Tabular out-of-distribution data synthesis for enhancing robustness

Bhagyashree Puranik[1], **Bugra Can**[2], **Yi Fan**[2]

[1] Amazon*
[2] Amazon Web Services
bpuranik@amazon.com, bugracan@amazon.com, fnyi@amazon.com

Abstract

Many critical machine learning applications in cybersecurity, healthcare and finance, encounter challenges like data privacy, distribution shifts and class imbalance. Often, minority class labels are scarce and may only be present for specific types of samples, which can pose challenges for developing effective models that handle new and unforeseen minority examples at inference time. Additionally, feeding sensitive data into downstream models is a significant privacy concern. Synthetic data generation offers a potential solution by enabling data privacy, creating data samples to rebalance the classes and also provides a way to generate out-of-distribution samples. We introduce TabOOD, a novel approach that generates synthetic tabular data samples to enhance robustness against unseen data and distribution shifts. TabOOD generates out-of-distribution samples that could augment the training set, simulating unobserved scenarios and enhancing downstream model robustness. It also allows for the conditional generation of in-distribution minority and majority class samples. Building on recent advances in tabular data synthesis using latent diffusion models, our approach maps tabular data to class-dependent Gaussian mixture components in a latent space, thereby separating latent representations, before training diffusion models on the latent space. We further manipulate the latent space to generate atypical, boundary data points. Experimental results across different datasets demonstrate that TabOOD significantly improves the performance of downstream models when faced with distribution shifts or novel out-of-distribution samples, offering a more balanced and robust approach to tabular data learning.

Introduction

Tabular data plays a crucial role in various machine learning applications, such as cybersecurity (Zoppi, Gazzini, and Ceccarelli 2024), healthcare (Ching et al. 2018; Miotto et al. 2018), and finance (Huang, Chai, and Cho 2020; Ozbayoglu, Gudelek, and Sezer 2020). In recent years, deep learning models have demonstrated superior performance compared to traditional statistical methods in analyzing this type of data. However, this improved performance can often be limited to the provided training data, as deep learning models may overfit to the specific distribution of the training set. This raises significant concerns about the real-world applicability and robustness of these models, particularly in high-stakes domains like cybersecurity, finance, and healthcare. In these fields, where tasks like anomaly detection, fraud identification, and disease diagnosis are critical, out-of-distribution (OOD) robustness is essential for reliable and effective model deployment. One promising strategy to address this challenge is the introduction of synthetic or artificial datasets during the model training process. By augmenting the existing training data with these synthetic examples, the diversity and volume of the training set can be increased, potentially enhancing the deep learning model's ability to generalize and perform well in OOD scenarios.

Early work (Xu et al. 2019) demonstrated the potential of generative models like GANs and VAEs to create synthetic tabular datasets. However, these generated datasets often fail to capture the full range of real-world data correlations, especially in complex datasets with numerical, categorical, and mixed features (Zhang et al. 2024a). Accurately synthesizing diverse data that faithfully mimics the original distribution remains a key challenge for these traditional generative approaches. To address this challenge, researchers have adapted diffusion models which have already achieved remarkable success in image generation tasks (Ho, Jain, and Abbeel 2020; Rombach et al. 2022). Applying diffusion processes directly to tabular data introduces new challenges related to handling heterogeneous or high-cardinality categorical features, which can increase model complexity and make it harder to learn meaningful joint distributions. Nevertheless, a recent algorithm, TabSyn proposed in (Zhang et al. 2024a) has shown promising results. This method combines a transformer-based variational autoencoder (VAE) to map the tabular data to a latent space, followed by a score-based diffusion model to generate synthetic samples. This hybrid architecture has resulted in stable training and remarkable performance across multiple evaluation metrics, suggesting diffusion models hold great potential for advancing the state-of-the-art in high-fidelity tabular data synthesis.

Another stream of research focuses on leveraging large language models (LLMs) for tabular data generation (Borisov et al. 2023; Kim, Kim, and Choo 2024; Xu et al. 2024). The GReaT approach (Borisov et al. 2023), which fine-tunes a pretrained LLM, stands out. This method encodes each row of tabular data using tokenization and

*This work was done during the author's internship at Amazon Web Services.

random permutations and then generates synthetic data in an auto-regressive manner. By avoiding one-hot encoding, this technique circumvents the curse of dimensionality but is computationally intensive, both in terms of memory and processing time.

Most current methods focus on learning the training data distribution. However, many real-world applications, such as cybersecurity, benefit from generation mechanisms that produce both in-distribution samples and OOD samples. In cybersecurity, for instance, the threat landscape evolves, making it crucial to handle unseen OOD attacks effectively. While methods like domain adaptation and transfer learning attempt to address these issues, they often require labeled data from new distributions or complex model adjustments.

Our work introduces a novel approach to address this by generating OOD samples alongside in-distribution data. By leveraging latent space manipulation and interpolation, we generate synthetic data that not only aligns with the training distribution but also creates edge samples simulating unseen OOD scenarios. This approach enhances model robustness to distribution shifts and diverse test-time scenarios, ensuring more reliable performance when encountering novel OOD instances.

Background Our approach builds upon the foundation of TabSyn (Zhang et al. 2024a), necessitating a brief overview of its methodology. TabSyn integrates a diffusion model within the latent space of a VAE to perform unconditional synthesis of tabular data. The authors propose a transformer-based VAE architecture, prefixed with a tokenizer (and similarly a detokenizer) which maps each feature of the tabular data sample into a d-dimensional embedding. This is fed to the β-VAE. The encoder of the VAE outputs the mean and log variance of the latent variables which are then fed into the decoder. Finally the decoder reconstructs the token matrix, on which a learnable linear transformation, referred to as a detokenizer, is applied to recover the tabular data sample $\hat{\boldsymbol{x}}$. The β-VAE is trained to minimize:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \beta \mathcal{L}_{\text{KL}}, \tag{1}$$

where $\mathcal{L}_{\text{recon}}$ is a reconstruction loss acting on true and recovered data samples $\boldsymbol{x}$ and $\hat{\boldsymbol{x}}$ respectively. $\mathcal{L}_{\text{KL}}$ imposes a regularization term that encourages the latent variables to follow a standard multivariate isotropic Gaussian distribution.

Once the VAE has been trained, the latent embeddings for the training samples are extracted from the VAE's latent space. The distribution of the latent embeddings is then modeled using a score-based diffusion model, following the forward and reverse processes outlined in standard diffusion approaches. (Please refer to (Zhang et al. 2024b) for the details). A denoising neural network, implemented as a multilayer perceptron (MLP), is trained to approximate the noise using the perturbed data. After training, synthetic tabular data can be generated by reversing the diffusion process.

In (Zhang et al. 2024a), the authors compare the performance of TabSyn with the several state-of-the-art (SOTA) baselines (CTGAN (Hong, Yi, and Lee 2024), TVAE (Ishfaq, Hoogi, and Rubin 2018), StaSy (Kim, Lee, and Park 2022), TabDDPM (Kotelnikov et al. 2023),GReat (Borisov et al. 2023), SMOTE (Chawla et al. 2002)) on six tabular datasets and their results demonstrate the effectiveness of the generated synthetic data across various criteria: performance on downstream classification tasks, feature-wise density estimation, pairwise column correlations, and higher-order metrics such as fidelity, diversity, and privacy preservation. We build upon TabSyn for two key reasons: (i) it produces high-quality samples that either outperform or match those generated by other SOTA methods (ii) latent generative models provide an opportunity for manipulation of embeddings based on domain knowledge, which is often challenging in LLM-based methods. Furthermore, our focus extends beyond pure data synthesis to the generation of OOD samples, making TabSyn a suitable building block for our approach.

Synthesis of Tabular OOD Samples

In this section, we introduce our TabOOD approach for generating controlled OOD samples[1]. The approach consists of three key components: (i) learning latent representations through a VAE that maps samples from different classes to distinct Gaussian mixture components in the latent space, with isotropic covariance and distinct means, resulting in well-separated representations; (ii) training score-based diffusion models using denoising score matching to model the distribution of each Gaussian mixture component; and (iii) generating OOD samples by interpolating between the latent embeddings and decoding these interpolated embeddings using the VAE's decoder.

Let $\boldsymbol{x}$ represent a tabular data sample and M denote the total number of features. The features are split into M_{num} numerical features and M_{cat} categorical features. The VAE architecture follows the design of TabSyn, which utilizes a tokenizer, detokenizer, encoder, and decoder denoted by $\texttt{tok}(\cdot)$, $\texttt{detok}(\cdot)$, $\texttt{Enc}(\cdot)$ and $\texttt{Dec}(\cdot)$ respectively. Each feature is processed by the $\texttt{tok}(\cdot)$, which is a learnable linear transformation that maps the feature to a d-dimensional vector. Categorical features are one-hot encoded before this transformation. The tokenized output of the data sample $\boldsymbol{x}$ is represented as:

$$\boldsymbol{T} = [\boldsymbol{t}_1^{\text{num}}, \cdots, \boldsymbol{t}_{M_{\text{num}}}^{\text{num}}, \boldsymbol{t}_1^{\text{cat}}, \cdots, \boldsymbol{t}_{M_{\text{cat}}}^{\text{cat}}], \tag{2}$$

where $\boldsymbol{T} \in \mathbb{R}^{M \times d}$. This is passed through the $\texttt{Enc} - \texttt{Dec}$ structure which are transformer blocks. The VAE encoder produces learns mean $\boldsymbol{\mu}$ and the log variance $\log \boldsymbol{\sigma}^2$ of the latent embeddings, via the standard reparametrization trick (Kingma and Welling 2014). These latent embeddings are passed through the decoder to reconstruct the token matrix $\hat{\boldsymbol{T}} \in \mathbb{R}^{M \times d}$. The detokenizer, which is another learnable linear transformation followed by a softmax layer for categorical features, is applied to recover the reconstructed sample $\hat{\boldsymbol{x}} = \texttt{detok}(\texttt{Dec}(\hat{\boldsymbol{T}}))$.

To ensure separation in the latent space, we propose learning distinct Gaussian mixture components corresponding to different groups. While our exposition and experiments focus on class-based separation assuming a binary classifica-

[1]We refer readers to Appendix for the algorithm outline

tion setting, these groups can be defined by any set of features, depending on the desired type of OOD generation. Let $\boldsymbol{z} = \texttt{Flat}(\texttt{Enc}(\boldsymbol{T}) \in \mathbb{R}^{Md}$ denote the flattened latent embedding output by the encoder. The approximate posterior distribution of the latent variable $\boldsymbol{z}$ given input $\boldsymbol{x}_k$, which belongs to class k, is denoted by $q(\boldsymbol{z}|\boldsymbol{x}_k)$, modeled as a multivariate Gaussian with mean $\boldsymbol{\mu}$ and diagonal covariance matrix diag$(\boldsymbol{\sigma}^2)$. Thus the overall posterior $q(\boldsymbol{z}|\boldsymbol{x})$ is a mixture of Gaussian components. The prior distribution $p(\boldsymbol{z}|\boldsymbol{x}_k)$ is assumed to be class-conditional:

$$p(\boldsymbol{z}|\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\text{prior},k}, I), \tag{3}$$

where $k \in \{0, 1\}$ represents the class label, $\boldsymbol{\mu}_{\text{prior},k}$ is the class-conditional mean vector, and I is the identity covariance matrix. The Kullback-Leibler (KL) divergence between the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x}_k)$ and the class-conditional prior $p(\boldsymbol{z}|\boldsymbol{x}_k)$ is given by:

$$D_{\text{KL}}(q(\boldsymbol{z}|\boldsymbol{x}_k) \parallel p(\boldsymbol{z}|\boldsymbol{x}_k)) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_k)}\left[\log \frac{q(\boldsymbol{z}_k|\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{x}_k)}\right]. \tag{4}$$

For the two multivariate Gaussian distributions $q(\boldsymbol{z}|\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2 I)$ and $p(\boldsymbol{z}|\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{\mu}_{\text{prior},k}, I)$, it can be seen that the KL divergence simplifies to;

$$\begin{aligned} &D_{\text{KL}}(q(\boldsymbol{z}|\boldsymbol{x}_k) \parallel p(\boldsymbol{z}|\boldsymbol{x}_k)) \\ &= -\frac{1}{2}\sum_{i=1}^{Md}\left(1 + \log\sigma_{k,i}^2 - \sigma_{k,i}^2 - (\mu_{k,i} - \mu_{\text{prior},k,i})^2\right). \end{aligned} \tag{5}$$

The β-VAE is trained by minimizing the following objective function:

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \beta D_{\text{KL}}, \tag{6}$$

where D_{KL} is as defined in Equation 5, and $\mathcal{L}_{\text{recon}}$ is the reconstruction loss, which is taken as mean-squared error for numerical and cross-entropy for categorical features. The factor β balances the trade-off between reconstruction and promoting a Gaussian mixture distribution in the latent space. Once the VAE has been trained, we extract the class-wise latent embeddings $\boldsymbol{Z}_k = \{\boldsymbol{z}_k\}$, for $k \in \{0, 1\}$, corresponding to the training samples. To model the conditional distributions $q(\boldsymbol{z}|\boldsymbol{x}_k)$, we train score-based diffusion model $\mathcal{S}_k$ for each class $k \in \{0, 1\}$. These models follow the standard score-based forward and reverse diffusion processes (see Appendix for detailed formulations). The denoising neural network, implemented as a multi-layer perceptron (MLP) following the TabSyn architecture. After training, we can conditionally generate synthetic tabular data that follows the training data distribution by sampling through the reverse diffusion process as $\hat{\boldsymbol{x}} = \texttt{detok}(\texttt{Dec}(\texttt{Unflatten}(\hat{\boldsymbol{z}})))$, where $\hat{\boldsymbol{z}} \sim q(\boldsymbol{z}|\boldsymbol{x}_k)$ using diffusion model $\mathcal{S}_k$.

To generate OOD samples, we first sample N latent embeddings $\boldsymbol{z}_0 \sim q(\boldsymbol{z}|\boldsymbol{x}_0)$ and $\boldsymbol{z}_1 \sim q(\boldsymbol{z}|\boldsymbol{x}_1)$ from the diffusion models $\mathcal{S}_0$ and $\mathcal{S}_1$, respectively. Next, given an interpolation parameter α, we interpolate between these samples in the latent space as:

$$\tilde{\boldsymbol{z}} = \alpha\boldsymbol{z}_0 + (1 - \alpha)\boldsymbol{z}_1. \tag{7}$$

Finally, we obtain the OOD samples as $\hat{\boldsymbol{x}}_{\text{OOD}} = \texttt{detok}(\texttt{Dec}(\texttt{Unflatten}(\tilde{\boldsymbol{z}})))$.

Experimental Results and Insights

In this section, we evaluate the performance of our method under two scenarios: (i) *distribution shifts*, and (ii) *unseen types of samples during test time*. These scenarios are common in various domains, particularly in cybersecurity, where evolving attack patterns pose unique challenges.For instance, the attack landscape frequently evolves, leading to the emergence of new types of attacks. Our goal is to generate both (i) synthetic samples from the training data distribution (ii) OOD samples to augment the in-distribution samples. The combined set of synthetic data can be utilized in training or evaluation of downstream models. We consider the following datasets :

- **NSL-KDD (Tavallaee et al. 2009):** This addresses some of the limitations of the original KDDCup99 dataset for network intrusion detection, including the removal of duplicate and redundant records. Importantly, the test data includes specific attack types (14 new types of attacks) not present in the training data, simulating real-world scenarios where novel attacks emerge over time. We synthesize both in-distribution and OOD samples that differ from both normal and known attack samples.
- **AnoShift (Dragoi et al. 2023):** This anomaly detection benchmark dataset is built on the Kyoto-2006+ traffic dataset spanning from 2006 to 2015 and captures naturally occurring changes in network intrusion data over time. AnoShift provides data splits as IID (2006-2010), NEAR (2011-2013), and FAR (2014-2015) splits. We formulate two datasets by considering: (i) IID+NEAR split and (ii) IID+FAR split. For training, we use all normal and attack samples from the IID split along with only normal samples from the NEAR/FAR split, while the test set includes samples from both IID and NEAR/-FAR splits. Our OOD generation leverages historical attack information and normal behavior to create *attacks* that are out-of-distribution.
- **ACS Income:** This dataset is derived from the original dataset proposed by (Ding et al. 2021). The dataset enables systematic studies of distribution shifts across different states and years. We focus on the CA state data for two well-separated years, 2014 and 2018, to simulate a large distribution shift. The prediction task is to determine whether an individual's income exceeds $50,000, where the high-income group represents the minority class. Our training set includes all data from 2014 and only the majority samples from 2018, while the test set contains all samples, including minority class samples from 2018, simulating a distribution shift.

We evaluate the effectiveness of our method by employing the Machine Learning Efficiency (MLE) metric[2]. This evaluates how downstream model performance varies on real test samples, when trained using synthetic data in comparison to the same model being trained using real data. We establish baselines by training the downstream models on the real training data and evaluating them on the real test data, which includes distribution-shifted or new types of attack/minority

[2]See (Zhang et al. 2024a, Appendix E.4) for details on MLE

	NSL-KDD				Ano (IID+FAR)				Ano (IID+NEAR)				ACS Income			
	LR	DT	XGB	Avg	LR	DT	XGB	Avg	LR	DT	XGB	Avg	LR	DT	XGB	Avg
F1-score																
Base	73.5	81.5	80.3	**78.43**	81.1	85.4	86.5	**84.33**	88.4	96.6	97.4	**94.13**	61.9	65.0	64.4	**63.77**
Ours	81.6	85.9	84.1	**83.87**	88.2	88.2	88.2	**88.21**	96.3	97.0	97.4	**96.90**	71.9	71.4	73.0	**72.10**
Accuracy																
Base	75.6	81.8	80.9	**79.43**	75.4	80.5	81.9	**79.27**	85.1	95.3	96.4	**92.31**	76.8	77.2	78.1	**77.37**
Ours	81.1	84.8	82.9	**82.93**	83.1	83.2	83.6	**83.31**	94.6	95.8	96.4	**95.62**	79.5	78.1	80.0	**79.21**
Precision																
Base	96.5	96.6	97	**96.72**	97.8	99.1	99.3	**98.73**	98	98.9	99.2	**98.7**	83.1	78.3	84.1	**81.83**
Ours	91.7	91.1	89.5	**90.77**	93.9	95.2	97.3	**95.47**	94	96.7	96.6	**95.77**	75.1	73.4	75.1	**74.53**
Recall																
Base	59.4	70.4	68.6	**66.13**	69.3	75	76.7	**73.67**	80.4	94.4	95.8	**90.20**	49.3	55.6	52.3	**52.41**
Ours	73.4	81.3	79.2	**77.97**	83.2	82.1	80.7	**82.09**	98.7	97.3	98.3	**98.17**	68.9	70.7	71.1	**70.23**

Table 1: MLE performance comparison of baseline vs our synthetic data across different datasets and model families.

class samples. Particularly on NSL-KDD, certain types of attack categories are exclusively present only in the test set. We then compare these baselines to models trained on a new data which includes existing normal samples and synthetic minority class samples and tested on the same real test data. In our context, MLE measures the robustness of downstream models trained with the synthetic data (in-distribution synthesis augmented with OOD samples) when faced with distribution shifts or the presence of OOD samples during testing. Our synthetic datasets are generated to match the size of the original training data. Without loss of generality, we assign $k = 0$ to the normal class and $k = 1$ to the attack class in NSL-KDD and AnoShift datasets. For the ACS Income dataset, $k = 0$ represents majority (low-income) class, and $k = 1$ represents high-income class. The interpolation factor α governs the weight assigned to the majority (normal) class during data generation. In our experiments, we aim to enhance robustness to distribution shifts and unseen minority or attack samples by generating more OOD samples from the minority or attack classes. Hence, α is set to a smaller value to give more weight to the minority/anomaly class. Specifically, we set $\alpha = 0.4$ for the NSL-KDD dataset and $\alpha = 0.2$ for the other datasets. These values were chosen based on empirical insights rather than an extensive grid search, as further detailed in Appendix.

Enhanced Robustness of Downstream Models. We train three types of classifiers: Logistic Regression (LR), Decision Trees (DT), and XGBoost (XGB), each with a diverse set of hyperparameters. The model with the best performance for each classifier type is selected for comparison. Table 1 presents the gains in robustness achieved by our method on four datasets, measured in terms of F1-score and accuracy. While our method results in an average precision drop of approximately 5% on an average across the four datasets, it improves recall (by about 18%) and overall F1-score and accuracy by 6.9% and 3.9% respectively. We note that introduction of OOD samples in the boundary of the latent space generates more edge cases of benign samples which causes classifiers to identify them as anomaly and the increase in the false positive rate. However, the improvements in recall and F1-score suggest that our method improves the classifiers performance of detecting minority classes. We also perform an ablation study to establish the gains in robustness contributed by the OOD samples, and contrast the results against purely in-distribution generation of TabSyn or class rebalancing, (please see Appendix, Table 2), which shows the benefits of OOD generation in identifying unseen samples during inference.

Note that our primary focus is on evaluating the utility of our OOD generation method rather than on the quality of in-distribution samples. Since our in-distribution generation follows the TabSyn strategy with an additional novelty of using Gaussian mixture representations in the latent space, we rely on the extensive evaluations in the original TabSyn paper to validate the quality of in-distribution samples. TabSyn has already demonstrated that its synthetic data closely follows the training data distribution and preserves feature correlations. Hence, we omit detailed evaluation of in-distribution samples in this work.

Conclusion

We present TabOOD, a novel framework for generating OOD tabular data through a combination of latent space interpolation and diffusion techniques. While existing approaches have primarily focused on in-distribution data synthesis, TabOOD is, to our knowledge, the first method specifically designed to generate OOD samples—a critical capability for high-stakes applications. By generating synthetic OOD data, our work addresses the fundamental challenge of distribution shifts and enables models to better prepare for real-world scenarios where test distributions inevitably differ from training distributions.

Extensive evaluation across four datasets and three classifier families demonstrates the effectiveness of our approach. Models trained with TabOOD-generated data showed consistent performance improvements on unseen test samples, achieving average gains of 6.9% in F1-score and 3.9% in accuracy. These substantial improvements validate TabOOD's capability to enhance model robustness, making it a valuable tool for developing reliable systems in critical applications where handling distribution shifts is paramount.

References

Bergman, L.; and Hoshen, Y. 2020. Classification-Based Anomaly Detection for General Data. arXiv:2005.02359.

Borisov, V.; Seßler, K.; Leemann, T.; Pawelczyk, M.; and Kasneci, G. 2023. Language Models are Realistic Tabular Data Generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; and Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16: 321–357.

Ching, T.; Himmelstein, D. S.; Beaulieu-Jones, B. K.; Kalinin, A. A.; Do, B. T.; Way, G. P.; Ferrero, E.; Agapow, P.-M.; Zietz, M.; Hoffman, M. M.; et al. 2018. Opportunities and obstacles for deep learning in biology and medicine. *Journal of the royal society interface*, 15(141): 20170387.

Ding, F.; Hardt, M.; Miller, J.; and Schmidt, L. 2021. Retiring Adult: New Datasets for Fair Machine Learning. *Advances in Neural Information Processing Systems*, 34.

Dragoi, M.; Burceanu, E.; Haller, E.; Manolache, A.; and Brad, F. 2023. AnoShift: A Distribution Shift Benchmark for Unsupervised Anomaly Detection. arXiv:2206.15476.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 6840–6851.

Hong, E.; Yi, J.-S.; and Lee, D. 2024. CTGAN-Based Model to Mitigate Data Scarcity for Cost Estimation in Green Building Projects. *Journal of Management in Engineering*, 40(4): 04024024.

Huang, J.; Chai, J.; and Cho, S. 2020. Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, 14(1): 13.

Ishfaq, H.; Hoogi, A.; and Rubin, D. 2018. TVAE: Deep metric learning approach for variational autoencoder. In *Proc. ICLR Workshop. Vancouver, BC, Canada*.

Kim, J.; Kim, T.; and Choo, J. 2024. Group-wise Prompting for Synthetic Tabular Data Generation using Large Language Models. *arXiv preprint arXiv:2404.12404*.

Kim, J.; Lee, C.; and Park, N. 2022. Stasy: Score-based tabular data synthesis. *arXiv preprint arXiv:2210.04018*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kotelnikov, A.; Baranchuk, D.; Rubachev, I.; and Babenko, A. 2023. TabDDPM: Modelling tabular data with diffusion models. In *International Conference on Machine Learning (ICML)*, 17564–17579. PMLR.

Miotto, R.; Wang, F.; Wang, S.; Jiang, X.; and Dudley, J. T. 2018. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6): 1236–1246.

Ozbayoglu, A. M.; Gudelek, M. U.; and Sezer, O. B. 2020. Deep learning for financial applications: A survey. *Applied soft computing*, 93: 106384.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10684–10695.

Ruff, L.; Vandermeulen, R.; Goernitz, N.; Deecke, L.; Siddiqui, S. A.; Binder, A.; Müller, E.; and Kloft, M. 2018. Deep One-Class Classification. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 4393–4402. PMLR.

Tavallaee, M.; Bagheri, E.; Lu, W.; and Ghorbani, A. A. 2009. A Detailed Analysis of the KDD Cup 99 Data Set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, 1–6. IEEE.

Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; and Veeramachaneni, K. 2019. Modeling tabular data using conditional GAN. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, 7335–7345.

Xu, S.; Lee, C.-T.; Sharma, M.; Yousuf, R. B.; Muralidhar, N.; and Ramakrishnan, N. 2024. Are LLMs Naturally Good at Synthetic Tabular Data Generation? *arXiv preprint arXiv:2406.14541*.

Zhang, H.; Zhang, J.; Shen, Z.; Srinivasan, B.; Qin, X.; Faloutsos, C.; Rangwala, H.; and Karypis, G. 2024a. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. In *The Twelfth International Conference on Learning Representations*.

Zhang, H.; Zhang, J.; Shen, Z.; Srinivasan, B.; Qin, X.; Faloutsos, C.; Rangwala, H.; and Karypis, G. 2024b. Mixed-Type Tabular Data Synthesis with Score-based Diffusion in Latent Space. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zoppi, T.; Gazzini, S.; and Ceccarelli, A. 2024. Anomaly-based error and intrusion detection in tabular data: No DNN outperforms tree-based classifiers. *Future Generation Computer Systems*, 160: 951–965.

Appendix

TabOOD Algorithm

Algorithm 1: TabOOD: Controlled OOD Sample Generation

Input: Dataset $\mathcal{D}$ of N tabular data samples, each denoted as $\boldsymbol{x}$, VAE parameter β, interpolation parameter $\alpha \in [0, 1]$
Output: OOD samples $\{\hat{\boldsymbol{x}}_{\text{OOD},j}\}_{j=1}^{N'}$
Step 1: Learn Gaussian mixture latent representations using β-VAE
for each sample $\boldsymbol{x}$ in a batch of samples from $\mathcal{D}$ **do**
  $\boldsymbol{T} = \texttt{tok}(\boldsymbol{x}) \in \mathbb{R}^{M \times d}$ {Tokenize tabular input data }
  $\hat{\boldsymbol{z}} = \texttt{Flat}(\texttt{Enc}(\boldsymbol{T})) \in \mathbb{R}^{Md}$ {Latent embeddings at encoder output flattened}
  $\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{recon}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) + \beta D_{\text{KL}}$ {Learn Gaussian mixture latent space by minimizing this objective, with D_{KL} as in Eqn 5. β tuned to ensure best possible reconstruction while encouraging regularization during training.}
end for
Denote the class-conditional latent distributions, for class $k \in \{0, 1\}$, by $q(\boldsymbol{z}|\boldsymbol{x}_k)$.
Step 2: Train Score-Based Diffusion Models
for each class $k \in \{0, 1\}$ **do**
  Train score-based diffusion model $\mathcal{S}_k$ using latent embeddings $\{\boldsymbol{z}_k\}$ to model the conditional distribution $q(\boldsymbol{z}|\boldsymbol{x}_k)$.
end for
Step 3: Generate OOD Samples
Given the trained VAE components and diffusion models:
for each OOD sample generation **do**
  Sample $\boldsymbol{z}_0 \sim q(\boldsymbol{z}|\boldsymbol{x}_0)$ from diffusion model $\mathcal{S}_0$.
  Sample $\boldsymbol{z}_1 \sim q(\boldsymbol{z}|\boldsymbol{x}_1)$ from diffusion model $\mathcal{S}_1$.
  Interpolate latent embedding: $\tilde{\boldsymbol{z}} = \alpha \boldsymbol{z}_0 + (1 - \alpha)\boldsymbol{z}_1$
  Reconstruct OOD sample: $\hat{\boldsymbol{x}}_{\text{OOD}} = \texttt{detok}(\texttt{Dec}(\texttt{Unflatten}(\tilde{\boldsymbol{z}})))$
end for
Return: Generated OOD samples $\{\hat{\boldsymbol{x}}_{\text{OOD},j}\}_{j=1}^{N'}$

Insights on Choosing the Interpolation Parameter

For conciseness, we focus on the NSL-KDD dataset for insights and ablation studies due to space constraints. Our goal is to assess how different the unseen test samples (distribution-shifted anomalies or minority class samples for other datasets) could be, from previously seen attacks, to provide insights on when our generated OOD samples effectively enhance the performance of downstream models. Note that no information about these new test samples is utilized during training or sampling. Suppose a small proportion of the new attack samples becomes available post model training. These samples, denoted as $x_{\texttt{new}}$, can be used to inform our sampling and manipulation process. To guide the interpolation, consider the following steps:

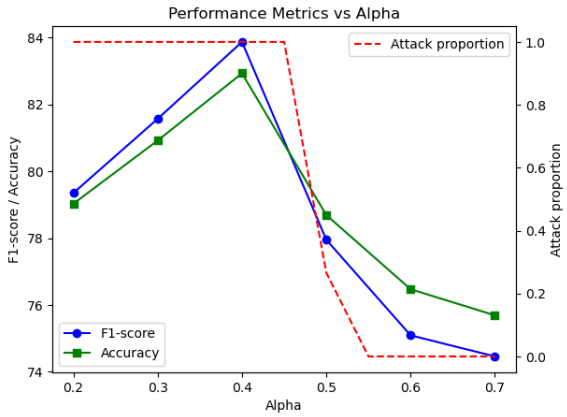


Figure 1: Effect of interpolation weight α on generated samples and downstream performance.

- Randomly sample 1% of the new attack samples, which amounts to approximately 40 samples in the NSL-KDD case.
- Compute the average pairwise ℓ_2 distance in the VAE latent space between the embeddings of new attack samples and the previously seen attack samples from the training set, denoted by as $d(A_{\texttt{new}}, A_{\texttt{train}})$, where A represents attack samples.
- Similarly, compute the average pairwise distance between the new attack samples and the normal samples from the training set, $d(A_{\texttt{new}}, N_{\texttt{train}})$, where N represents normal samples.
- Assign weights to the respective components based on the inverse ratio of these distances, i.e., $\alpha = d(A_{\texttt{new}}, A_{\text{train}})/d(A_{\texttt{new}}, N_{\texttt{train}})$.

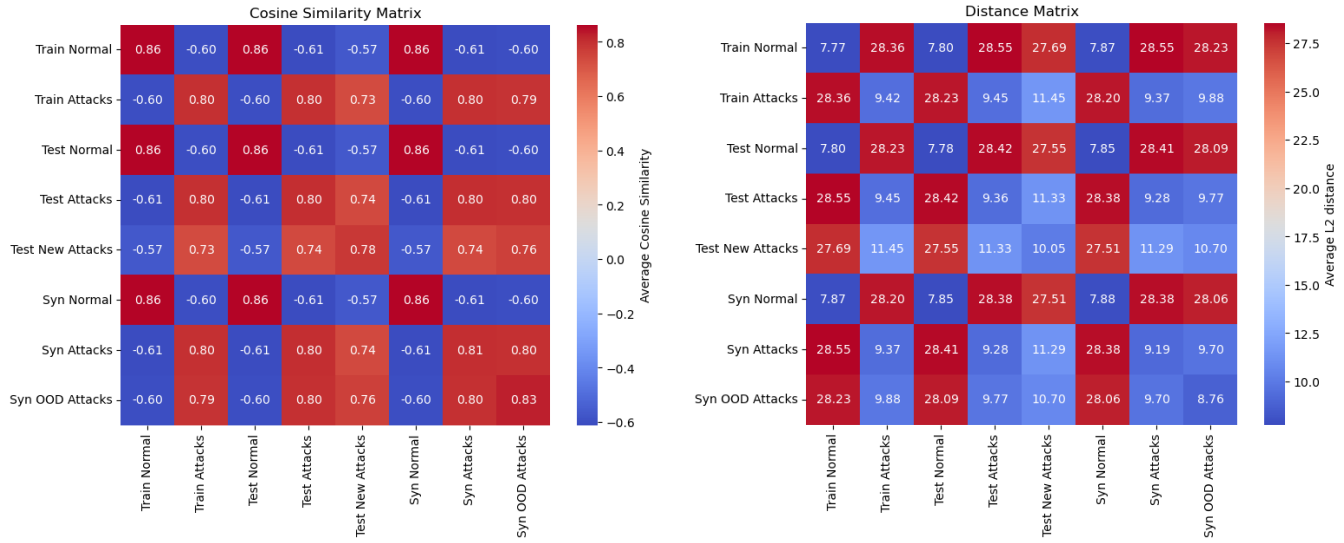
This naive approach yields $\alpha = 0.37$. We now experimentally study the effect of this manipulation weight α on the proportion of attack samples produced and its impact on the MLE performance. Note that in tabular synthesis, labels are treated as features, so when moving from latents to raw samples, both normal and attack samples are synthesized based on the interpolation. We observe in Fig. 1 that varying α results in a sharp change in the proportion of generated attack samples. Additionally, certain ranges of α lead to superior generalization. However, if the new attacks differ significantly from the previously seen ones, such that α becomes larger than a dataset-dependent threshold, our latent manipulations may not improve downstream performance.

Ablation Study

We perform an ablation on the NSL-KDD data to evaluate key factors driving the performance gains in our approach. Specifically, we want to demonstrate: (i) significant improvement comes from OOD samples in particular (ii) compare our in-distribution synthetic samples with those generated by TabSyn (iii) improvements are not simply due to rebalancing class proportions, as the training and test distributions differ. The test data remains the same across all experiments, with variations only in the training data. The

	F1-score					Accuracy				
	Base	Base-Rebal	TabSyn-Rebal	Ours-ID-Rebal	Ours	Base	Base-Rebal	TabSyn-Rebal	Ours-ID-Rebal	Ours
LR	73.5	76.8	73.1	77.4	81.6	75.6	77.7	75.3	77.7	81.1
DT	81.5	81.2	76.8	83.4	85.9	81.8	81.6	78.1	82.9	84.8
XGB	80.3	80.6	78.8	80.3	84.1	80.9	81.1	79.6	80.0	82.9
Avg	**78.43**	**79.53**	**76.23**	**80.37**	**83.87**	**79.43**	**80.13**	**77.67**	**80.21**	**82.93**
	Precision					Recall				
	Base	Base-Rebal	TabSyn-Rebal	Ours-ID-Rebal	Ours	Base	Base-Rebal	TabSyn-Rebal	Ours-ID-Rebal	Ours
LR	96.5	94.4	96.1	91.3	91.7	59.4	64.7	58.9	67.2	73.4
DT	96.6	96.8	96.7	93.1	91.1	70.4	69.9	63.7	75.5	81.3
XGB	97.0	97.0	96.8	91.3	89.5	68.6	69.1	66.4	71.8	79.2
Avg	**96.7**	**96.07**	**96.53**	**91.9**	**90.77**	**66.13**	**67.9**	**63.0**	**71.5**	**77.97**

Table 2: Performance comparison across different models on the NSL-KDD dataset.



(a) Average pairwise cosine similarity of embeddings.

(b) Average pairwise L2 distance of embeddings.

Figure 2: Comparison of pairwise similarities and distances between real and synthesized embeddings for test and synthesized attacks on the NSL-KDD dataset.

results in Table 2 demonstrate that our method, which includes OOD samples, improves generalization to unseen attack types. Base-Rebal refers to the real training samples, with class proportions rebalanced to match that in the test set. Similarly, TabSyn-Rebal and Ours-ID-Rebal refers to in-distribution training samples generated by TabSyn and our method respectively, but with the class proportions rebalanced.

Other evaluation metrics

We evaluate the latent embeddings generated by the encoder, comparing real samples with synthesized ones, focusing on the NSL-KDD dataset. We employ cosine similarity and L2 Euclidean distance as evaluation metrics. Specifically, we compute the average pairwise similarity and distance between embeddings, as shown in Figures 2a and 2b.

Note that "Test Attacks" refer to in-distribution attacks (attack types present in the training data), and "Syn Attacks" refer to synthesized attacks generated to resemble these in-distribution attacks.

We also evaluate the effectiveness of synthesized data using unsupervised anomaly detection methods. Specifically, we employ DeepSVDD (Ruff et al. 2018) and GOAD (Bergman and Hoshen 2020), leveraging their implementations in the DeepOD library. In these methods, only normal samples are used for training anomaly scoring models, which are then tested on a mix of normal and anomalous data. The models are expected to produce low anomaly scores for normal samples and high scores for anomalies.

We perform two types of evaluations using common metrics such as AUROC, Average Precision (AP), and F1-score. First, we establish a baseline by training the anomaly detection models on real normal samples and testing on real test samples. We then compare this performance with mod-

Model		Train: Real Normal, Test: Real			Train: Synth Normal, Test: Real			Train: Real Normal, Test: Synth		
Dataset	**Method**	**AUC**	**AP**	**F1**	**AUC**	**AP**	**F1**	**AUC**	**AP**	**F1**
NSL-KDD	DeepSVDD	0.77	0.85	0.76	0.73	0.82	0.73	0.81	0.89	0.78
	GOAD	0.92	0.91	0.87	0.90	0.87	0.86	0.94	0.93	0.89

Table 3: Unsupervised anomaly detection performance on the NSL-KDD dataset using DeepSVDD and GOAD models.

els trained on synthesized normal samples and tested on real data. The goal is to see if the synthetic data can replicate the performance of models trained on real data, indicating that the synthetic normal samples closely resemble the real normal ones.

Additionally, we evaluate whether models trained on real normal samples can detect attacks generated in our synthesized dataset. We expect these models to perform similarly, or even better, as the synthesized OOD attacks might be easier to detect than new, unseen real test attacks. Results are summarized in Table 3.

Cross-validation results on NSL-KDD

We conduct extensive cross-validation experiments comparing TabOOD with TabSYN using the NSL-KDD dataset. While other datasets like AnoShift and ACSIncome are available, their temporal and seasonal characteristics made them unsuitable for efficient cross-validation.

Our experimental setup involved creating 5 folds of train-test splits, carefully partitioning the attack samples to ensure that the test sets contained only out-of-distribution attacks. For each fold, we implemented the following procedure:

1. Trained VAE on the training data with Gaussian mixture modeling
2. Trained diffusion models using 5 different random seeds on the training data
3. Reversed the train-test split and repeated the process

This methodology generated 50 distinct comparison scenarios (5 folds $\times$ 5 random seeds $\times$ 2 train-test configurations). Once the models are trained, we sample appropriate benign and attack samples using the diffusion models to form the synthetic datasets. For TabOOD, we generated two types of samples: (i) in-distribution samples from the latent space comprising two Gaussian distributions; (ii) OOD samples using Equation equation 7 with $\alpha = 0.2$. For evaluation, we use the train dataset of each generated fold to train three families of classifier models (LR, DT, XGB) and assess the classification performance on the test dataset which only contains out-of-distribution attack samples. Next, we swapped the train and test datasets as suggested by cross validation procedure and trained the classifiers on test data and compute their classification performance on the training data. Note that this evaluation is distinct in flavor from the MLE evaluations reported earlier. Particularly, we experimentally evaluate whether incorporating our synthetic data generation approach into model training improves classifier detection performance compared to the existing TabSyn method.

We employed t-tests to statistically validate TabOOD's performance improvements over TabSyn across various classification methods. Our hypothesis testing framework examined whether TabOOD's precision was lower than TabSyn's, while for all other metrics, we tested whether TabOOD's performance was higher. Our model selection criteria varied by metric: for F1 scores and accuracy comparisons, we used models with the best respective scores, while for precision and recall evaluations, we considered models with the best F1 score, accuracy, and AUROC (the latter being particularly relevant for imbalanced problems). The statistical analysis results are presented in Table 4 with statistically significant results ($p < 0.05$) in bold. The F1 score showed significant improvement only for LR, while accuracy demonstrated significant improvements for both Logistic Regression and XGBoost. Notably, the Decision Tree Classifier showed no significant improvements across any metrics. Regarding recall, we observed significant improvements across all selected Logistic Regression models, though for XGBoost, significant improvement was limited to models with the best F1 score. In terms of precision, we found a statistically significant decrease only in LR with best AUROC, suggesting no strong evidence of precision degradation in most models when using TabOOD.

Performance metric	Model selection method	DecisionTreeClassifier	LogisticRegression	XGBClassifier
F1 Score	**Best F1 Score**	0.161	**0.001**	0.087
Accuracy	**Best Accuracy**	0.376	**0.001**	**0.016**
Precision	**Best AUROC**	0.871	**0.017**	0.155
	Best F1 Score	0.274	0.646	0.214
	Best Accuracy	0.730	0.672	0.233
Recall	**Best AUROC**	0.378	**0.006**	0.576
	Best F1 Score	0.274	**0.002**	**0.038**
	Best Accuracy	0.528	**0.004**	0.155

Table 4: The t-test p-value results for the cross validation on classification models.

Our comprehensive evaluation demonstrates that TabOOD offers significant improvements over TabSyn in detecting out-of-distribution samples, particularly when used with Logistic Regression and XGBoost classifiers. Statistical validation through t-tests confirms that TabOOD achieves enhanced performance across multiple metrics, notably in accuracy and recall, while maintaining competitive precision levels in most model configurations. The only observed trade-off was a minor precision decrease in a single model setup (Logistic Regression with best AUROC), suggesting that TabOOD successfully balances the generation of synthetic data with out-of-distribution detection capabilities. These results establish TabOOD as a robust framework for synthetic data generation in scenarios where out-of-distribution detection is crucial.