Modelling Tabular Data by Gaussian Mixture Diffusion Models with Prior-Distributional Guidance

Minseo Yoon
Department of Data Science, 2021320322
June 29, 2024
`cooki0615@korea.ac.kr`

Abstract

In the era of big data, generating high-fidelity synthetic data for training machine learning models has become increasingly important. This paper presents a novel approach for generating synthetic tabular data using Gaussian Mixture Diffusion Models (GMDMs) enhanced with a U-Net architecture and Prior-Distributional Guidance (PDG). Our method leverages the flexibility of Gaussian Mixture Models (GMMs) to approximate complex distributions, including multimodal and simpler distributions like multinomial and uniform distributions. We conduct extensive experiments using both classification and regression tasks to evaluate the effectiveness of our approach. Results demonstrate that our method significantly improves machine learning efficiency and preserves the intrinsic correlations of the original datasets. Notably, our models deliver superior performance across diverse datasets, demonstrating their versatility and robustness. The detailed variable-wise absolute differences of correlation matrices further validate our enhancements, providing valuable insights for practitioners aiming to generate high-quality synthetic data.

1 Introduction

In the era of big data, we navigate complex data types, from structured numerical databases to unstructured text and images. This diversity offers immense opportunities and challenges. For instance, Large Language Models (LLMs) like GPT-4 [1] have significantly increased model complexity. The upcoming GPT-5 exacerbates this by requiring massive training data due to its vast size, highlighting the paradox of abundant computational resources paired with data scarcity [2]. Tabular data, characterized by both numerical and categorical variables, is foundational across many fields. A common assumption is that categorical variables can be fully captured by multinomial distributions, given their finite categories. However, modeling numerical variables accurately is crucial for reflecting the dataset's intricacies and ensuring robust data representation.

Gaussian Mixture Models (GMMs) emerge as a potent solution for modeling numerical variables due to their ability to approximate multimodal distributions effectively. Unlike simpler models like multinomial or uniform distributions, GMMs can capture complex, multimodal characteristics inherent in numerical data, thus providing a more nuanced and precise modeling approach. Recent advancements in generative models, particularly diffusion models [3, 4, 5, 6, 7], have revolutionized data generation techniques. Diffusion models, known for their efficacy in generating high-quality synthetic data, are particularly useful for structured data types like tabular data. By leveraging the principles of Gaussian mixture distributions, these models can simulate the complexity of real-world data more accurately.

Our approach draws inspiration from the U-Net [8] structure used in stable diffusion models [7], which have shown remarkable success in conditional generation tasks. The conditional aspect allows the model to treat original data points as conditional inputs, facilitating the generation of new, high-fidelity tabular data entries. This method provides a significant advantage in capturing and preserving the underlying distributions and dependencies within the dataset. Additionally, incorporating residual connections [9] within the U-Net architecture enhances the model's ability to generate high-quality synthetic data. Residual connections help maintain the integrity of the data's original distribution, leading to more accurate and reliable data generation.

By focusing on generating tabular data using Gaussian Mixture Diffusion Models, our research addresses the challenge of data scarcity in the context of large-scale AI systems. The generated data can be used to supplement training datasets, thus improving model generalization and reducing overfitting.

We summarize our contributions as follows:

- We propose a novel method for generating synthetic tabular data using Gaussian Mixture Diffusion Models (GMDMs), which effectively capture complex multimodal distributions and maintain statistical properties of the original data.
- We enhance the generative model with a U-Net architecture incorporating residual connections, significantly improving the fidelity and consistency of the generated data.
- We introduce Prior-Distributional Guidance (PDG) to further refine data generation, ensuring high-quality synthetic data that preserves intrinsic correlations present in the original dataset.
- Through comprehensive experiments on both classification and regression tasks, we demonstrate the superior performance of our method compared to existing models, validating its robustness and scalability in addressing data scarcity challenges.

2 Related Work

Diffusion Models. Diffusion models have gained significant traction in recent years due to their impressive performance in generative tasks, particularly in image and text synthesis. These models, such as those proposed by [3, 4, 5, 6], leverage a forward process that adds noise to data and a reverse process that denoises it, resulting in high-quality data generation. The stable diffusion models [7], which employs a U-Net [8] architecture with residual connections [9], has demonstrated remarkable success in conditional generation tasks, setting a strong precedent for its application in structured data types like tabular data. The effectiveness of these models in capturing complex data distributions and generating realistic synthetic data highlights their potential for broader applications beyond their original domains.

Generative AI and Data Enhancement. Historically, techniques such as data augmentation [10, 11, 12, 13, 14, 15] and contrastive learning [16, 17, 18, 19, 20] have enhanced dataset utility by increasing the variability and robustness of training data. These methods have been pivotal in improving model performance by artificially expanding the dataset and ensuring better generalization. Moreover, generative models have the potential to fill gaps in datasets where data collection is challenging or expensive. By leveraging well-trained embeddings and generative processes, these models can produce high-fidelity data that mimics the statistical properties of real-world data, thus addressing the issue of data scarcity and improving the scalability of AI systems.

Synthetic Tabular Data Generation. Generative models for tabular data are gaining more attention in machine learning [21, 22]. Notably, tabular VAEs (Variational Autoencoders) [23, 24] and GANs (Generative Adversarial Networks) [25, 26, 27, 28, 29] have shown promise in this field. These models have been pivotal in addressing the challenges of tabular data generation, such as capturing complex dependencies between features and ensuring data quality. Recently, TabDDPM [30] (Tabular Diffusion Probabilistic Models) has emerged as a powerful method for tabular data generation, leveraging the strengths of diffusion models to provide more accurate and diverse synthetic data. Our research builds upon the foundational work of TabDDPM, targeting both tabular data generation and imputation. By incorporating Gaussian Mixture Models (GMMs) and enhancing the U-Net architecture with residual connections, we aim to improve the fidelity and consistency of the generated data, ensuring it preserves the integrity of the original data distributions.

3 Methodology

3.1 Preliminaries

Diffusion Models [3] are likelihood-based generative models that process data through forward and reverse Markov chains. The forward process $q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$ gradually adds noise to an initial sample x_0 from the data distribution $q(x_0)$. Each $q(x_t|x_{t-1})$ is defined by a variance schedule $\{\beta_1, ..., \beta_T\}$.

The reverse diffusion process $p(x_{0:T}) = \prod_{t=1}^{T} p(x_{t-1}|x_t)$ progressively denoises a latent variable $x_T \sim q(x_T)$, generating new data samples from $q(x_0)$. Since $p(x_{t-1}|x_t)$ distributions are generally unknown, they are approximated using neural networks with parameters θ. These parameters are learned by optimizing a variational lower bound:

$$\log q(x_0) \geq \mathbb{E}_{q(x_{0:T})}\left[\log p_\theta(x_{0:T}) - \sum_{t=2}^{T} \mathrm{KL}(q(x_{t-1}|x_t, x_0)\|p_\theta(x_{t-1}|x_t))\right]$$

Gaussian diffusion models operate in continuous spaces ($x_t \in \mathbb{R}^n$), where forward and reverse processes are characterized by Gaussian distributions:

$$\begin{aligned} q(x_t|x_{t-1}) &:= \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I\right) \\ p_\theta(x_{t-1}|x_t) &:= \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right) \end{aligned} \tag{1}$$

[3] use diagonal $\Sigma_\theta(x_t, t)$ with a constant σ_t and compute $\mu_\theta(x_t, t)$ as a function of x_t and $\epsilon_\theta(x_t, t)$:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

where $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{i=1}^{t} \alpha_i$. The training objective simplifies to the mean-squared error between $\epsilon_\theta(x_t, t)$ and the true noise ϵ:

$$L_{\text{simple}} = \mathbb{E}_{x_0, \epsilon, t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right]$$

Multinomial diffusion models [6] are designed for categorical data where $x_t \in \{0, 1\}^K$ is a one-hot encoded categorical variable with K values. The forward process $q(x_t|x_{t-1})$ corrupts the data by adding uniform noise over K classes:

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

The reverse distribution $p_\theta(x_{t-1}|x_t)$ is parameterized by $q(x_{t-1}|x_t, \hat{x}_0(x_t, t))$, where $\hat{x}_0$ is predicted by a neural network. The model is trained to maximize the variational lower bound as described earlier.

Gaussian Mixture Models (GMMs) are probabilistic models that assume all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. Each Gaussian distribution in the mixture is defined by its mean and variance, and the entire model is characterized by the mixture weights, means, and variances of the component Gaussians. The probability density function of a GMM is given by:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{2}$$

where π_k are the mixture weights, μ_k and Σ_k are the mean and covariance of the k-th Gaussian component, respectively, and K is the total number of components. The parameters are typically estimated using the Expectation-Maximization (EM) algorithm.

Some of Probability Density Functions (PDFs) are used to specify the probability of a continuous random variable falling within a particular range of values. Two important examples of PDFs in the context of our work are the Multinomial distribution and the Uniform distribution.

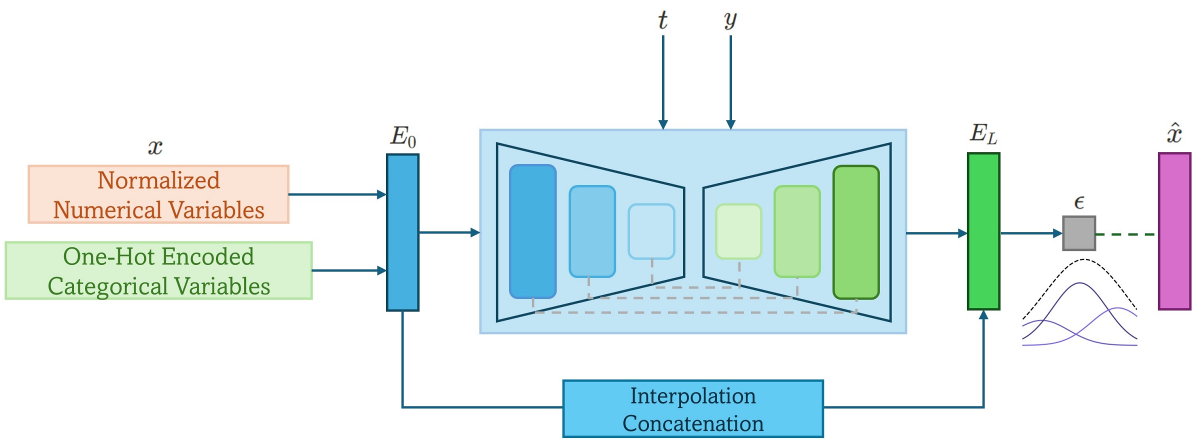
t
y
x
Normalized Numerical Variables
One-Hot Encoded Categorical Variables
E_0
E_L
ϵ
$\hat{x}$
Interpolation Concatenation

Figure 1: Overall architecture of our method. The input x is encoded into E_0, which is processed through a U-Net architecture with multi-layer perceptron to produce the embedding E_L at timestep t with the target variable y. The gray dotted lines represent residual connections. The embeddings E_0 and E_L are either interpolated or concatenated. The noise ϵ is modeled as a mixture of Gaussian noise, and $\hat{x}$ represents the synthesized output.

The Multinomial distribution is a generalization of the binomial distribution. It models the probability of counts for K categories and is defined as:

$$p(x_1, x_2, \dots, x_K | n, p_1, p_2, \dots, p_K) = \frac{n!}{x_1! x_2! \dots x_K!} \prod_{i=1}^{K} p_i^{x_i}$$

where x_i is the count for category i, p_i is the probability of category i, and n is the total count across all categories.

The Uniform distribution models a random variable that has an equal probability of taking any value within a specified range. The PDF of a uniform distribution over the interval $[a, b]$ is given by:

$$p(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

These distributions play a crucial role in various probabilistic modeling approaches, including the methods used in our work.

3.2 Mixture of Gaussian Noise

The first line of Equation 1 can be written as:

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_t$$

where ϵ_t is the Gaussian noise of step t. This can be generalized by adding a mixture of Gaussians at each step, i.e.,

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \sum_{i=k}^{K} \pi_k \epsilon_t^k$$

where π_k are the mixture weights, so $\sum_{k=1}^{K} \pi_k = 1$.

For simplicity, we focus on the case that $K = 2, \pi := \pi_1 = \pi_2 = 0.5$ with zero means and same variances:

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} (p \epsilon_t^1 + (1 - p) \epsilon_t^2)$$

where $\epsilon_t^1, \epsilon_t^2 \sim N(0, \phi_t^2)$ and $p \sim$ Bernoulli(π). In addition to the zero mean property that we assume, we scale the variables such that ϕ_t subject to the variance of $p\epsilon_t^1 + (1 - p)\epsilon_t^2$ is one. In the generalized cases of real application, we reparametrize the added noise at each step subject to Equation 3 in Section 3.3.

3.3 Gaussian Mixture Diffusion Models (GMDMs)

Gaussian Mixture Models (GMMs) are chosen because they effectively approximate multimodal distributions, which can also mimic simpler distributions like multinomial and uniform distributions. This flexibility makes GMMs a powerful tool for modeling the complex distributions often found in numerical data.

To approximate a multimodal continuous distribution, we first observe the distribution via histograms to determine the number of modalities. Then, we use the Gaussian mixture function from scikit-learn [31] to determine the weights of each Gaussian component. Instead of directly using the original noise $N(0,1)$, we ensure the following conditions are met:

$$\sum_{i=1}^{M} \pi_i \mu_i = 0, \quad \sum_{i=1}^{M} \pi_i(\sigma_i^2 + \mu_i^2) = 1 \tag{3}$$

where M is the number of modalities and π_i, μ_i, and σ_i are defined as Equation 2. These conditions ensure that the first and second moments match those of the original standard normal distribution while still approximating the modality through the Gaussian mixture.

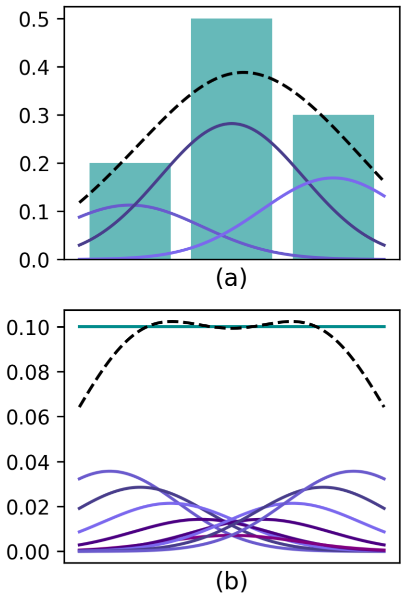

Figure 2: Gaussian mixture approximation of (a) multinomial distribution and (b) uniform distribution. Each colored line represents a Gaussian PDF, and the dotted line represents the Gaussian mixture model.

To approximate a multinomial distribution, the weights π_i are set as the probability mass for each category, adhering to the constraints in Equation 3. Note that this is for ordinal variables where the order of categories is clear. For nominal variables, the simple multinomial diffusion model is used without approximation. For the uniform distribution, the weights are designed to decrease gradually for $i = 1$ to $\lceil M/2 \rceil$ and then increase, while still satisfying the constraints. This approach ensures that the Gaussian mixture can effectively approximate the target distribution's modality and moments. The approximation of these distributions through GMMs allows for more flexible and accurate modeling of numerical data, which is crucial in many machine learning applications. By ensuring that the generated noise follows the desired statistical properties, we can better capture the underlying patterns in the data.

Furthermore, the use of GMDMs provides a robust framework for handling data with multiple modes, which is common in real-world datasets. This method ensures that the generated synthetic data retains the complexity and diversity of the original dataset, making it more suitable for training complex machine learning models. The visualization of this explanation can be seen in Figure 2.

3.4 Prior-Distributional Guidance (PDG)

The rationale behind using the U-Net [8] architecture in our approach is inspired by stable diffusion models [7], which has shown remarkable success in conditional generation tasks. In this context, we treat the original data as a conditional input to guide the generation process. PDG is an innovative approach that enhances the model's ability to generate high-quality data by integrating the original input data through the network. This method leverages two main strategies: Interpolation and Concatenation.

In the Interpolation strategy, the final output $\hat{x}$ is a weighted sum of the encoder's final output $E_L(x)$ and the adjusted original input $E_0(x)$. The parameter α controls the influence of each component, allowing the model to balance between the learned features and the original input:

$$\hat{x} = (1 - \alpha) \cdot E_L(x) + \alpha \cdot E_0(x)$$

In the Concatenation strategy, the final output $\hat{x}$ is produced by concatenating the encoder's final output $E_L(x)$ and the adjusted original input $E_0(x)$, which is then passed through a multi-layer

Abbr	Name	# Train	# Validation	# Test	# Num	# Cat	Task type
CH	Churn Modelling	6400	1600	2000	7	4	Classification
CA	California Housing	13209	3303	4128	8	0	Regression

Table 1: Summary of the datasets used in the experiments, including the abbreviation (Abbr), dataset name (Name), number of training samples (# Train), validation samples (# Validation), test samples (# Test), numerical features (# Num), categorical features (# Cat), and the type of task (Task type).

perceptron (MLP) to generate the final embedding:

$$\hat{x} = \text{MLP}(\text{concat}(E_L(x), E_0(x)))$$

This approach allows the model to leverage both the detailed information from the original input and the abstract features learned through the encoder. By integrating these two sources of information, the model can generate synthetic data that is both accurate and consistent with the original data distribution. The overall architecture of the proposed method, incorporating PDG, can be seen in Figure 1. This architecture ensures that the generated data maintains the statistical properties of the original dataset, leading to improved performance in downstream machine learning tasks.

4 Experiments

Datasets. We conducted experiments on two datasets: the Churn Modelling dataset (CH) and the California Housing dataset (CA). Detailed information about each dataset is provided below and Table 1.

The **Churn Modelling dataset** is used to predict customer churn in a bank. It includes customer data such as credit score, geographic location, gender, age, tenure, balance, number of products, credit card status, activity status, and estimated salary. The target variable indicates whether a customer has churned (left the bank). This dataset allows us to evaluate our method's effectiveness in a binary classification context, aiming to identify customers likely to leave the bank.

The **California Housing dataset** is used to predict house prices in various California districts. It includes features such as median income, average number of rooms and bedrooms, population, households, latitude, and longitude. The target variable is the median house value. This dataset tests our method in a regression context, aiming to predict a continuous variable (house prices) based on several input features. To identify the distribution of each variable, we truncated the top 5% of numerical variables to mitigate the impact of outliers, ensuring a more robust analysis as shown in Appendix D.

Baselines and Ablation Study. We compare the performance of our approach against various versions of models to conduct a thorough evaluation and ablation study:

- **Zero-Shot Generation**: A model that generates data with not trained model but still with diffusion models, serving as a lower bound for performance.
- **TabDDPM** $\cdots$(1): The baseline model of our study, which uses Tabular Diffusion Probabilistic Models without any enhancements.
- **Baseline + GMDMs** $\cdots$(2): The TabDDPM model enhanced with Gaussian Mixture Diffusion Models to better capture multimodal distributions in the data.
- **Baseline + GMDMs + U-Net** $\cdots$(3): This version incorporates the U-Net architecture into the Baseline + GMDMs model, leveraging its success in stable diffusion models for conditional generation tasks.
- **Baseline + GMDMs + U-Net + PDG**: The most advanced version, which integrates Prior-Distributional Guidance into the Baseline + GMDMs + U-Net model to improve the quality of generated data by conditioning on the original input data. This version has two strategies for achieving to model PDG - Interpolation and Concatenation. In the cases of using the Interpolation strategy, we set $\alpha = 0.5$.

By comparing these versions, we systematically evaluate the impact of each component in our model, providing insights into their individual and combined contributions to performance improvement. This

Method		Accuracy	F1 Score	ROC-AUC Score
Zero-Shot Generation		79.65	44.34	50.10
TabDDPM [30] (Baseline)	··· (1)	85.85	75.07	85.71
(1) + GMDMs	··· (2)	<u>86.15</u>	<u>75.43</u>	<u>86.12</u>
(2) + U-Net	··· (3)	**86.30**	**75.67**	85.83
(3) + PDG (Interpolation)		86.10	75.14	85.65
(3) + PDG (Concatenation)		86.05	74.96	**86.23**

Table 2: Performance metrics for the churn modeling dataset, where the target is a categorical variable. The table shows the values of machine learning efficiency computed with regards to the tuned CatBoost model, demonstrating the effectiveness of different model enhancements. Bold indicates the best performance, and underline indicates the second best.

Method		R-Squared	RMSE
Zero-Shot Generation		0.0423	1.1175
TabDDPM [30] (Baseline)	··· (1)	0.8344	0.4647
(1) + GMDMs	··· (2)	0.8320	0.4680
(2) + U-Net	··· (3)	<u>0.8387</u>	<u>0.4586</u>
(3) + PDG (Interpolation)		**0.8399**	**0.4569**
(3) + PDG (Concatenation)		0.8370	0.4611

Table 3: Performance metrics for the California housing dataset, where the target is a numerical variable. The table shows the values of machine learning efficiency computed with regards to the tuned CatBoost model, demonstrating the effectiveness of different model enhancements. Bold indicates the best performance, and underline indicates the second best.

ablation study helps us understand the effectiveness of Gaussian Mixture Models, U-Net architecture, and Prior-Distributional Guidance in enhancing the baseline model.

Metrics. Our primary evaluation metric is machine learning (ML) efficiency (or utility), as proposed by [28]. ML efficiency quantifies the performance of classification or regression models trained on synthetic data when evaluated on a real test set. The rationale is that models trained on high-quality synthetic data should perform comparably to, or even better than, models trained on real data.

For our experiments, we employ an evaluation protocol that calculates ML efficiency using the CatBoost model, noted for its superior performance on tabular tasks [32]. This protocol provides a more practical measure of the value of synthetic data, as practitioners typically prefer robust and optimal classifiers and regressors in real-world scenarios.

Additionally, we report the absolute differences between the correlation matrices computed on real and synthetic data across various datasets. To compute these correlation matrices, we use the Pearson correlation coefficient, transforming categorical variables using the LabelEncoder function in scikit-learn [31]. This metric helps assess how well the synthetic data preserves the relationships between variables in the real data.

4.1 Machine Learning Efficiency

We evaluate the machine learning efficiency of our proposed models using both classification and regression tasks. The performance metrics are calculated with respect to the tuned CatBoost model, which provides superior performances for tabular data tasks. The models are compared based on their accuracy, F1 score, and ROC-AUC score for classification, as well as R-squared and RMSE for regression.

Classification Performance. Table 2 presents the performance metrics for the churn modeling dataset, demonstrating the effectiveness of various model enhancements. The GMDMs and U-Net model achieved the highest Accuracy (86.30) and F1 Score (75.67), while the PDG (Concatenation) model had the best ROC-AUC Score (86.23). The baseline model showed significant improvement over random generation, and adding GMDMs and U-Net further enhanced performance. PDG models, despite slight drops in some metrics, maintained competitive scores, indicating their potential depending on application needs.

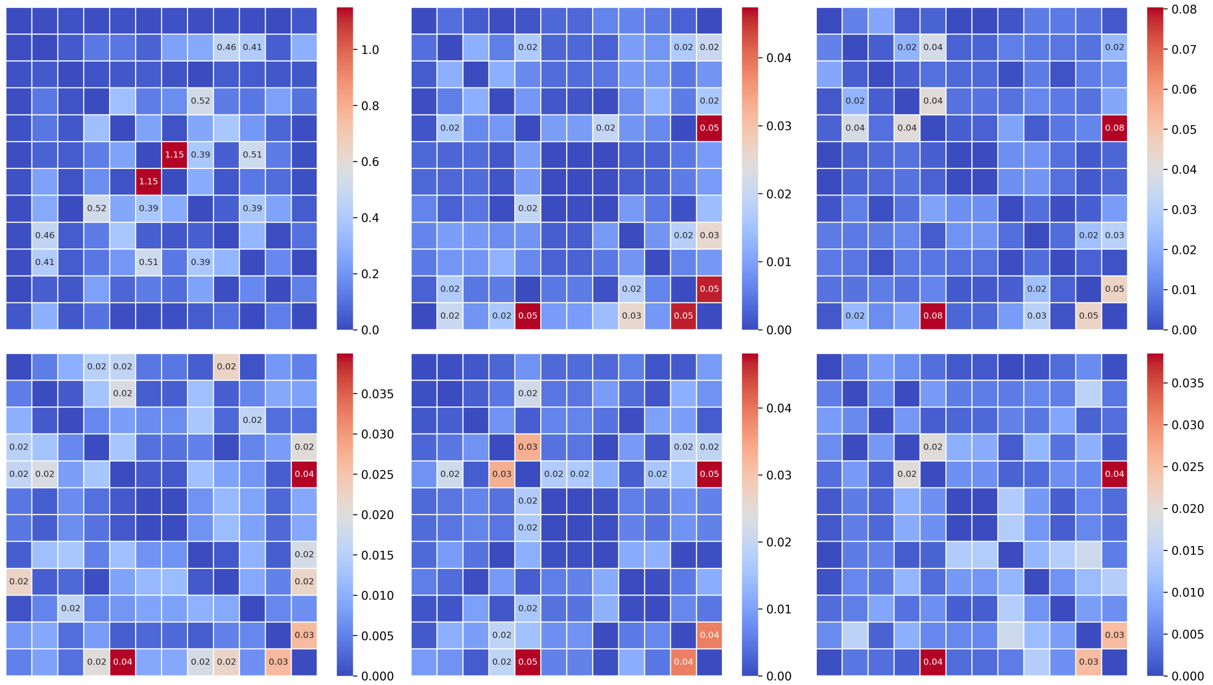
Figure 3: Variable-wise absolute differences of correlation matrices for the churn modeling dataset. The subplots are arranged in the same order as reported in the metric table: (Top row, left to right) Zero-Shot Generation, TabDDPM (Baseline), (1) + GMDMs, (Bottom row, left to right) (2) + U-Net, (3) + PDG (Interpolation), (3) + PDG (Concatenation).

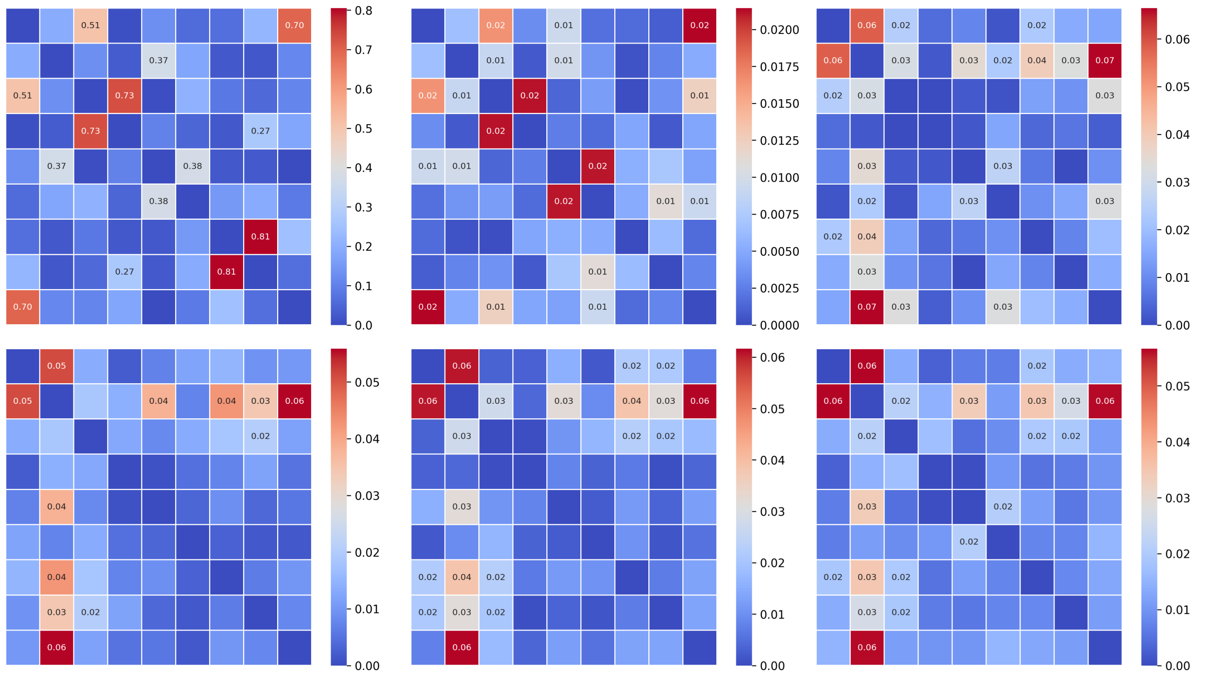
Figure 4: Variable-wise absolute differences of correlation matrices for the California housing dataset. The subplots are arranged in the same order as reported in the metric table: (Top row, left to right) Zero-Shot Generation, TabDDPM (Baseline), (1) + GMDMs, (Bottom row, left to right) (2) + U-Net, (3) + PDG (Interpolation), (3) + PDG (Concatenation).

Method		CH	CA
Zero-Shot Generation		9.593	6.525
TabDDPM [30] (Baseline)	··· (1)	0.522	**0.224**
(1) + GMDMs	··· (2)	0.700	0.620
(2) + U-Net	··· (3)	0.543	<u>0.497</u>
(3) + PDG (Interpolation)		<u>0.521</u>	0.530
(3) + PDG (Concatenation)		**0.434**	0.555

Table 4: Absolute Differences of Correlation Matrices for the churn modeling dataset (CH) and the California housing dataset (CA). The reported metrics are calculated as the summation of lower triangular entries. Bold indicates the best performance, and underline indicates the second best.

Regression Performance. Table 3 presents the performance metrics for the California housing dataset, demonstrating the effectiveness of different model enhancements. The (3) + PDG (Interpolation) model outperformed others, with the highest R-Squared (0.8399) and lowest RMSE (0.4569). The (2) + U-Net model also performed well, indicating that adding U-Net to the baseline + GMDMs configuration significantly improves performance. The baseline model showed strong performance, validating its robustness, while GMDMs, U-Net, and PDG provided additional benefits.

In conclusion, the ablation study confirms that each enhancement contributes to improved performance. GMDMs capture multimodal distributions better, U-Net improves data representation, and PDG ensures high-quality data generation by conditioning on original inputs.

4.2 Absolute Differences of Correlation Matrices

Table 4 presents the absolute differences between the correlation matrices computed on real and synthetic data for the churn modeling dataset (CH) and the California housing dataset (CA).

For the churn modeling dataset, the (3) + PDG (Concatenation) model had the lowest absolute difference (0.434), indicating it best preserves original data correlations, followed by the (3) + PDG (Interpolation) model (0.521).

For the California housing dataset, the baseline model showed the best performance (0.224), followed by the (2) + U-Net model (0.497). The slightly worse metrics for GMM-enhanced models are due to the distributional instability of the second variable, which exhibits a multimodal Gaussian distribution with values clustered at a specific upper bound. More detailed analysis can be seen in Appendix C. Despite this, overall performance improved, capturing complex data distributions effectively.

These results validate the effectiveness of different enhancements in preserving intrinsic correlations of the original datasets. The PDG models significantly improve the churn modeling dataset, highlighting the benefits of incorporating original data as conditional inputs. The baseline model's strong performance in the California housing dataset confirms its robustness, and the U-Net model proves beneficial in maintaining data integrity. The variable-wise absolute differences of correlation matrices are shown in Figures 3 and 4.

5 Conclusion

This study introduces a method for generating synthetic tabular data using Gaussian Mixture Diffusion Models, a U-Net architecture, and Prior-Distributional Guidance. Our approach significantly enhances the machine learning efficiency of models trained on synthetic data. Comprehensive evaluations demonstrate that our method models show substantial improvements in both classification and regression tasks, particularly in the churn modeling and California housing datasets. Additionally, our method effectively maintains intrinsic data correlations, providing a robust framework for generating high-fidelity synthetic data and offering practical benefits for various machine learning applications.

Future work will explore further enhancements and broader applications of this approach to solidify its utility in the field of synthetic data generation. Specifically, we aim to investigate methods for more accurately modeling a wider variety of probability distributions, enhancing the versatility and precision of synthetic data generation.

References

[1] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Seohee Lee. “gpt-5” *Gongburyang 8baena neuneunde... deo hakseupsikil deiteoga eopda.* , Apr 2024.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[4] Yang Song and Stefano Ermon. Denoising diffusion implicit models. In *CVPR*, 2021.

[5] Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, 2021.

[6] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. 2021.

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[10] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[11] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NeurIPS*, 2014.

[12] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

[13] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[14] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[15] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[16] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[18] Xinlei Chen and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2103.07579*, 2021.

[19] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2022.

[20] Kaiming He, Xinlei Chen, Saining Xie, Yang Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[21] Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *AAAI*, 2021.

[22] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Trans. Neural Netw. Learn. Syst.*, 2024.

[23] Chao Ma, Sebastian Tschiatschek, José Miguel Hernández-Lobato, Richard Turner, and Cheng Zhang. Vaem: A deep generative model for heterogeneous mixed type data. *arXiv preprint arXiv:2006.11941*, 2020.

[24] Patricia A Apellániz, Juan Parras, and Santiago Zazo. An improved tabular data generator with vae-gmm integration. *arXiv preprint arXiv:2404.08434*, 2024.

[25] Karim Armanious, Chenming Jiang, Marc Fischer, Thomas Küstner, Tobias Hepp, Konstantin Nikolaou, Sergios Gatidis, and Bin Yang. Medgan: Medical image translation using gans. *Comput. Med. Imaging Graph.*, 2020.

[26] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *VLDB Endowment*, 2018.

[27] Ying Yu, Bingying Tang, Ronglai Lin, Shufa Han, Tang Tang, and Ming Chen. Cwgan: Conditional wasserstein generative adversarial nets for fault data generation. In *ROBIO*, 2019.

[28] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *arXiv preprint arXiv:1907.00503*, 2019.

[29] Zilong Zhao, Aditya Kunar, Hiek Van der Scheer, Robert Birke, and Lydia Y Chen. Ctab-gan: Effective table data synthesizing. *arXiv preprint arXiv:2104.08730*, 2021.

[30] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *ICML*, 2023.

[31] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *JMLR*, 2011.

[32] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, 2018.

Parameter	Churn Modelling	California Housing
Model Parameters		
Dimension of Layers	[512, 1024, 1024, 1024, 1024, 512]	[512, 1024, 1024, 1024, 1024, 512]
Dropout Rate	0.0	0.0
Diffusion Parameters		
Number of Timesteps	100	1000
Gaussian Loss Type	MSE	MSE
Scheduler	Cosine	-
Training Parameters		
Steps	30000	30000
Learning Rate	0.0008057094292475385	0.0013275991211473216
Weight Decay	0.0	0.0
Batch Size	4096	4096
Normalization	Quantile	Quantile
Sampling Parameters		
Number of Samples	26000	52800
Sampling Batch Size	10000	8192

Table 5: Hyperparameter settings for the Churn Modelling and California Housing datasets.

A Implementation Detail

The experiments were conducted using either a single T4 GPU or an L4 GPU on Colab Pro. On average, each experiment took approximately 20 minutes with a T4 GPU. The U-Net architecture, originally designed for segmentation tasks with a CNN encoder-decoder structure, was adapted in our implementation by using an MLP. Despite this change, we retained the use of residual connections to enhance model performance and stability.

For the diffusion model, we utilized Denoising Diffusion Probabilistic Models (DDPM) [3]. The choice of DDPM was driven by its robustness and suitability for our synthetic data generation tasks. Although our codebase supports Denoising Diffusion Implicit Models (DDIM) [4], we opted not to conduct experiments with DDIM. This decision was based on the consideration that the complex distributions used to model the noise would make the diffusion process with fewer timesteps less appropriate. Simplifying the process in this way might compromise the ability to accurately capture and reproduce the intricate patterns present in the original datasets.

B Hyperparameter Setting

In this section, we outline the hyperparameters used in our experiments for the Churn Modelling and California Housing datasets. These hyperparameters are critical in ensuring optimal performance and effective learning during the training process.

For both datasets, we used a similar model architecture with specified dimensions of layers and dropout rates. The diffusion parameters, including the number of timesteps and the Gaussian loss type, were adjusted according to the complexity of the dataset. Training parameters such as the number of steps, learning rate, weight decay, and batch size were carefully selected to balance computational efficiency and model accuracy. We also applied quantile normalization in scikit-learn [31] to preprocess the data, ensuring consistent input distributions. Finally, sampling parameters were set to generate a sufficient number of synthetic samples for evaluation.

The detailed hyperparameter settings for each dataset are summarized in Table 5.

C Exploratory Data Analysis

In this section, we include the Exploratory Data Analysis (EDA) for the two datasets used in our experiments. Our primary objective is to approximate numerical variables using Gaussian Mixture Models (GMMs). Therefore, even for datasets like the churn modeling dataset, which include categorical variables, our analysis will focus primarily on the numerical variables.

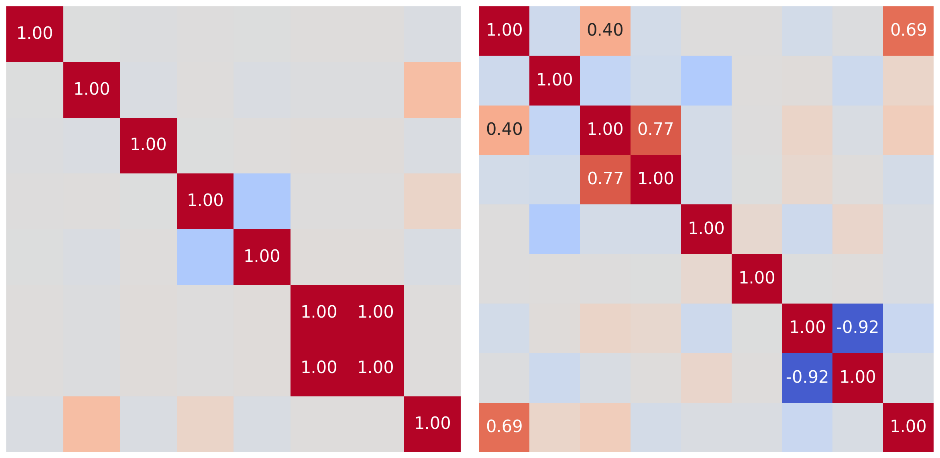

Figure 5: Variable-wise correlation matrices for the Churn Modeling dataset (left) and the California Housing dataset (right).

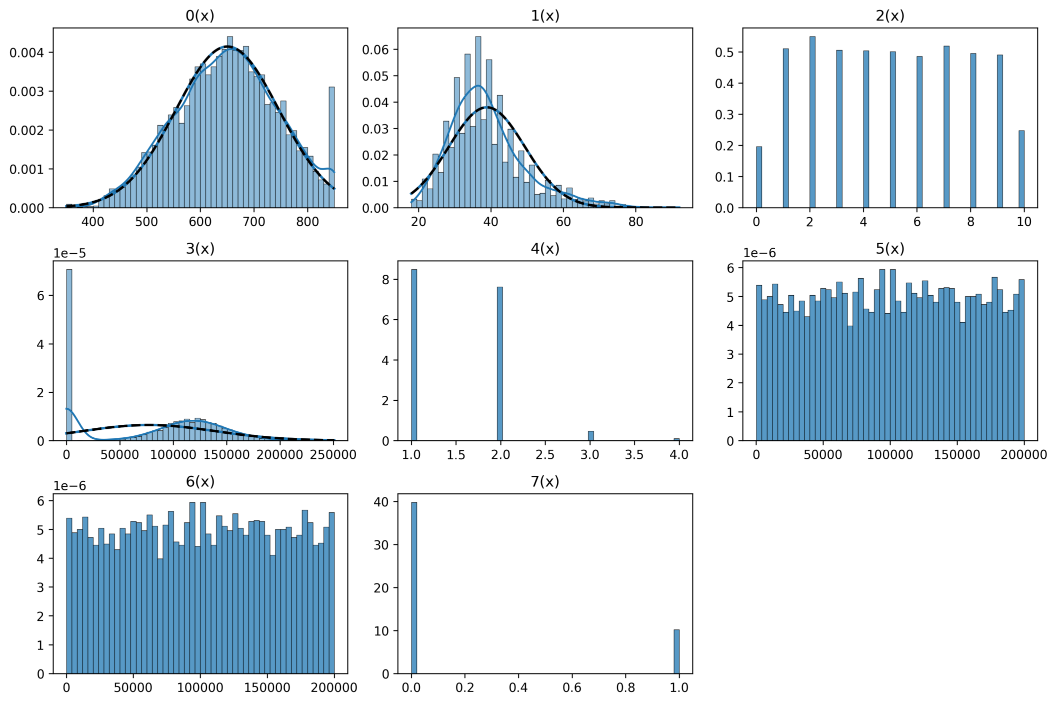

Figure 6: Histogram with the kernel density estimation (KDE) plot and estimated probability density functions for the Churn Modeling dataset. The dotted lines represent the estimated Gaussian probability density functions.

Through this analysis, we aim to understand the underlying distributions and characteristics of the numerical variables in each dataset. This understanding is crucial for effectively modeling these variables using GMMs and for identifying any preprocessing steps necessary to ensure data quality and stability.

The correlation matrices and probability density function graphs are shown in Figure 5 for the correlation matrices, and Figures 6 and 7 for the probability density functions.

The estimated probability density functions for each variable of the Churn Modelling dataset are provided below.

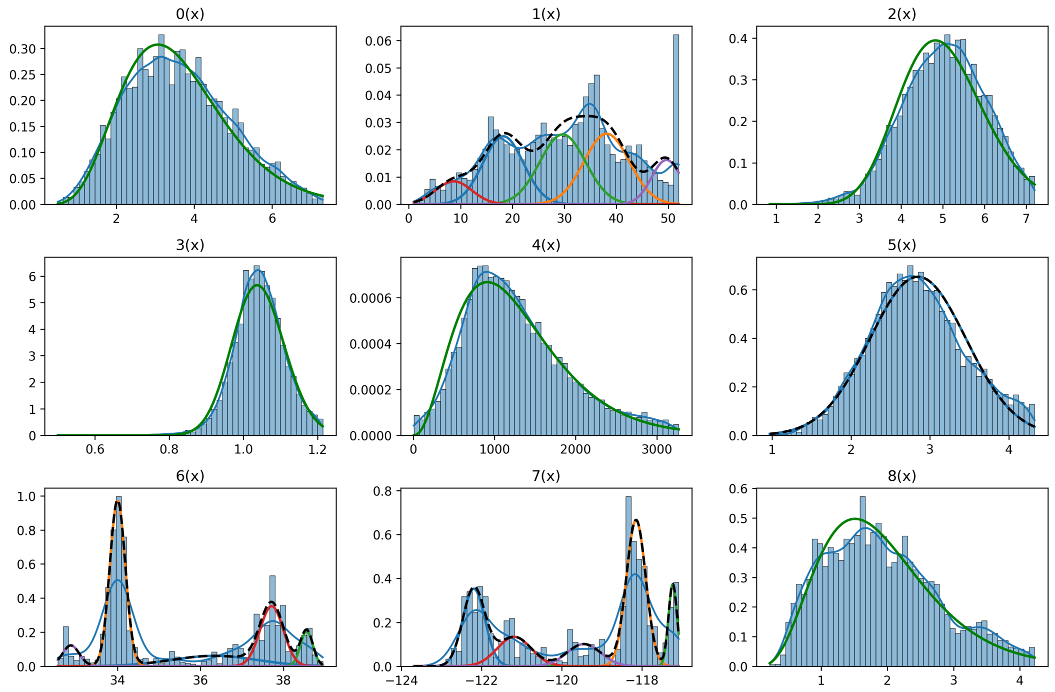

Figure 7: Histogram with the kernel density estimation (KDE) plot and estimated probability density functions for the California Housing dataset. The solo green line represents the estimated Gamma probability density function. The other solid lines represent the estimated Gaussian probability density functions, while the dotted lines indicate the Gaussian Mixture Model (GMM) estimates.

$$0(x) \sim \text{Gaussian} : f(x) = \frac{1}{\sqrt{2\pi \cdot 9266.639}} e^{-\frac{(x-650.649)^2}{2\cdot 9266.639}}$$
$$1(x) \sim \text{Gaussian} : f(x) = \frac{1}{\sqrt{2\pi \cdot 110.459}} e^{-\frac{(x-38.871)^2}{2\cdot 110.459}}$$
$$\begin{aligned}2(x) \sim \text{Multinomial} : f(x) = &\ 0.039 \cdot \delta(x-0) + 0.102 \cdot \delta(x-1) + 0.110 \cdot \delta(x-2) \\ &+0.101 \cdot \delta(x-3) + 0.101 \cdot \delta(x-4) + 0.100 \cdot \delta(x-5) + 0.097 \cdot \delta(x-6) \\ &+0.104 \cdot \delta(x-7) + 0.099 \cdot \delta(x-8) + 0.098 \cdot \delta(x-9) + 0.049 \cdot \delta(x-10)\end{aligned}$$
$$3(x) \sim \text{Gaussian} : f(x) = \frac{1}{\sqrt{2\pi \cdot 3.867 \times 10^9}} e^{-\frac{(x-77196.331)^2}{2\cdot 3.867\times 10^9}}$$
$$4(x) \sim \text{Multinomial} : f(x) = 0.509\cdot\delta(x-1)+0.457\cdot\delta(x-2)+0.028\cdot\delta(x-3)+0.006\cdot\delta(x-4)$$
$$5(x) \sim \text{Uniform} : f(x) = \frac{1}{199992.484 - 11.580} \quad \text{for} \quad 11.580 \le x \le 199992.484$$
$$6(x) \sim \text{Uniform} : f(x) = \frac{1}{199992.484 - 11.580} \quad \text{for} \quad 11.580 \le x \le 199992.484$$
$$7(x) \sim \text{Multinomial} : f(x) = 0.796 \cdot \delta(x-0) + 0.204 \cdot \delta(x-1)$$

In the above equations, $\delta(x-k)$ represents the Dirac delta function, which is used to indicate the probability mass at each discrete value k. The Dirac delta function is defined as:

$$\delta(x-k) = \begin{cases} \infty & \text{if } x = k \\ 0 & \text{if } x \neq k \end{cases}$$

with the property that $\int_{-\infty}^{\infty} \delta(x-k)\, dx = 1$. This function effectively captures the discrete nature of the probabilities in the multinomial distribution.

The estimated probability density functions for each variable of the California Housing dataset are provided below.

$$0(x) \sim \text{Gamma} : f(x) = \frac{x^{5.750} e^{-x/0.534}}{0.534^{6.750}\Gamma(6.750)}$$

$$1(x) \sim \text{GMM} : f(x) = 0.244 \cdot \frac{1}{\sqrt{2\pi \cdot 15.228}} e^{-\frac{(x-18.211)^2}{2 \cdot 15.228}} + 0.283 \cdot \frac{1}{\sqrt{2\pi \cdot 19.125}} e^{-\frac{(x-38.139)^2}{2 \cdot 19.125}}$$
$$+0.283 \cdot \frac{1}{\sqrt{2\pi \cdot 19.453}} e^{-\frac{(x-29.663)^2}{2 \cdot 19.453}} + 0.074 \cdot \frac{1}{\sqrt{2\pi \cdot 12.250}} e^{-\frac{(x-8.673)^2}{2 \cdot 12.250}}$$
$$+0.117 \cdot \frac{1}{\sqrt{2\pi \cdot 8.197}} e^{-\frac{(x-49.640)^2}{2 \cdot 8.197}}$$

$$2(x) \sim \text{Gamma} : f(x) = \frac{x^{22.878} e^{-x/0.211}}{0.211^{23.878}\Gamma(23.878)}$$

$$3(x) \sim \text{Gamma} : f(x) = \frac{x^{216.610} e^{-x/0.005}}{0.005^{217.610}\Gamma(217.610)}$$

$$4(x) \sim \text{Gamma} : f(x) = \frac{x^{2.486} e^{-x/366.629}}{366.629^{3.486}\Gamma(3.486)}$$

$$5(x) \sim \text{Gaussian} : f(x) = \frac{1}{\sqrt{2\pi \cdot 0.373}} e^{-\frac{(x-2.848)^2}{2 \cdot 0.373}}$$

$$6(x) \sim \text{GMM} : f(x) = 0.158 \cdot \frac{1}{\sqrt{2\pi \cdot 1.068}} e^{-\frac{(x-36.332)^2}{2 \cdot 1.068}} + 0.455 \cdot \frac{1}{\sqrt{2\pi \cdot 0.035}} e^{-\frac{(x-34.999)^2}{2 \cdot 0.035}}$$
$$+0.077 \cdot \frac{1}{\sqrt{2\pi \cdot 0.022}} e^{-\frac{(x-38.562)^2}{2 \cdot 0.022}} + 0.248 \cdot \frac{1}{\sqrt{2\pi \cdot 0.078}} e^{-\frac{(x-37.718)^2}{2 \cdot 0.078}}$$
$$+0.062 \cdot \frac{1}{\sqrt{2\pi \cdot 0.041}} e^{-\frac{(x-32.873)^2}{2 \cdot 0.041}}$$

$$7(x) \sim \text{GMM} : f(x) = 0.245 \cdot \frac{1}{\sqrt{2\pi \cdot 0.077}} e^{-\frac{(x+122.193)^2}{2 \cdot 0.077}} + 0.405 \cdot \frac{1}{\sqrt{2\pi \cdot 0.059}} e^{-\frac{(x+118.143)^2}{2 \cdot 0.059}}$$
$$+0.101 \cdot \frac{1}{\sqrt{2\pi \cdot 0.012}} e^{-\frac{(x+117.226)^2}{2 \cdot 0.012}} + 0.142 \cdot \frac{1}{\sqrt{2\pi \cdot 0.180}} e^{-\frac{(x+121.184)^2}{2 \cdot 0.180}}$$
$$+0.107 \cdot \frac{1}{\sqrt{2\pi \cdot 0.178}} e^{-\frac{(x+119.446)^2}{2 \cdot 0.178}}$$

$$8(x) \sim \text{Gamma} : f(x) = \frac{x^{3.715} e^{-x/0.407}}{0.407^{4.715}\Gamma(4.715)}$$

D More Detailed Information about Datasets

As introduced in Section 4, we truncated the top 5% of each variable in the California Housing dataset to determine their distributions. The probability density function graphs before truncating are shown in Figure 8.

The following equations illustrate the instability of the distributions before truncation.

$$0(x) \sim \text{Gamma} : f(x) = \frac{x^{3.717} e^{-x/0.823}}{0.823^{4.717}\Gamma(4.717)}$$

$$1(x) \sim \text{GMM} : f(x) = 0.184 \cdot \frac{1}{\sqrt{2\pi \cdot 21.897}} e^{-\frac{(x-46.864)^2}{2 \cdot 21.897}} + 0.334 \cdot \frac{1}{\sqrt{2\pi \cdot 22.810}} e^{-\frac{(x-19.961)^2}{2 \cdot 22.810}}$$
$$+ 0.364 \cdot \frac{1}{\sqrt{2\pi \cdot 19.722}} e^{-\frac{(x-33.522)^2}{2 \cdot 19.722}} + 0.118 \cdot \frac{1}{\sqrt{2\pi \cdot 19.051}} e^{-\frac{(x-9.519)^2}{2 \cdot 19.051}}$$

$$2(x) \sim \text{Lognormal} : f(x) = \frac{1}{x \cdot 5.207 \cdot \sqrt{2\pi}} e^{-\frac{(\ln(x)-0.000)^2}{2 \cdot 0.272^2}}$$

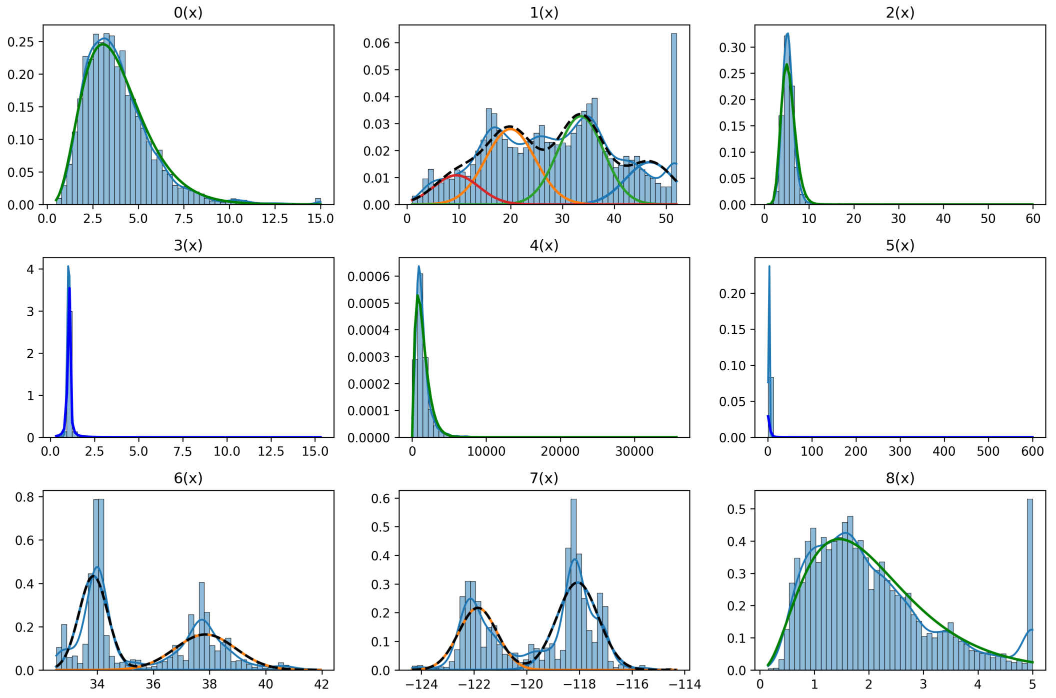

Figure 8: Histogram with the kernel density estimation (KDE) plot and estimated probability density functions for the California Housing dataset. The solo green line represents the estimated Gamma or lognormal probability density function and the solo deep blue line represents the estimated Cauchy probability density function. The other solid lines represent the estimated Gaussian probability density functions, while the dotted lines indicate the Gaussian Mixture Model (GMM) estimates.

$$3(x) \sim \text{Cauchy} : f(x) = \frac{1}{\pi \cdot 0.044 \left[1 + \left(\frac{x-1.045}{0.044}\right)^2\right]}$$

$$4(x) \sim \text{Lognormal} : f(x) = \frac{1}{x \cdot 1114.725 \cdot \sqrt{2\pi}} e^{-\frac{(\ln(x)-0.000)^2}{2\cdot 0.742^2}}$$

$$5(x) \sim \text{Cauchy} : f(x) = \frac{1}{\pi \cdot 0.389 \left[1 + \left(\frac{x-2.784}{0.389}\right)^2\right]}$$

$$6(x) \sim \text{GMM} : f(x) = 0.556 \cdot \frac{1}{\sqrt{2\pi \cdot 0.262}} e^{-\frac{(x-33.861)^2}{2\cdot 0.262}} + 0.444 \cdot \frac{1}{\sqrt{2\pi \cdot 1.168}} e^{-\frac{(x-37.850)^2}{2\cdot 1.168}}$$

$$7(x) \sim \text{GMM} : f(x) = 0.604 \cdot \frac{1}{\sqrt{2\pi \cdot 0.623}} e^{-\frac{(x+118.066)^2}{2\cdot 0.623}} + 0.396 \cdot \frac{1}{\sqrt{2\pi \cdot 0.532}} e^{-\frac{(x+121.854)^2}{2\cdot 0.532}}$$

$$8(x) \sim \text{Gamma} : f(x) = \frac{x^{2.359} e^{-x/0.618}}{0.618^{3.359}\Gamma(3.359)}$$

The above distributions show the variability and potential instability in the original dataset, highlighting the need for truncation to achieve more stable and reliable data distributions. The lognormal distributions indicate a long tail distribution, suggesting the presence of many values far from the mean. This can lead to significant instability in the data. Additionally, the Cauchy distributions are known for having extreme outliers, which further contribute to the instability of the dataset. These factors combined result in variables that are inherently unstable, necessitating preprocessing steps such as truncation to ensure the robustness of the dataset for modeling purposes.

Table 6 and Figure 9 provide further evidence of how outliers adversely affect the California Housing dataset.

Method	R-Squared	RMSE	Abs. Diff.
TabDDPM [30] + GMDMs	-6.479	3.123	13.457

Table 6: Performance metrics for the California housing dataset, where the top 5% of numerical variables are not truncated when identifying the distribution of each variable. Abs. Diff. represents Absolute Differences of Correlation Matrices.

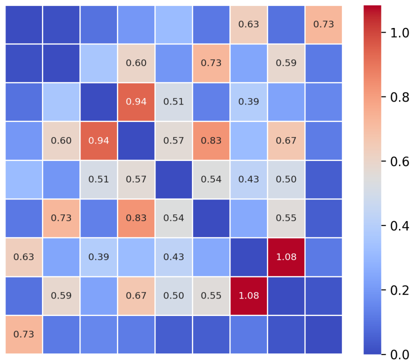

Figure 9: Variable-wise absolute differences of correlation matrices for the California housing dataset, where the top 5% of numerical variables are not truncated when identifying the distribution of each variable.