Tabular Data Generation using Binary Diffusion

Vitaliy Kinakh
Department of Computer Science
University of Geneva
Geneva, Switzerland
vitaliy.kinakh@unige.ch

Slava Voloshynovskiy
Department of Computer Science
University of Geneva
Geneva, Switzerland

Abstract

Generating synthetic tabular data is critical in machine learning, especially when real data is limited or sensitive. Traditional generative models often face challenges due to the unique characteristics of tabular data, such as mixed data types and varied distributions, and require complex preprocessing or large pretrained models. In this paper, we introduce a novel, lossless binary transformation method that converts any tabular data into fixed-size binary representations, and a corresponding new generative model called Binary Diffusion, specifically designed for binary data. Binary Diffusion leverages the simplicity of XOR operations for noise addition and removal and employs binary cross-entropy loss for training. Our approach eliminates the need for extensive preprocessing, complex noise parameter tuning, and pretraining on large datasets. We evaluate our model on several popular tabular benchmark datasets, demonstrating that Binary Diffusion outperforms existing state-of-the-art models on Travel, Adult Income, and Diabetes datasets while being significantly smaller in size. Code and models are available at: https://github.com/vkinakh/binary-diffusion-tabular

1 Introduction

The generation of synthetic tabular data is a critical task in machine learning, particularly when dealing with sensitive, private, or scarce real-world data. Traditional generative models often struggle with the inherent complexity and diversity of tabular data, which typically encompasses mixed data types and complex distributions.

In this paper, we introduce a method to transform generic tabular data into a binary representation, and a generative model named Binary Diffusion, specifically designed for binary data. Binary Diffusion leverages the simplicity of XOR operations for noise addition and removal, fundamental components of probabilistic diffusion models. This method eliminates the need for extensive preprocessing and complex noise parameter tuning, streamlining the data preparation process.

Our approach offers several key advantages. First, by converting all columns into unified binary representations, the proposed transformation removes the necessity for column-specific preprocessing commonly required in handling mixed-type tabular data. Secondly, the Binary Diffusion model itself is optimized for binary data, utilizing binary cross-entropy (BCE) loss for predictions during the training of the denoising network.

We evaluate our model on several popular tabular benchmark datasets, including Travel [tej21], Sick [SED+88], HELOC [lia18, FIC18], Adult Income [BK96], California Housing [PB97, nug17], and Diabetes [SDG+14, Kag21] tabular datasets. The Binary Diffusion model outperforms existing state-of-the-art models on Travel, Adult Income and Dianetes datasets. Additionally, our model is significantly smaller in size compared to contemporary models and does not require pretraining

Table Representation Learning Workshop at NeurIPS 2024.

on other data modalities, unlike methods based on large language models (LLMs) such as GReaT [BSL+22].

2 Related Work

TVAE (Tabular Variational Autoencoder) adapts the Variational Autoencoder (VAE) framework to handle mixed-type tabular data by separately modeling continuous and categorical variables. **CTGAN** (Conditional Tabular GAN) employs a conditional generator to address imbalanced categorical columns, ensuring the generation of diverse and realistic samples by conditioning on categorical data distributions. **CopulaGAN** integrates copulas with GANs to capture dependencies between variables, ensuring that synthetic data preserves the complex relationships present in the original dataset [XSCIV19].

GReaT (Generation of Realistic Tabular data) [BSL+22] leverages a pretrained auto-regressive language model (LLM) to generate highly realistic synthetic tabular data. The process involves fine-tuning the LLM on textually encoded tabular data, transforming each row into a sequence of words. This approach allows the model to condition on any subset of features and generate the remaining data without additional overhead.

Existing data generation methods show several shortcomings. Models such as CopulaGAN, CTGAN, and TVAE attempt to generate columns with both continuous and categorical data simultaneously, employing activation functions like *softmax* and *tanh* in the outputs. These models also require complex preprocessing of continuous values and rely on restrictive approximations using Gaussian mixture models and mode-specific normalization. Additionally, large language model-based generators like GReaT need extensive pretraining on text data, making them computationally intensive with large parameter counts with potential bias from the pretraining data.

The proposed data transformation and generative model address these shortcomings as follows: (i) by converting all columns to unified binary representations; (ii) the proposed generative model for binary data, with fewer than 2M parameters, does not require pretraining on large datasets and offers both fast training and sampling capabilities.

3 Data transformation

To apply the Binary Diffusion model to tabular data, we propose an invertible lossless transformation $\mathcal{T}$, shown on the Figure 1, that converts tabular data columns into fixed-size binary representations. The transformations is essential for preparing tabular data for the Binary Diffusion model, enabling it to process and generate tabular data without the need for extensive preprocessing. This approach ensures that the data retains its original characteristics.

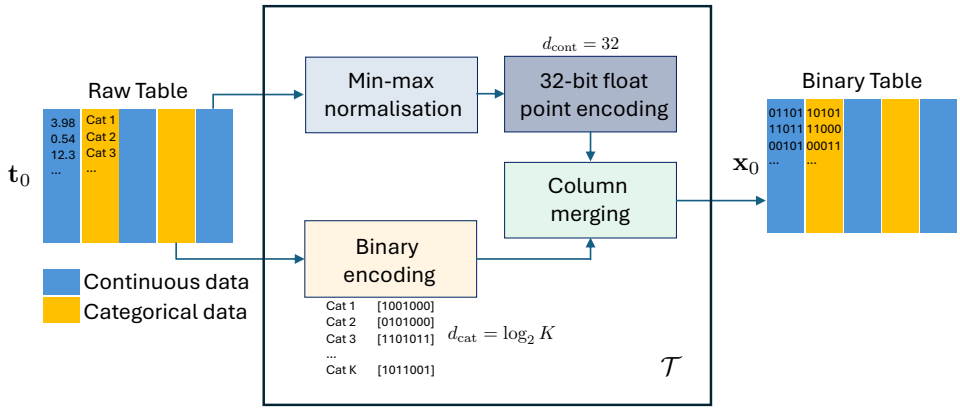


Figure 1: Transformation of tabular data $\mathbf{t}_0$ into the binary form $\mathbf{x}_0$. The considered transformation is reversible. The continuous column records are presented with the length $d_{\text{cont}} = 32$ and the categorical ones with $d_{\text{cat}} = \log_2 K$, where K stands for the number of categorical classes.

The transformation method converts each column of the table into a binary format. For continuous data, this process includes applying min-max normalization to the columns, followed by converting these normalized values into a binary representation via 32-bit floating-point encoding. For categorical data, binary encoding is used. The encoded columns are concatenated into fixed-size rows.

The inverse transformation $\mathcal{T}^{-1}$ converts the binary representations back into their original form. For continuous data, the decoded values are rescaled to their original range using metadata generated during the initial transformation. For categorical data, the binary codes are mapped back to their respective categories using a predefined mapping scheme.

4 Binary Diffusion

Binary Diffusion shown in Figure 2 is a novel approach for generative modeling that leverages the simplicity and robustness of binary data representations. This method involves adding and removing noise through XOR operation, which makes it particularly well-suited for handling binary data. Below, we describe the key aspects of the Binary Diffusion methodology in detail.

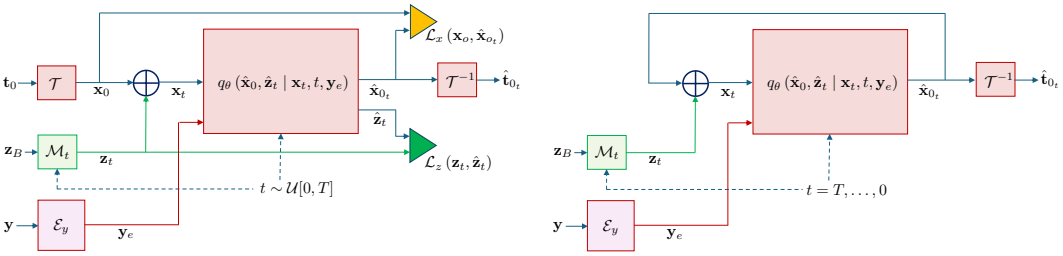


Figure 2: Binary Diffusion training (left) and sampling (right) schemes.

In Binary Diffusion, noise is added to the data by flipping bits using the XOR operation with a random binary mask. The amount of noise added is quantified by the proportion of bits flipped. Let $\mathbf{x}_0 \in \{0,1\}^d$ be the original binary vector of dimension d, and $\mathbf{z}_t \in \{0,1\}^d$ be a random binary noise vector at timestep t. The noisy vector $\mathbf{x}_t$ is obtained as: $\mathbf{x}_t = \mathbf{x}_0 \oplus \mathbf{z}_t$, where $\oplus$ denotes the XOR operation. The noise level is defined as the fraction of bits flipped in $\mathbf{z}_t$ in the mapper $\mathcal{M}_t$ at step t, with the number of bits flipped ranging within $[0, 0.5]$ as a function of the timestep.

The denoising network $q_\theta(\hat{\mathbf{x}}_0, \hat{\mathbf{z}}_t | \mathbf{x}_t, t, \mathbf{y}_e)$ is trained to predict both the added noise $\mathbf{z}_t$ and the clean-denoised vector $\mathbf{x}_0$ from the noisy vector $\mathbf{x}_t$. We employ binary cross-entropy (BCE) loss (1) to train the denoising network. The loss function is averaged over both the batch of samples and the dimensions of the vectors:

$$
\begin{aligned}
\mathcal{L}(\theta) &= \frac{1}{B}\sum_{b=1}^{B}\left[\mathcal{L}_x(\hat{\mathbf{x}}_0^{(b)}, \mathbf{x}_0^{(b)}) + \mathcal{L}_z(\hat{\mathbf{z}}_t^{(b)}, \mathbf{z}_t^{(b)})\right] \\
&= -\frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{d}\left[\mathbf{x}_{0i}^{(b)} \log \hat{\mathbf{x}}_{0i}^{(b)} + (1 - \mathbf{x}_{0i}^{(b)}) \log(1 - \hat{\mathbf{x}}_{0i}^{(b)})\right] \\
&\quad - \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{d}\left[\mathbf{z}_{ti}^{(b)} \log \hat{\mathbf{z}}_{ti}^{(b)} + (1 - \mathbf{z}_{ti}^{(b)}) \log(1 - \hat{\mathbf{z}}_{ti}^{(b)})\right],
\end{aligned}
\tag{1}
$$

where B is the batch size, θ represents the parameters of the denoising network, $\mathbf{x}_0^{(b)}$ and $\hat{\mathbf{x}}_0^{(b)}$ are the b -th samples of the true clean vectors and the predicted clean vectors, respectively. Similarly, $\mathbf{z}_t^{(b)}$ and $\hat{\mathbf{z}}_t^{(b)}$ are the b-th samples of the true added noise vectors and the predicted noise vectors, respectively. $\mathbf{y}_e = \mathcal{E}_y(\mathbf{y})$ denotes the encoded label $\mathbf{y}$, one-hot encoded for classification and min-max normalized for regression. $\mathcal{L}_x$ and $\mathcal{L}_z$ denotes binary cross-entropy (BCE) loss. The indices i and b correspond to the i-th dimension of the vectors and the b-th sample in the batch, respectively.

During training (Figure 2 left), we use classifier-free guidance [HS22]. For classification tasks, the conditioning input class label $\mathbf{y}$ is a one-hot encoded label $\mathbf{y}_e$. For regression tasks, $\mathbf{y}$ consists of

min-max normalized target values $\mathbf{y}_e$, allowing the model to generate data conditioned on specific numerical outcomes. In unconditional training, we use an all-zeros conditioning vector for classification tasks and a value of -1 for regression tasks to indicate the absence of conditioning.

When sampling (Figure 2 right), we start from a random binary vector $\mathbf{x}_t$ at timestep $t = T$, along with the conditioning variable $\mathbf{y}$, encoded into $\mathbf{y}_e$. For each selected timestep in the sequence $[T, \ldots, 0]$, denoising is applied to the vector. The denoised vector $\hat{\mathbf{x}}_0$ and the estimated binary noise $\hat{\mathbf{z}}_t$ are predicted by the denoising network. These predictions are then processed using a sigmoid function and binarized with a threshold. During sampling, we use the denoised vector $\hat{\mathbf{x}}_0$ directly. Then, random noise $\mathbf{z}_t$ is generated and added to $\hat{\mathbf{x}}_0$ via the XOR operation: $\mathbf{x}_t = \hat{\mathbf{x}}_0 \oplus \mathbf{z}_t$. The sampling algorithm is summarized in Algorithm 1.

5 Results

We evaluate the performance of Binary Diffusion on widely-recognized tabular benchmark datasets, including Travel [tej21], Sick [SED+88], HELOC [lia18, FIC18], Adult Income [BK96], California Housing [PB97, nug17], and Diabetes [SDG+14, Kag21]. For classification tasks (Travel, Sick, HELOC, Adult Income, and Diabetes), classification accuracy was used as metric, while mean squared error (MSE) was used for the regression task (California Housing). Following the evaluation protocol established in [BSL+22], we employed Linear/Logistic Regression (LR), Decision Tree (DT), and Random Forest (RF) as downstream models to assess the quality of the synthetic data. The datasets were split into training and test sets with an 80/20 split. The generative models were trained on the training set, and the test set was reserved for evaluation. To ensure robustness, 5 sets of synthetic training data were generated, and the results are reported as average performances with corresponding standard deviations. Table 1 shows the detailed results. Binary Diffusion achieved superior performance compared to existing state-of-the-art models on the Travel, Adult Income, and Diabetes datasets. Notably, Binary Diffusion maintained competitive results on the HELOC and Sick datasets, despite having a significantly smaller parameter footprint (ranging from 1.1M to 2.6M parameters) compared to models like GReaT, which utilize large language models with hundreds of millions of parameters. Binary Diffusion does not require pretraining on external data modalities, enhancing its efficiency and reducing potential biases associated with pretraining data. In the regression task (California Housing), Binary Diffusion demonstrated competitive MSE scores. Additionally, Binary Diffusion offers faster training and sampling times, as detailed in Appendix C. Implementation details are summarized in Appendix D.

Table 1: Quantitative results on table dataset benchmarks. The best results are marked in **bold**, second-best are underlined. The number of parameters for every model and dataset are provided in 4-th row for every dataset.

Dataset	Model	Original	TVAE	CopulaGAN	CTGAN	Distill-GReaT	GReaT	Binary Diffusion
Travel (↑)	LR	82.72±0.00	79.58±0.00	73.30±0.00	73.30±0.00	78.53±0.00	80.10±0.00	**83.79±0.08**
	DT	89.01±0.00	81.68±1.28	73.61±0.26	73.30±0.00	77.38±0.51	83.56±0.42	**88.90±0.57**
	RF	85.03±0.53	81.68±1.19	73.30±0.00	71.41±0.53	79.50±0.53	84.30±0.33	**89.95±0.44**
	Params	-	36K	157K	155K	82M	355M	1.1M
Sick (↑)	LR	96.69±0.00	94.70±0.00	94.57±0.00	94.44±0.00	96.56±0.00	**97.72±0.00**	96.14±0.63
	DT	98.94±0.12	95.39±0.18	93.77±0.01	92.05±0.41	95.39±0.18	**97.72±0.00**	97.07±0.24
	RF	98.28±0.06	94.91±0.06	94.57±0.01	94.57±0.00	97.72±0.00	**98.30±0.13**	96.59±0.55
	Params	-	46K	226K	222K	82M	355M	1.4M
HELOC (↑)	LR	71.80±0.00	71.04±0.00	42.03±0.00	57.72±0.00	70.58±0.00	**71.90±0.00**	71.76±0.30
	DT	81.90±1.06	76.39±0.50	42.36±0.10	61.34±0.09	81.40±0.15	**79.10±0.07**	70.25±0.43
	RF	83.19±0.71	77.24±0.25	42.35±0.34	62.35±0.35	**82.14±0.13**	80.93±0.28	70.47±0.32
	Params	-	62K	276K	277K	82M	355M	2.6M
Adult Income (↑)	LR	85.00±0.00	80.53±0.00	80.61±0.00	83.20±0.00	84.65±0.00	84.77±0.00	**85.45±0.11**
	DT	85.27±0.01	82.80±0.08	76.29±0.06	81.32±0.02	84.49±0.04	84.81±0.04	**85.27±0.11**
	RF	85.93±0.11	83.48±0.11	80.46±0.21	83.53±0.07	85.25±0.07	85.42±0.05	**85.74±0.11**
	Params	-	53K	300K	302K	82M	355M	1.4M
Diabetes (↑)	LR	58.76±0.00	56.34±0.00	40.27±0.00	50.93±0.00	57.33±0.00	57.34±0.00	**57.75±0.04**
	DT	57.29±0.03	53.30±0.08	38.50±0.02	49.73±0.02	54.10±0.04	55.23±0.04	**57.13±0.15**
	RF	59.00±0.08	55.17±0.10	37.59±0.31	52.23±0.17	58.03±0.16	**58.34±0.09**	57.52±0.12
	Params	-	369K	9.4M	9.6M	82M	355M	1.8M
California Housing (↓)	LR	0.40±0.00	0.65±0.00	0.98±0.00	0.61±0.00	0.57±0.00	**0.34±0.00**	0.55±0.00
	DT	0.32±0.01	0.45±0.01	1.19±0.01	0.82±0.01	0.43±0.01	**0.39±0.01**	0.45±0.00
	RF	0.21±0.01	0.35±0.01	0.99±0.01	0.62±0.01	0.32±0.01	**0.28±0.01**	0.39±0.00
	Params	-	45K	201K	197K	82M	355M	1.5M

6 Conclusions

This paper proposed a novel lossless binary transformation method for tabular data, which converts any data into fixed-size binary representations. Building upon this transformation, we introduced the Binary Diffusion model, a generative model specifically designed for binary data that utilizes XOR operations for noise addition and removal and is trained using binary cross-entropy loss. Our approach addresses several shortcomings of existing methods, such as the need for complex preprocessing, reliance on large pretrained models, and computational inefficiency.

We evaluated our model on several tabular benchmark datasets, and demonstrated that Binary Diffusion achieves state-of-the-art performance on these datasets while being significantly smaller in size compared to existing models. Our model does not require pretraining on other data modalities, which simplifies the training process and avoids potential biases from pretraining data. Our findings indicate that the proposed model works particularly well with datasets that have a high proportion of categorical columns.

References

[BK96] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[BSL+22] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. *arXiv preprint arXiv:2210.06280*, 2022.

[FIC18] FICO. Explainable machine learning challenge. `https://community.fico.com/s/explainable-machine-learning-challenge`, 2018. Accessed: 2024-09-13.

[GDW+22] Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`, 2022.

[HG16] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[HS22] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[Kag21] Kaggle. Lab diabetes readmission prediction. `https://www.kaggle.com/c/1056lab-diabetes-readmission-prediction`, 2021. Accessed: 2024-09-13.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[lia18] Home equity line of credit (heloc) dataset. `https://www.kaggle.com/datasets/averkiyoliabev/home-equity-line-of-creditheloc`, 2018. Accessed: 2024-09-13.

[nug17] California housing prices. `https://www.kaggle.com/datasets/camnugent/california-housing-prices`, 2017. Accessed: 2024-09-13.

[PB97] R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[SDG+14] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. Impact of hba1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. *BioMed Research International*, 2014:1–11, 2014.

[SED+88] Jack W. Smith, James E. Everhart, William C. Dickson, William C. Knowler, and Robert S. Johannes. Using the adap learning algorithm to forecast the onset of diabetes mellitus. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pages 261–265, 1988.

[tej21] Tour travels customer churn prediction dataset. `https://www.kaggle.com/datasets/tejashvi14/tour-travels-customer-churn-prediction`, 2021. Accessed: 2024-09-13.

[XSCIV19] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

A Sampling algorithm

Algorithm 1 Sampling algorithm.

$\mathbf{x}_t \leftarrow$ random binary tensor
$\mathbf{y} \leftarrow$ condition/label
$\mathbf{y}_e \leftarrow$ apply condition enxoding
$threshold \leftarrow$ threshold value to binarize ▷ Default 0.5
$q_\theta(\hat{\mathbf{x}}_0, \hat{\mathbf{z}}_t | \mathbf{x}_t, t, \mathbf{y}_e) \leftarrow$ pre-trained denoiser network
for $t \in \{T, \ldots, 0\}$ **do** ▷ Selected timesteps
    $\hat{\mathbf{x}}_0, \hat{\mathbf{z}}_t \leftarrow q_\theta(\hat{\mathbf{x}}_0, \hat{\mathbf{z}}_t | \mathbf{x}_t, t, \mathbf{y}_e)$
    $\hat{\mathbf{x}}_0 \leftarrow \sigma(\hat{\mathbf{x}}_0) > threshold$ ▷ Apply sigmoid and compare to threshold
    $\mathbf{z}_t \leftarrow get\_binary\_noise(t)$ ▷ Generate random noise
    $\mathbf{x}_t \leftarrow \hat{\mathbf{x}}_0 \oplus \mathbf{z}_t$ ▷ Update $\mathbf{x}_t$ using XOR with $\mathbf{z}_t$
end for
return $\mathbf{x}_t$

B Evaluation models hyperparameters

During evaluation, we follow the evaluation proposed in [BSL+22]. The hyperparameter configuration of the evaluation models for the ML efficiency experiments are provided in Table 2.

Table 2: Evaluation models hyperparameters.

	LR	DT	RF	
Dataset	max_iter	max_depth	max_depth	n_estimators
Travel	100	6	12	75
Sick	200	10	12	90
HELOC	500	6	12	78
Adult Income	1000	8	12	85
Diabetes	500	10	20	120
California Housing	-	10	12	85

C Runtime comparison

We compare the training and sampling times, the number of training epochs, batch sizes, and peak VRAM utilization of generative models. The results, including the number of training epochs and batch sizes required for each model to converge, are summarized in Table 3. Specifically, for TVAE, CopulaGAN, and CTGAN, we employed the default batch size of 500 and trained for 200 epochs; for Distill-GReaT and GReaT, we used a batch size of 32 and trained for 200 epochs; and for Binary Diffusion, a batch size of 256 and 500 epochs were utilized to ensure model convergence. For this study, we utilized the Adult Income dataset. All experiments were conducted on a PC with a single RTX 2080 Ti GPU, an Intel Core i9-9900K CPU 3.60 GHz with 16 threads, 64 GB of RAM, and Ubuntu 20.04 LTS as the operating system.

Table 3: Comparison of training and sampling times, and peak VRAM utilization.

Model	Epochs	Batch size	Training time	Sampling time (s)	Peak VRAM use
TVAE	200	500	2 min 21 sec	0.036 ± 0.001	240 MiB
CopulaGAN	200	500	4 min 26 sec	0.101 ± 0.003	258 MiB
CTGAN	200	500	4 min 33 sec	0.055 ± 0.005	258 MiB
Distill-GReaT	200	32	5 h 7 min	7.104 ± 0.025	8184 MiB
GReaT	200	32	7 h 33 min	11.441 ± 0.034	8548 MiB
Binary Diffusion	5000	256	53 min 2 sec	0.347 ± 0.006	266 MiB

D Implementation details

Denoiser Architecture. We use a similar denoiser architecture across all datasets, which takes as input a noisy vector x_t of size d, a timestep t, and an input condition y. The input size d corresponds to the size of the binary vector in each dataset. The input vector x_t is projected through a linear layer with 256 output units. The timestep t is processed using a sinusoidal positional embedding, followed by two linear layers with 256 output units each, interleaved with GELU activation functions [HG16]. The input condition y is processed through a linear projector with 256 output units. The outputs of the timestep embedding and the condition projector are then combined via element-wise addition. This combined representation is subsequently processed by three ResNet blocks that incorporate timestep embeddings. Depending on the size of the binary representation for each dataset, the number of parameters varies between 1.1 million and 1.4 million.

Training and Sampling Details. We trained the denoiser for 50,000 steps using the Adam optimizer [KB14] with a learning rate of 1×10^{-4}, a weight decay of 0, and a batch size of 256. To maintain a distilled version of the denoiser, we employed an Exponential Moving Average (EMA) with a decay rate of 0.995, updating it every 10 training steps. This distilled model was subsequently used for sampling. During training, we utilized classifier-free guidance with a 10% probability of using a zero token. The diffusion model was configured to perform 1,000 denoising steps during training. Given the relatively small size of our models, we opted for full-precision training. All training parameters are summarized in Table 4.

Table 4: Binary Diffusion training details.

config	value
optimizer	Adam
learning rate	1e-4
weight decay	0
batch size	256
training steps	500000
EMA decay	0.995
EMA update frequency	10
classifier-free guidance zero token	0.1
precision	fp32
diffusion timesteps	1000

We empirically observed that model performance, measured by accuracy for classification tasks and mean squared error (MSE) for regression tasks deteriorates as the number of sampling steps increases. We selected 5 sampling steps and a guidance scale of 5 for all datasets to optimize performance.

Table 5: Binary Diffusion sampling details.

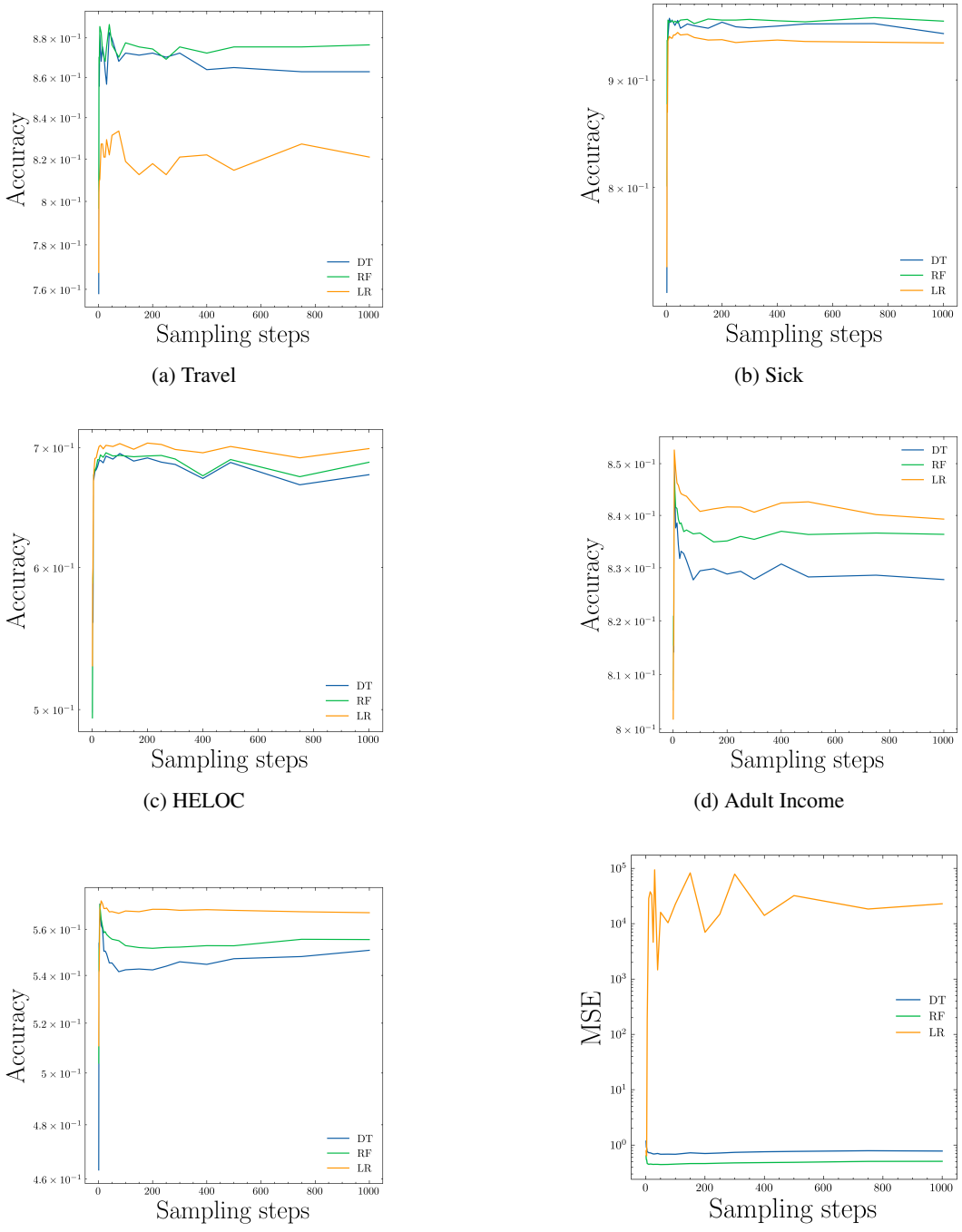
config	value
sampling steps	5
guidance scale	5
EMA	True

Environment. All experiments were conducted on a PC with a single RTX 2080 Ti GPU, an Intel Core i9-9900K CPU 3.60 GHz with 16 threads, 64 GB of RAM, and Ubuntu 20.04 LTS as the operating system. We utilized PyTorch [PGM+19] with the Accelerate [GDW+22] library for training generative models, and the scikit-learn [PVG+11] library for evaluating models.

E Effect of sampling steps

We empirically observed that model performance, measured by accuracy for classification tasks and mean squared error (MSE) for regression tasks, deteriorates as the number of sampling steps increases. Notably, for regression tasks, linear regression models show significantly poorer performance with an increasing number of sampling steps. For our analysis, we utilized an Exponential Moving Average (EMA) denoiser with a guidance scale of 5. Across all datasets, the optimal results were consistently

achieved when the number of sampling steps was 5. The relationship between the number of sampling steps and model performance is illustrated in Figure 3.



(a) Travel

(b) Sick

(c) HELOC

(d) Adult Income

(e) Diabetes

(f) California Housing

Figure 3: Analysis of model performance for different numbers of sampling steps. DT stands for Decision Tree model, RF stands for Random Forest model and LR stands for Linear/Logistic regression model.