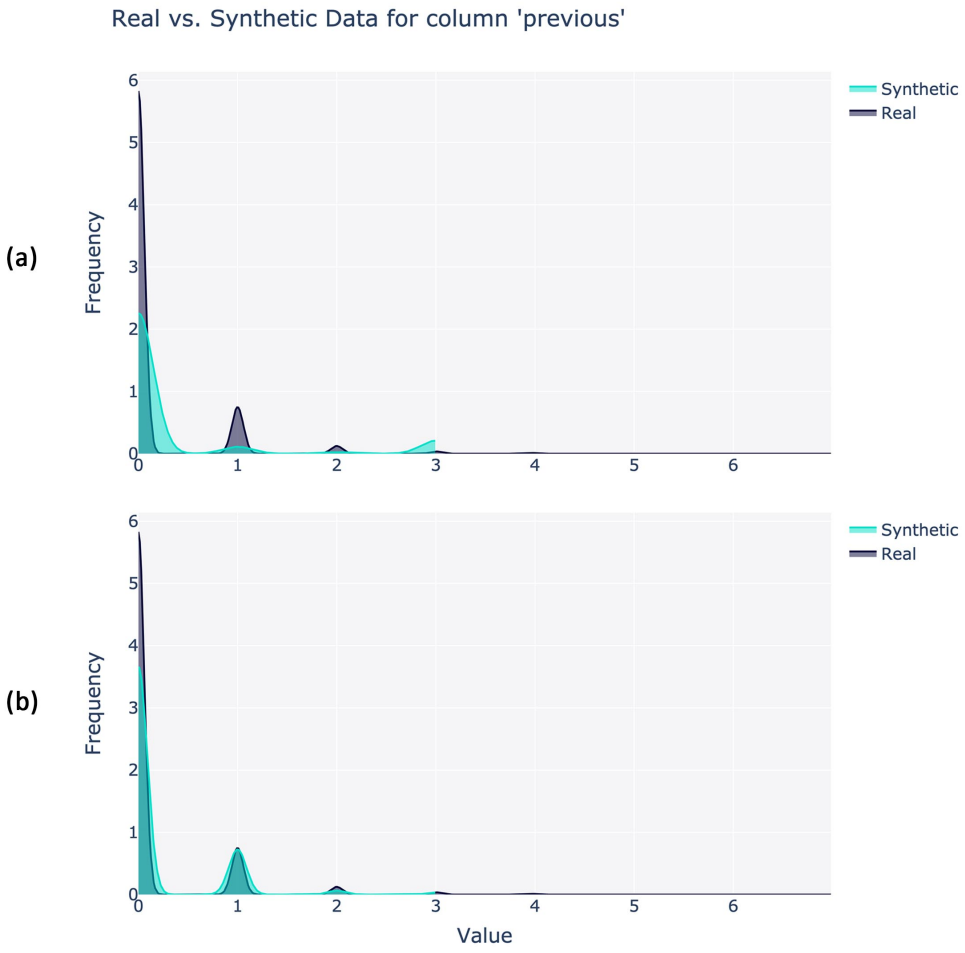

Figure 5.23: Distribution analysis of 'previous' column in Bank Marketing dataset. (a) DATGAN (b) CA-CTGAN

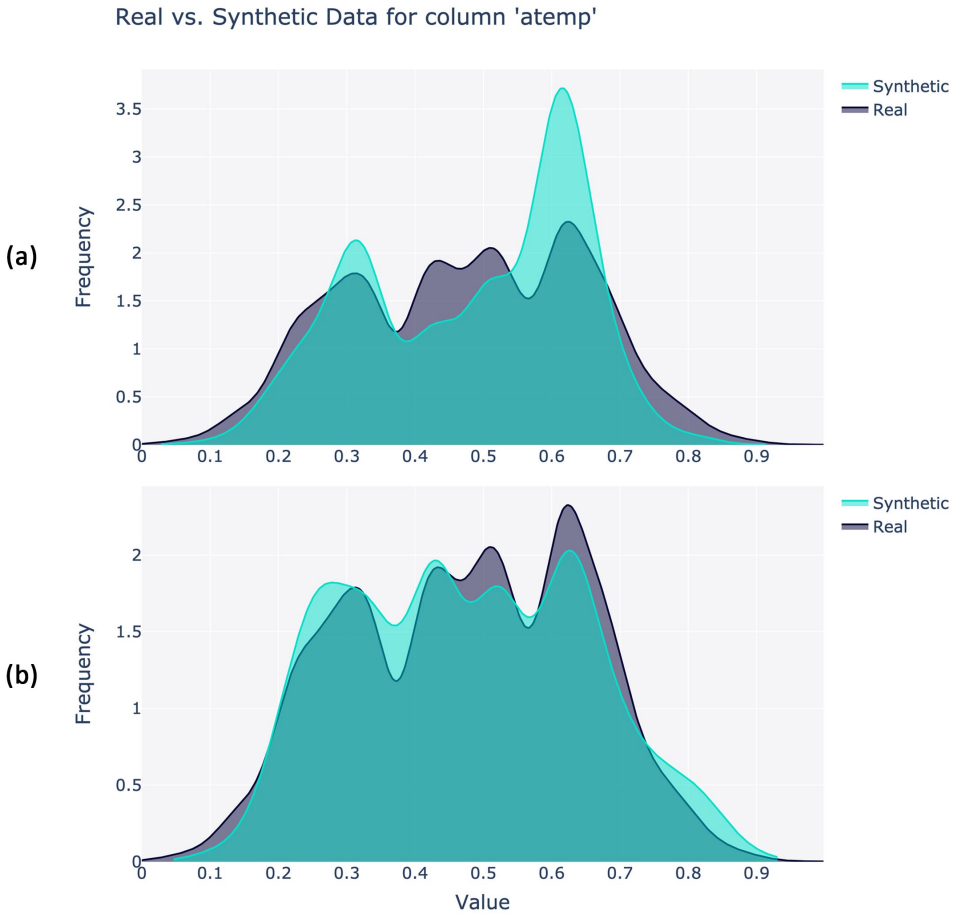

Figure 5.24: Distribution analysis of 'Atemp' in Bike Sharing dataset. (a) DATGAN (b) CA-CTGAN

data generation.

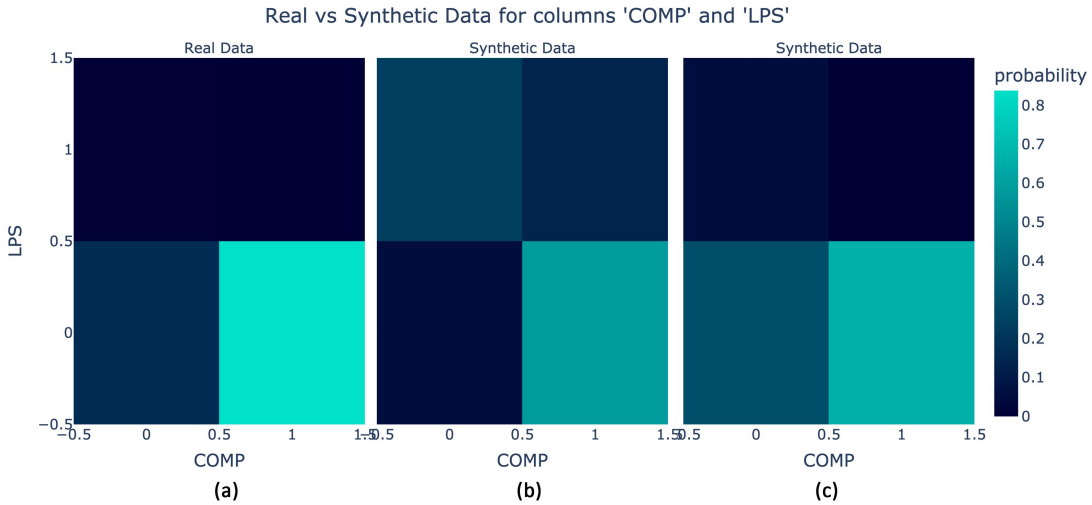

Figure 5.25: Heatmap visualization of 'LPS' and 'COMP' Relationship in Metro PT dataset. (a) Real Data (b) CTABGAN (c) CA-CTGAN

In addition, we introduce an approach integrating a pre-trained Contractive Autoencoder (CAE) for generating semantically-consistent noise. Conducting experiments on datasets highlights how the noise quality in CA-CTGAN is essential for enhancing the data it generates and stabilizing and speeding up convergence [4]. GANs often face a challenge known as the non-convergence problem during training [57]. This issue arises when the generator and discriminator networks fail to reach a stable equilibrium, leading to oscillations in training dynamics.

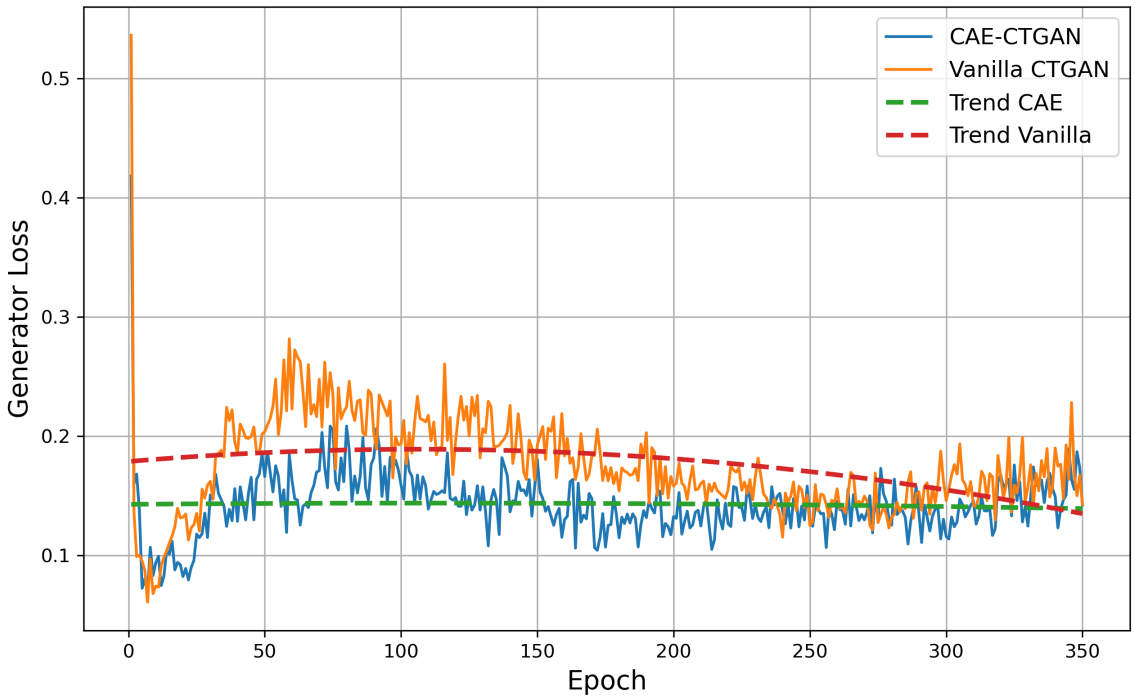

Figure 5.26: Generator loss over the epochs for both training approaches (with and without CAE) - Adult dataset [4].

In such cases, the generator might produce nonsensical outputs or fail to capture the diversity of the training data [57]. Addressing non-convergence requires careful tuning of hyperparameters, novel training strategies, or modifications to the GAN architecture. Figure 5.26 offers a compelling view of the comparative convergence rates of the CTAGN model integrated with CAE and the CTGAN base model. Notably, the integrated model showcases a faster convergence, reaching a stabilized performance near the 150th epoch. In contrast, the CTGAN demands more than double the number of epochs (close to 350) to attain a similar performance level, and also, the trend lines indicate a lengthier journey to stabilization. This accelerated convergence of the CA-CTGAN underscores the effectiveness of introducing noise generated by CAE in the GAN training process.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

The Context-Aware Conditional Tabular Generative Adversarial Network (CA-CTGAN) is a notable breakthrough in the field of synthetic data creation. CA-CTGAN is specifically designed to overcome the constraints associated with conventional approaches. It distinguishes itself by integrating context-specificity, hence enabling the generation of datasets that are customized to suit various experimental environments. This study has shown that CA-CTGAN is highly effective in creating synthetic datasets that accurately replicate the complex features of real-world data and strictly adhere to certain research contexts.

The challenge of generating realistic synthetic data that accurately reflects the intricacies of experimental data has been a persistent challenge. Existing methods often lack flexibility or result in synthetic datasets failing to capture specific requirements or subtle details present in real-world data. CA-CTGAN directly addresses this issue through its multifaceted approach, which includes transfer learning, an innovative auxiliary classifier, and entity embedding techniques, resulting in a contextually integrated framework that allows for precise control throughout the generative process.

The meticulous evaluation of CA-CTGAN reveals a series of impressive results, solidifying its superior capabilities. The framework achieved exceptional data coverage, indicating exceptional reproduction of even subtle distributional patterns within the original dataset. CA-CTGAN's strong performance on Kolmogorov-Smirnov, Total Variation Distance, Chi-squared tests, and standard statistical measures highlights its ability to replicate complex underlying distributions.

In terms of preserving semantic relationships, results demonstrate CA-CTGAN's success in maintaining strong statistical correlations and semantic relationships across

columns in the synthetic data. This ensures the synthetic samples retain the contextual essence of the real data. Furthermore, in the context of indistinguishability and ML Efficiency, the model produced synthetic data closely resembling genuine data, proving challenging to discern by machine learning detection methods. Moreover, the synthetic datasets trained robust ML models effectively, mirroring real-world predictive performance. This showcases the potential for CA-CTGAN to be used across research pipelines.

In addition, CA-CTGAN consistently demonstrated superior performance compared to established GAN-based methods (CTGAN, DATGAN, CTAB-GAN). This underscores its significant contribution to the field of synthetic data generation.

Significance & Implications:

The implications of this research are far-reaching and transformative. CA-CTGAN can streamline experimental design and reduce costs. The potential ability to decrease or even replace some real-world experiments with synthetic data without compromising the validity of results has profound implications on resources, time, and the feasibility of studies. In fields such as healthcare, where data privacy is paramount, CA-CTGAN enables researchers to work with realistic synthetic data, mitigating ethical challenges that arise when using sensitive patient information. Moreover, across numerous domains, having access to contextually accurate synthetic data can enhance data analysis capabilities and foster more informed decision-making processes.

Limitations and Future Directions:

Acknowledging limitations is a hallmark of rigorous research. CA-CTGAN, while exceptionally capable, does have certain aspects to address in future development, including:

- **Outlier Handling:** Although extreme outliers are frequently observed as a consequence of errors in real-world datasets, it is important to improve their handling in order to accurately represent the entire distribution, including those

values.

- **Classifier Refinement:** Exploring alternative classifier architectures could potentially boost performance and efficiency.
- **Semantic Control via LLMs:** The integration of Large Language Models could further enhance nuanced semantic and contextual representation in the generated data.
- **Time-series Support:** Expanding the CA-CTGAN architecture to handle time-series data, opening up broad applications in dynamic process analysis and modeling.

In conclusion, the CA-CTGAN framework presents a pivotal step towards data democratization and the advancement of rigorous, ethical, and cost-effective research methodologies. Its ability to generate high-fidelity, contextually nuanced synthetic data marks a significant contribution to scientific fields. As the research around CA-CTGAN continues to evolve, it has the potential to redefine the way experiments are conducted, unlocking new possibilities and accelerating the pace of discovery.

REFERENCES

[1] Z. Zhao, A. Kunar, R. Birke, and L. Y. Chen, "Ctab-gan: Effective table data synthesizing," in *Asian Conference on Machine Learning*, pp. 97–112, PMLR, 2021.

[2] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," *Advances in neural information processing systems*, vol. 31, 2018.

[3] DataCebo, Inc., *Synthetic Data Metrics*, 9 2022. v0.7.0.

[4] H. Fallahian, M. Dorodchi, and K. Kreth, "Beyond noise: Incorporating pre-trained contractive autoencoders for enhanced gan-based tabular data creation," in *2024 7th International Conference on Information and Computer Technologies (ICICT)*, p. to appear, 2024.

[5] 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.

[6] M. Fallahian, M. Dorodchi, and K. Kreth, "Gan-based tabular data generator for constructing synopsis in approximate query processing: Challenges and solutions," *Machine Learning and Knowledge Extraction*, vol. 6, no. 1, pp. 171–198, 2024.

[7] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

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

[9] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan, "Do deep generative models know what they don't know?," in *International Conference on Learning Representations*, 2018.

[10] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] G. Harshvardhan, M. K. Gourisaria, M. Pandey, and S. S. Rautaray, "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review*, vol. 38, p. 100285, 2020.

[13] Z. Wang, Q. She, and T. E. Ward, "Generative adversarial networks in computer vision: A survey and taxonomy," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[14] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pp. 52–59, Springer, 2011.

[15] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[16] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[17] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[18] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive autoencoders: Explicit invariance during feature extraction," in *Proceedings of the 28th international conference on international conference on machine learning*, pp. 833–840, 2011.

[19] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[20] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[21] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[22] C. Sun, X. Qiu, Y. Xu, and X. Huang, "How to fine-tune bert for text classification?," in *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pp. 194–206, Springer, 2019.

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

[24] U. Khurana and S. Galhotra, "Semantic annotation for tabular data," *arXiv preprint arXiv:2012.08594*, 2020.

[25] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Machine learning for healthcare conference*, pp. 286–305, PMLR, 2017.

[26] A. Mottini, A. Lheritier, and R. Acuna-Agost, "Airline passenger name record generation using generative adversarial networks," *arXiv preprint arXiv:1807.06657*, 2018.

[27] M. G. Bellemare, I. Danihelka, W. Dabney, S. Mohamed, B. Lakshminarayanan, S. Hoyer, and R. Munos, "The cramer distance as a solution to biased wasserstein gradients," *arXiv preprint arXiv:1705.10743*, 2017.

[28] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proceedings of the VLDB Endowment*, vol. 11, no. 10, pp. 1071–1083, 2018.

[29] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[30] 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.

[31] G. Lederrey, T. Hillel, and M. Bierlaire, "Datgan: Integrating expert knowledge into deep learning for synthetic tabular data," *arXiv preprint arXiv:2203.03489*, 2022.

[32] L. Deecke, I. Murray, and H. Bilen, "Mode normalization," in *International Conference on Learning Representations*, 2018.

[33] 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.

[34] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International conference on machine learning*, pp. 2642–2651, PMLR, 2017.

[35] J. Fonseca and F. Bacao, "Tabular and latent space synthetic data generation: a literature review," *Journal of Big Data*, vol. 10, no. 1, p. 115, 2023.

[36] A. Pathare, R. Mangrulkar, K. Suvarna, A. Parekh, G. Thakur, and A. Gawade, "Comparison of tabular synthetic data generation techniques using propensity and cluster log metric," *International Journal of Information Management Data Insights*, vol. 3, no. 2, p. 100177, 2023.

[37] Ã. Figueira and B. Vaz, "Survey on synthetic data generation, evaluation methods and gans," *Mathematics*, 2022.

[38] 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.

[39] P. Marecha and L. Ye, "Generation and evaluation of tabular data in different domains using gans," *Asian Journal of Research in Computer Science*, 2023.

[40] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[41] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *arXiv preprint arXiv:1604.06737*, 2016.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[43] J. Heaton, "Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, pp. 305–307, 2018.

[44] L. Weng, "From gan to wgan," *arXiv preprint arXiv:1904.08994*, 2019.

[45] B. Becker and R. Kohavi, "Adult." UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[46] S. Vito, "Air Quality." UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C59K5F.

[47] "Apartment for Rent Classified." UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5X623.

[48] R. P. Moro S. and C. P., "Bank Marketing." UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C5K306.

[49] S. Chen, "Beijing PM2.5 Data." UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5JS49.

[50] H. Fanaee-T, "Bike Sharing Dataset." UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

[51] G. Hebrail and A. Berard, "Individual household electric power consumption." UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C58K54.

[52] J. Hogue, "Metro Interstate Traffic Volume." UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5X60B.

[53] R. R. Davari Narjes, Veloso Bruno and G. Joao, "MetroPT-3 Dataset." UCI Machine Learning Repository, 2023. DOI: https://doi.org/10.24432/C5VW3R.

[54] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola, "A kernel method for the two-sample-problem," *Advances in neural information processing systems*, vol. 19, 2006.

[55] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *International Conference on Learning Representations (ICLR 2016)*, pp. 1–10, 2016.

[56] H. Fallahian, "Context aware conditional tabular gan." `https://github.com/samfallahian/ContextAwareTabular-CGAN`, 2024.

[57] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," 2017.