Diffusion and Flow Matching Models for Tabular Data: A Survey

Zhong Li, Qi Huang, Lincen Yang*, Jiayang Shi, Zhao Yang,
Niki van Stein (IEEE Senior Member), Thomas Bäck (IEEE Fellow), Matthijs van Leeuwen

arXiv:2502.17119v2 [cs.LG] 22 May 2026

***Abstract*—Deep generative models have made rapid progress in image, text, audio, and video generation, and are increasingly being applied to structured records. For tabular data, however, generative modeling remains difficult: a dataset may contain numerical and categorical attributes, missing values, sensitive fields, imbalanced categories, complex feature dependencies, and domain constraints. Earlier tabular data modeling methods based on GANs or VAEs have achieved useful results, but they can suffer from unstable training, mode collapse, weak modeling of multimodal distributions, and fragile handling of mixed-type features. Diffusion models have therefore attracted growing interest because their noising-and-denoising formulation provides a flexible and stable way to model complex data distributions, and has been adapted to tabular synthesis, missing-value imputation, trustworthy data generation, and anomaly detection. Flow matching offers a closely related route by learning transport vector fields along probability paths, often with more direct control over path design and sampling efficiency. Despite this progress, the literature on diffusion and flow matching models for tabular data remains difficult to compare because methods target different tasks and rely on different representations, objectives, evaluation protocols, and domain assumptions. To the best of our knowledge, this is the first survey dedicated specifically to diffusion and flow matching models for tabular data. We review work from June 2015 to May 2026, organize it around data-engineering challenges, tasks, design choices, and evaluation dimensions, and discuss open problems in scalability, feature dependency modeling, privacy, fairness, benchmarking, and constraint-aware generation. We maintain updates in a GitHub repository.**

***Index Terms*—Diffusion Models, Flow Matching, Tabular Data, Synthetic Data, Generative Models**

I. Introduction

Tabular data is a data modality in which information is organized into rows, representing individual records, and columns, representing features or attributes. It is ubiquitous in real-world domains such as healthcare [1], finance [2], education [3], transportation [4], and psychology [5]. The demand for high-quality generative models in these domains is acute due to data privacy regulations such as GDPR [6] and CCPA [7]. Consequently, real user data is often restricted from public release, whereas synthetic data generated by generative models can preserve machine learning utility while being more easily shared [8]. Beyond privacy concerns, real-world tabular datasets often contain missing values caused by human errors, survey non-response, data integration failures, or sensor malfunction. Generative models have therefore been employed for missing-value imputation [9]. In addition, class imbalance is common in tabular applications [10], and synthetic samples can be used to augment minority classes [11]. These applications underscore the growing importance of generative modeling for tabular data, ranging from privacy protection [12], [13], missing-value imputation [14], [15], to training data augmentation [16].

Deep generative models for tabular data include Energy-based Models (EBMs) [17], Variational Autoencoders (VAEs) [18], Generative Adversarial Networks (GANs) [19], Autoregressive Models [20], Normalizing Flows [21], Diffusion Models [22], and Flow Matching models [23]. Diffusion models have become especially influential because they avoid several common difficulties of earlier generative models: they are generally more stable to train than GANs, less prone to mode collapse, and better suited to representing complex multimodal distributions than simple latent-variable formulations. Flow matching shares the continuous-time generative modeling viewpoint, but learns transport vector fields along probability paths rather than reversing a fixed noising process. Together, diffusion and flow matching provide flexible tools for modeling complex data distributions, while leaving important design choices about representation, probability paths, sampling, privacy risk, and evaluation.

Motivated by their success in images [24], [25], audio [26], [27], text [28], [29], video [30], and graphs [31], recent studies have begun adapting diffusion and flow matching models to tabular data [32], [8], [33], [34], [35], [36], [37], [38], [39], [40], [41]. These works suggest clear potential, but tabular data also poses distinctive challenges: missing values, non-Gaussian and multimodal numerical distributions, heterogeneous and imbalanced categorical features, mixed-type columns, feature dependencies, limited data availability, privacy constraints, and domain-specific validity rules. As a result, models designed for images, text, or audio cannot be directly transferred to tabular data without careful changes in representation, objective design, conditioning, sampling, post-processing, and evaluation.

Numerous surveys have summarized diffusion models across domains [42], [43] or for specific modalities such as images [44], text [45], videos [30], time series [46], and graphs [31]. Surveys on tabular generative models [47], [48], [49] provide useful broader context across multiple model families, end-to-end synthesis pipelines, post-processing, applications, and general challenges, while recent benchmarks [50] highlight the importance of tuning budgets and fair evaluation. However, these works are broader in scope or benchmark-

*Lincen Yang is the corresponding author (l.yang@liacs.leidenuniv.nl). Zhong Li is with Great Bay University and Zhao Yang is with Vrije Universiteit Amsterdam. All other authors are with LIACS, Leiden University, the Netherlands.

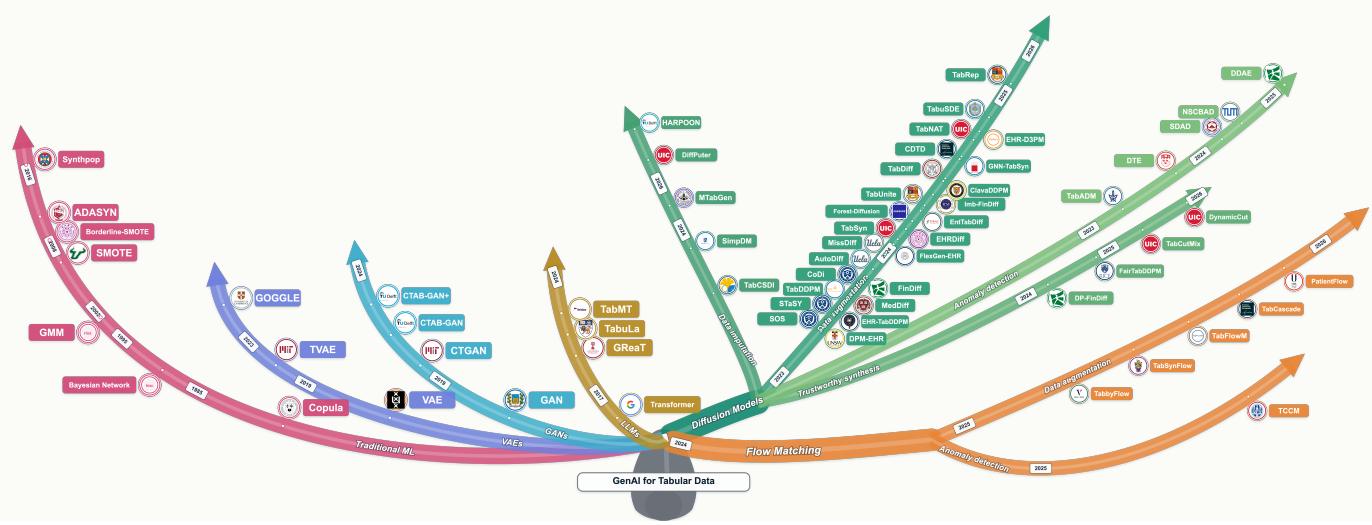


Fig. 1. Evolutionary timeline tree of generative models for tabular data. The figure traces six model families from a shared tabular-generation root, with branch-specific chronological markers and method nodes attached to curved branches. Since statistical, VAE-, GAN-, and LLM-based tabular generators serve only as historical context rather than the main focus of this survey, each of these families is represented by a few representative works. In contrast, diffusion and flow matching methods are expanded into task-oriented branches and annotated with first-author affiliation markers when available.

oriented rather than dedicated method surveys of diffusion and flow matching models for tabular data. They therefore provide limited analysis of representation choices, feature-type handling, noising processes, sampling cost, privacy and memorization, and task-specific evaluation across synthesis, imputation, trustworthy generation, and anomaly detection.

To address this gap, we present a comprehensive and critical survey of diffusion and flow matching models for tabular data. To the best of our knowledge, this is the first survey dedicated specifically to this topic. Our main contributions are as follows: 1) We review the historical development of generative models for tabular data and identify the key data-specific challenges that shape diffusion and flow matching methods (Sec. II); 2) We introduce the necessary preliminaries for diffusion and flow matching models from a tabular perspective, emphasizing the design choices that affect mixed-type data modeling (Sec. III); 3) We organize and review the literature by application tasks, including synthesis and augmentation, imputation, trustworthy data synthesis, and anomaly detection (Secs. IV–VII); and 4) We consolidate the lessons from these task-oriented reviews into a discussion of evaluation practice and future directions, including how to compare utility, fidelity, privacy, fairness, sampling cost, and constraint satisfaction; where flow matching has not yet been systematically studied; and what open problems remain in mixed-type representation, feature dependency modeling, relational structure, memorization, and benchmarking (Sec. VIII). Our coverage spans work from June 2015 to May 2026, with updates maintained in a GitHub repository[1].

II. Generative Models for Tabular Data

This section reviews the history of generative models for tabular data (see Sec. II-A), then discusses the unique characteristics of tabular data and the challenges these pose for generative models (see Sec. II-B), and finally outlines the application-driven taxonomy of diffusion and flow matching models for tabular data adopted in this survey (see Sec. II-C).

A. History of Generative Models for Tabular Data

As shown in Figure 1, prior to the advent of models explicitly designed for data generation, probabilistic models like Copula [51], Gaussian Mixture Models [52] and Bayesian Networks [53] were commonly employed for data synthesis. Later, specialized methods, including distance-based approaches such as SMOTE [54] and ADASYN [55], and probabilistic models like Synthpop [56], were introduced for data synthesis and imputation. However, distance-based methods, including SMOTE, encounter challenges when dealing with large datasets and complex data distributions. Additionally, probabilistic methods, such as Copulas and Synthpop, often struggle with heterogeneous data, impose predefined distributions, and are prone to assumption biases [47].

In contrast, deep generative models explicitly designed for data generation have gained increasing prominence in the tabular domain, offering significant advancements and more successful applications compared to traditional tabular data generation techniques. For example, VAE [57] based methods—such as TVAE [58] and GOGGLE [59]—were shown to achieve superior performance. Importantly, GOGGLE was the first to explicitly model the correlations among features, using a VAE-based model with Graph Neural Networks (GNN) as the encoder and decoder models. However, VAEs-based methods are prone to certain limitations, including blurry outputs in generated data due to the inherent randomness introduced by the latent space and potential difficulty in balancing reconstruction loss and regularization during training, which can affect the quality of the synthetic data. Additionally, VAEs may struggle with accurately capturing multimodal distributions, a common characteristic in real-world tabular datasets.

[1]https://github.com/Diffusion-Model-Leiden/awesome-diffusion-models-for-tabular-data

GAN [19] based methods such as CTGAN [58] have shown promising results for tabular data synthesis. These methods typically use Gaussian Mixture Models to model continuous features, which may be suboptimal for certain real-world data. Additionally, they handle categorical features using one-hot encoding, substantially increasing data dimensionality. Further, GANs suffer from limitations such as mode collapse, failing to capture the full data distribution, and training instability.

LLMs [60] have also been explored for tabular data synthesis. For example, GReaT [61] converts each row into a natural language sentence and learns sentence-level distributions using GPT [62]. This approach suffers from some limitations, including potential information loss during data-to-text conversion, increased dimensionality leading to higher computational costs, and context length limitations, which can affect scalability when handling large datasets.

Diffusion models [22] have demonstrated strong performance over VAEs and GANs on image synthesis tasks [63]. Recent studies, including but not limited to SOS [64], STaSy [32], TabDDPM [8], CoDi [34], and TabSyn [35], suggest that some of these advantages can also transfer to tabular data synthesis. Flow matching has recently become a closely related direction for tabular modeling. Existing studies use it for direct mixed-type synthesis, as in TabbyFlow [37]; latent-space synthesis, as in TabSynFlow [38] and TabFlowM [39]; cascaded coarse-to-fine generation for mixed-type features, as in TabCascade [40]; and longitudinal clinical data generation, as in PatientFlow [41].

B. Challenges with Generative Models for Tabular Data

Training generative models for tabular data is especially difficult due to the following seven data-specific challenges.

1) Missing Values: This phenomenon often happens in real-world tabular datasets for several reasons [65], e.g., privacy concerns (people's refusal to answer questions about their employment or income information in census data [66]), difficulty of data collection (drop-out in studies and merging unrelated data in healthcare data [67]), human operation errors when processing data [68], or machine error due to malfunctioning of equipment [68]. Values can be Missing At Random (MAR), Missing Completely At Random (MCAR), or Missing Not At Random (MNAR) [69]. Most generative models for tabular data cannot be directly trained on incomplete data [36].

2) Intricate Individual Feature Distribution: A feature in tabular data, i.e., a column, may have a 'complicated distribution' [58]: 2.1) for a numerical feature, the distribution can be non-Gaussian and/or it can have multiple distribution modes; 2.2) for a categorical feature, the class distribution can be highly imbalanced; or 2.3) within the same feature type (numerical or categorical), different features usually have different statistical properties (e.g., feature-wise marginal distribution) due to the fact that the meanings of features can be different.

3) Heterogeneous Features: Heterogeneous features refer to columns that contain different types of data, which has been considered the most challenging issue [33]. These types may include numerical, categorical, ordinal, boolean, text, or datetime data. This diversity makes modeling such data more complex than datasets consisting of only one feature type. It is unclear whether it is effective to combine different types of models, that were designed for different types of features.

4) Feature Dependencies (Correlations between Features): Tabular data synthesis models should learn the joint probability of multiple columns, but these columns are usually not independent of each other (i.e., their joint probability cannot be decomposed into the product of their marginal probabilities). Unlike image data, which contains only continuous pixel values with local spatial correlations, or text data, which comprises tokens sharing the same dictionary space [70], capturing correlations among features has been a long-standing challenge [33]. The heterogeneous nature of tabular data makes it even more challenging than capturing the correlations among purely categorical or numerical features.

5) Mixed-Type Feature: A mixed-type feature refers to a single feature that contains more than one type of data. Unlike heterogeneous features, which describe the diversity of feature types across columns, mixed-type features involve a single column containing values of different data types, such as numerical and categorical data mixed together.

6) Small Data Size: Compared to image or text datasets, which typically have massive amounts of available data, tabular datasets are usually smaller in volume. Specifically, the number of samples (or rows) is often limited, making it challenging to effectively train deep models.

7) Domain-specific Constraints: Real-world tabular datasets span various domains, such as healthcare, finance, and engineering, each with domain-specific features and constraints. For example, healthcare data must maintain realistic ranges for age or blood pressure, while finance data requires valid transaction amounts and balances. These constraints are often difficult to enforce or even verify.

C. Taxonomy of Diffusion and Flow Matching Models for Tabular Data

On a high level, generative models for tabular data are generally designed for one of four key tasks: data augmentation (see Sec. IV), data imputation (see Sec. V), trustworthy data synthesis (see Sec. VI), and anomaly detection (see Sec. VII). While this applies to tabular generative models in general, the focus of this survey is on diffusion and flow matching models. Therefore, we systematically examine diffusion- and, where available, flow-matching-based approaches for each task. This application-driven taxonomy was chosen instead of a classification based only on the underlying generative model types (see Sec. III) because it provides a more intuitive and practical perspective on how these models are applied to real-world problems. The taxonomy, summarized in Figure 2, allows us to systematically categorize and evaluate the literature on diffusion and flow matching models for tabular data.

Sec. IV reviews *data augmentation* methods that generate synthetic data, including single-table synthesis (see Sec. IV-A) and multi-relational synthesis (see Sec. IV-B). Then, Sec. V discusses *data imputation* approaches focused on handling missing values in tabular datasets. Next, Sec. VI explores *trustworthy data synthesis*, which includes privacy-preserving,

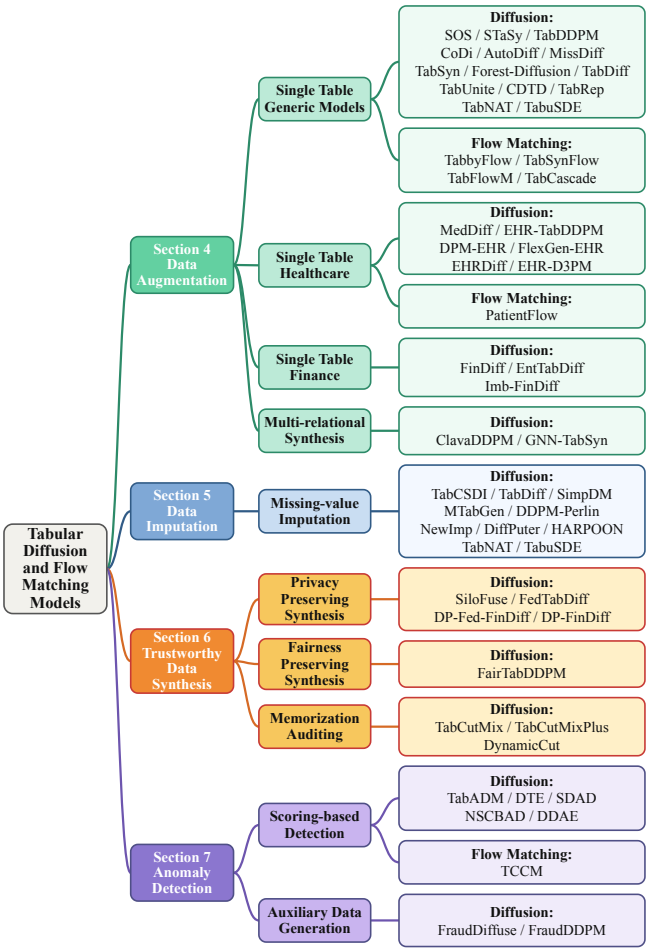


Fig. 2. Taxonomy of diffusion and flow matching models for tabular data. The taxonomy is organized by survey section and task category, grouping representative methods by application role rather than by publication time.

fairness-preserving, and memorization-aware data generation techniques. Finally, Sec. VII reviews *anomaly detection* methods that leverage diffusion or flow matching models to identify anomalies by learning the normal data distribution or a normal-data transport field [71]. The absence of dedicated flow-matching methods in some branches is itself informative: it identifies task settings where flow-matching methods for tabular data have not yet been systematically studied, rather than implying that diffusion and flow matching are equally represented in every section.

For completeness, we next introduce diffusion and flow matching models and the prominent model types used for tabular data in Sec. III. Readers familiar with these concepts may skip this section.

III. Diffusion and Flow Matching Models Preliminaries

This section first introduces the core mechanism of diffusion models, covering both the forward diffusion process (i.e., gradual noise addition) and the reverse process (i.e., learning to denoise). We then present prominent diffusion model types, including Gaussian diffusion models (DDPMs) (see Sec. III-1), multinomial diffusion models (see Sec. III-2), score-based generative models (SGMs) (see Sec. III-3), score-based generative models through stochastic differential equations (SDEs) (see Sec. III-4), and conditional diffusion models (see Sec. III-5). Finally, we introduce flow matching (see Sec. III-6), which replaces explicit denoising or score estimation with learning a time-dependent velocity field that transports samples from a base distribution to the data distribution. Throughout this survey, bold lowercase letters denote vectors, bold uppercase letters denote matrices, and task-specific notation is introduced locally where each problem is defined.

Diffusion probabilistic models, more commonly known as *diffusion models* [22], are deep generative models defined from a forward diffusion process and a reverse denoising process. First, the *diffusion process* aims to gradually 'corrupt' a sample $\mathbf{x}_0$ (drawn from training data distribution $q_{\text{data}}(\cdot)$) to a noisy instance $\mathbf{x}_T$ (from a prior distribution $q_{\text{noise}}(\cdot)$) using

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{1}$$

with $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ the forward transition probability. Second, the *denoising process* attempts to remove noise and generate a synthetic but realistic sample $\hat{\mathbf{x}}_0$ from $\mathbf{x}_T$ using

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t), \tag{2}$$

with $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ an approximation of the reverse of the forward transition probability, which is often learned by a neural network with parameters $\boldsymbol{\theta}$. Particularly, they learn $\boldsymbol{\theta}$ by minimizing the following variational upper bound (L_{VUB}) on the negative log-likelihood:

$$\begin{aligned} -\log\, p(\mathbf{x}) \leq\; & \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\underbrace{-\log p_{\boldsymbol{\theta}}(\mathbf{x}_0|\mathbf{x}_1)}_{L_0\,(L_{\text{reconstruction}})}] \\ & + \underbrace{D_{KL}[q(\mathbf{x}_T|\mathbf{x}_0)||p(\mathbf{x}_T)]}_{L_T\,(L_{\text{prior}})} \\ & + \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \underbrace{D_{KL}[q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)||p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)]}_{L_t\,(L_{\text{diffusion}})}. \end{aligned} \tag{3}$$

Here, L_0 can be interpreted as a reconstruction term that predicts the log probability of original sample $\mathbf{x}_0$ given the noised latent $\mathbf{x}_1$, while L_T measures how the final corrupted sample $\mathbf{x}_T$ resembles the noise prior distribution. Meanwhile, L_t measures how close the estimated $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is to the ground-truth posterior transition probability $q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)$.

The formulations above only define the generic structure of diffusion models: depending on the specific data type (namely continuous or discrete), the definitions of prior noise distribution $p(\mathbf{x}_T)$, forward transition probability $q(\mathbf{x}_t|\mathbf{x}_{t-1})$, the reverse of forward transition probability $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, and their training objectives can be different. In the following, we concisely review the five diffusion model frameworks most commonly used for tabular data modeling.

1) Gaussian Diffusion Models: Gaussian diffusion models, often known as Denoising Diffusion Probabilistic Models (DDPMs) [24], operate in continuous spaces. They define the diffusion and reverse processes as

$$p(\mathbf{x}_T) := \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}), \tag{4a}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}), \tag{4b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)), \tag{4c}$$

where Gaussian noises are gradually injected into the sample based on a time-dependent variance schedule $\{\beta_t\}_{t=1}^T$, with $\beta_t \in (0,1)$ determining the amount of noise added at time step t. To approximate $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, [24] define

$$\begin{aligned} \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\Big), \\ \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_t, t) &= \sigma_t\mathbf{I}, \end{aligned} \tag{5}$$

where σ_t controls the noise level added at time step t, $\alpha_t := 1-\beta_t, \bar{\alpha}_t := \prod_{i=1}^t \alpha_i$, and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is a neural network to predict ground truth noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ that has been added to noise sample $\mathbf{x}_t$. As a result, they propose to optimize the following simplified objective function (rather than Eq. 3):

$$L_{\text{simple}}^{\text{Gauss}}(\boldsymbol{\theta}) := \mathbb{E}_t\mathbb{E}_{\mathbf{x}_0\sim q(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\big[\lambda(t)\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\big], \tag{6}$$

where $\lambda(t)$ is a weighting function to adjust the noise scales, and $\|\cdot\|_2$ denotes the Euclidean norm.

2) Multinomial Diffusion Models: Multinomial diffusion models [28] [29] operate in discrete spaces, designed to generate categorical data $\mathbf{x}_t \in \{0,1\}^K$ (i.e., one-hot encoding with K distinct values). They are defined as follows:

$$p(\mathbf{x}_T) := \text{Cat}(\mathbf{x}_T; 1/K), \tag{7a}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \text{Cat}(\mathbf{x}_t; (1-\beta_t)\mathbf{x}_{t-1} + \beta_t/K), \tag{7b}$$

$$p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t) := \sum_{\hat{\mathbf{x}}_0=1}^{K} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{\mathbf{x}}_0)p_{\boldsymbol{\theta}}(\hat{\mathbf{x}}_0|\mathbf{x}_t), \tag{7c}$$

with $\text{Cat}(\cdot)$ categorical distribution and K the number of categories (the computation between scalars and vectors are done in an element-wise way). Importantly, *uniform noise* (rather than Gaussian noise) is added to the sample according to the noise schedule β_t. As we can see, $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is parameterized as $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \hat{\mathbf{x}}_0(\mathbf{x}_t, t))$, with $\hat{\mathbf{x}}_0(\mathbf{x}_t, t)$ predicted by a neural network, which can be trained via the multinomial diffusion loss defined using Eq. 3. Note that this model can handle only one categorical feature at a time. In other words, for a table with C categorical features, C separate multinomial diffusion models would need to be built.

3) Score-based Generative Models (SGMs): For these models, the forward diffusion process follows the same structure as the Gaussian diffusion model, whereas the reverse process is defined differently, as shown below. Given an instance $\mathbf{x}$ and its distribution $p(\mathbf{x})$, its score function is defined as $\nabla_{\mathbf{x}}\log p(\mathbf{x})$. To estimate the score function, one can train a neural network $S_{\boldsymbol{\theta}}(\cdot)$ with the following objective:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\|S_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}}\log p(\mathbf{x})\|_2^2. \tag{8}$$

However, Song & Ermon [72] point out that the estimated score functions are inevitably imprecise in low density regions when the low-dimensional manifolds are embedded into a high-dimensional space. To mitigate this, in the diffusion process they perturb the original data $\mathbf{x}$ with a sequence of random Gaussian noises with intensifying scales $0 < \sigma_1 < \cdots < \sigma_T$. In other words, $p_{\sigma_1} \approx p(\mathbf{x}_0)$, $p_{\sigma_T} \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$, and $p_{\sigma_t} \approx \mathcal{N}(\mathbf{x}_t; \mathbf{x}_0, \sigma_t^2\mathbf{I})$. In the reverse process, they utilize a noise-conditioned score network $S_{\boldsymbol{\theta}}(\cdot)$ to approximate $\nabla_{\mathbf{x}}\log p_{\sigma_t}(\mathbf{x})$, which analytically equals $(\mathbf{x}_0 - \mathbf{x}_t)/\sigma_t$. As a result, the training objective is as follows:

$$\frac{1}{T}\sum_{t=1}^{T}\lambda(\sigma_t)\mathbb{E}_{p(\mathbf{x}_0)}\mathbb{E}_{\mathbf{x}_t\sim p_{\sigma_t}(\mathbf{x}_t|\mathbf{x})}\left\|S_{\boldsymbol{\theta}}(\mathbf{x}_t, \sigma_t) + \left(\frac{\mathbf{x}_t - \mathbf{x}_0}{\sigma_t}\right)\right\|_2^2. \tag{9}$$

After training $S_{\boldsymbol{\theta}}(\cdot)$, new samples are generated with the annealed Langevin dynamics (see [72] for details).

4) Score-based Generative Models through Stochastic Differential Equations (SDEs) [25]: This is a continuous-time generalization of denoising diffusion models and score based generative models. Particularly, the *diffusion process* is defined with the following stochastic differential equation [73]:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \tag{10}$$

where $\mathbf{f}(\mathbf{x}, t) = f(t)\mathbf{x}$, and $f(\cdot)$, $g(\cdot)$ are referred to as the drift and diffusion coefficients of $\mathbf{x}_t$, respectively. Moreover, $\mathbf{w}$ is the standard Wiener process. The most widely studied and commonly used diffusion models can be broadly categorized into three main types: 1) Variance Exploding (VE), 2) Variance Preserving (VP), and 3) sub-Variance Preserving (sub-VP) based on the types of functions $f(\cdot)$ and $g(\cdot)$ as follows:

$$\mathbf{f}(\mathbf{x}, t) = \begin{cases} 0, & \text{if VE,} \\ -\frac{1}{2}\gamma_t\mathbf{x}, & \text{if VP,} \\ -\frac{1}{2}\gamma_t\mathbf{x}, & \text{if sub-VP,} \end{cases} \tag{11}$$

$$g(t) = \begin{cases} \sqrt{\frac{\mathrm{d}[\sigma_t^2]}{\mathrm{d}t}}, & \text{if VE,} \\ \sqrt{\gamma_t}, & \text{if VP,} \\ \sqrt{\gamma_t(1 - e^{-2\int_0^t \gamma_s \,\mathrm{d}s})}, & \text{if sub-VP,} \end{cases} \tag{12}$$

where γ_t and σ_t are noise functions w.r.t. the time variable t. Meanwhile, the *denoising process* is defined as the reverse of the diffusion process:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2\nabla_{\mathbf{x}}\log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}, \tag{13}$$

where $\bar{\mathbf{w}}$ is a Wiener process running backward in time, and the score function $\nabla_{\mathbf{x}}\log p_t(\mathbf{x})$ is approximated by a learnable neural network $S_{\boldsymbol{\theta}}(\mathbf{x}, t)$. However, directly approximating the score function is computationally intractable and thus they propose to train $S_{\boldsymbol{\theta}}(\cdot)$ by estimating the transition probability $\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x}_0)$ as follows [25]:

$$\begin{aligned} \arg\min_{\theta} \mathbb{E}_t\Big\{\lambda(t)\mathbb{E}_{\mathbf{x}_0}\Big[\mathbb{E}_{\mathbf{x}_t|\mathbf{x}_0}\Big[\\ \|S_{\boldsymbol{\theta}}(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x}_0)\|_2^2\Big]\Big]\Big\}, \end{aligned} \tag{14}$$

where $\nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\mathbf{x}_0)$ follows the Gaussian distribution and can be collected during the diffusion process. Moreover, $\lambda(t)$ is used to trade-off between sample quality and likelihood.

After training $S_{\boldsymbol{\theta}}(\cdot)$, we can generate new samples with either the *predictor-corrector* or the *probability flow* framework.

In general, the latter is preferred due to its fast sampling and exact log-probability computation compatibilities (see [25]).

5) Conditional Diffusion Models: The diffusion models introduced so far are all unconditional, meaning the posterior estimator function $p_{\boldsymbol{\theta}}(\cdot)$ does not know the label of the data it is modeling. Given a label vector $\mathbf{y}$, [22] and [63] suggest that this can be achieved through a so-called *conditional reverse process* in DDPM, defined as:

$$p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{y}) \propto p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)p_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x}_{t-1}), \tag{15}$$

which requires to train a classifier $p_{\boldsymbol{\phi}}(\cdot)$ and thus is known as *classifier-guided* DDPM. Dhariwa and Nichol [63] further approximate the logarithm of it with a perturbed Gaussian transition as follows:

$$\log(p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{y})) \approx \log(p(\mathbf{z})) + C, \tag{16}$$

where C is a constant, $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}+\boldsymbol{\Sigma}\mathbf{g},\boldsymbol{\Sigma})$, and $\mathbf{g} = \nabla_{\mathbf{x}_{t-1}} \log(p_{\boldsymbol{\phi}}(\mathbf{y}|\mathbf{x}_{t-1}))|_{\mathbf{x}_{t-1}=\boldsymbol{\mu}}$ is computed from the classifier $p_{\boldsymbol{\phi}}(\cdot)$. To avoid training a separate classifier, Ho & Salimans [74] propose a *classifier-free guided* DDPM:

$$\bar{\boldsymbol{\epsilon}}(\mathbf{x}_t,\mathbf{y},t) = \hat{\boldsymbol{\epsilon}}(\mathbf{x}_t,t) + \omega_g[\hat{\boldsymbol{\epsilon}}(\mathbf{x}_t,\mathbf{y},t) - \hat{\boldsymbol{\epsilon}}(\mathbf{x}_t,t)], \tag{17}$$

where $\hat{\boldsymbol{\epsilon}}(\cdot)$ is the DDPM noise estimator $\boldsymbol{\epsilon}_\theta(\cdot)$, $[\hat{\boldsymbol{\epsilon}}(\mathbf{x}_t,\mathbf{y},t) - \hat{\boldsymbol{\epsilon}}(\mathbf{x}_t,t)]$ is guidance of $\mathbf{y}$, and ω_g the guidance weight.

6) Flow Matching Models: Flow matching [23] is a continuous-time generative modeling framework that learns a velocity field rather than a denoising network or score function. Let p_0 be a simple base distribution, such as a standard Gaussian, and let $p_1 = p_{\text{data}}$ be the target data distribution. A time-dependent flow map ϕ_t transports samples from p_0 to p_1 according to the ordinary differential equation

$$\frac{d}{dt}\phi_t(\mathbf{x}) = \mathbf{v}_{\boldsymbol{\theta}}(\phi_t(\mathbf{x}),t), \quad \phi_0(\mathbf{x}) = \mathbf{x}. \tag{18}$$

Training can be formulated as velocity regression along a chosen probability path. For example, with a linear interpolation between a noise sample $\mathbf{x}_0 \sim p_0$ and a data sample $\mathbf{x}_1 \sim p_1$, one can define

$$\mathbf{x}_t = (1-t)\mathbf{x}_0 + t\mathbf{x}_1, \quad \mathbf{u}_t(\mathbf{x}_t) = \mathbf{x}_1 - \mathbf{x}_0, \tag{19}$$

and optimize

$$\mathcal{L}_{\text{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1}\left[\|\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t,t) - \mathbf{u}_t(\mathbf{x}_t)\|_2^2\right]. \tag{20}$$

Unlike diffusion models, which typically specify a forward noising process and learn to reverse it through denoising or score estimation, flow matching directly learns a velocity field defined along a probability path. This distinction is important for tabular data because the choice of probability path, representation space, sampling procedure, and feature-type parameterization can affect mixed-type feature handling, sampling cost, privacy risk, and the preservation of cross-feature dependencies. We therefore use these axes to compare diffusion and flow matching methods in later sections, rather than treating flow matching only as a faster sampler.

Figure 3 translates the mathematical preliminaries above into a compact reading map for the remainder of this survey. Diffusion models differ in whether they use separate mixed-type corruption processes, latent or unified continuous representations, or feature-adaptive continuous-time dynamics. Flow matching models, in turn, differ in whether the velocity field is learned in observed data space, a latent/token space, a cascaded coarse-to-fine space, or a task-specific space. The shared middle column makes these routes directly comparable: diffusion typically learns a denoiser, score, or reverse kernel such as $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)$, $S_{\boldsymbol{\theta}}(\mathbf{x}_t,t)$, or $p_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}|\mathbf{x}_t)$, whereas flow matching learns a velocity field $\mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_t,t)$ and generates by integrating a flow map ϕ_t as in Eq. 18. This comparison is not merely about sampling speed; it also affects how methods represent mixed numerical–categorical states, choose probability paths or noising schedules, balance feature-type losses, and expose utility–privacy–cost tradeoffs. Accordingly, the following sections evaluate methods by both application task and these shared design axes across data augmentation (see Sec. IV), data imputation (Sec. V), trustworthy data synthesis (Sec. VI), and anomaly detection (Sec. VII), rather than treating all synthetic data generators as interchangeable.

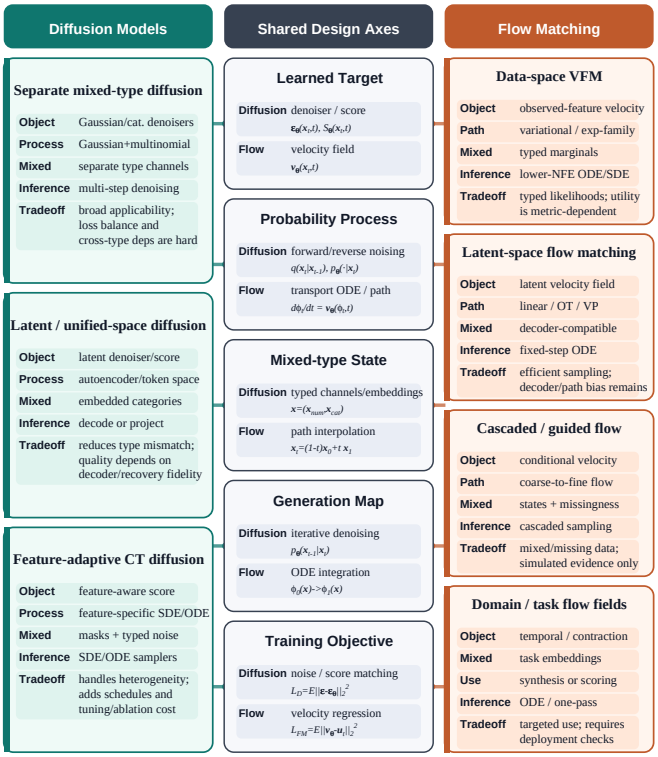


Fig. 3. Design-axis map for tabular diffusion and flow matching models. The left and right columns summarize representative diffusion and flow-matching routes, respectively, while the middle column aligns them through shared mathematical axes: learned target, probability process, mixed-type state, generation map, and training objective. The connectors emphasize that diffusion and flow matching make different modeling choices along comparable tabular design dimensions.

IV. Diffusion and Flow Matching Models for Data Augmentation

Data augmentation for tabular data is a long-standing research problem [75]. In general, it can be divided into two different tasks: 1) *data synthesis*, which is the process

of generating synthetic data that mimics the characteristics of real-world data, and can be further divided into *single table synthesis* and *multi-relational dataset synthesis*; and 2) *over-sampling*, which balances an imbalanced table by increasing the number of samples in the minority class(es). Particularly, over-sampling can be considered as a special case of *single table synthesis* where we only generate a part of the table. Accordingly, in this survey, we categorize relevant works into *diffusion and flow matching models for single-table synthesis* (including over-sampling) and *diffusion models for multi-relational dataset synthesis* , providing their formal definitions and reviewing related studies in Secs. IV-A and IV-B, respectively.

A. Diffusion/Flow Matching Models for Single Table Synthesis

This subsection reviews works on *diffusion and flow matching models for single table synthesis* (including over-sampling). For clarity, we subdivide related works into three sub-categories: 1) generic diffusion and flow matching models for tabular data (see Sec. IV-A1), 2) diffusion and flow matching models for tabular data in the healthcare domain (see Sec. IV-A2), and 3) diffusion models for tabular data in the finance domain (see Sec. IV-A3). To ensure readers can follow along, we present the task-specific setup and problem definition below (see Problem 1) and provide a summary of the models in Table I. Within each sub-category, we further separate diffusion-based and flow-matching-based methods when both are present.

Given a table R, we utilize ${}_i^j\mathbf{x}_t$ to denote the j-th feature value of the i-th sample at time point t. On this basis, let the number of numerical features be M_{num} and the number of categorical features be M_{cat}. By reorganizing the order of the features, a sample (i.e., a row) can be represented as $\mathbf{x} = (\mathbf{x}^{\text{num}}, \mathbf{x}^{\text{cat}})$, where $\mathbf{x}^{\text{num}} \in \mathbb{R}^{M_{\text{num}}}$ and $\mathbf{x}^{\text{cat}} \in \mathbb{Z}^{M_{\text{cat}}}$. The j-th categorical feature can have C_j distinct feature values $\{1, \cdots, C_j\}$, namely ${}^j\mathbf{x}^{\text{cat}} \in \{1, \cdots, C_j\}$. With this, the *single table synthesis problem* can be defined as:

Problem 1 (Single Table Synthesis)**.** *Given a table $R = \{\mathbf{x}\}$, we aim to learn a generative model $p_{\boldsymbol{\theta}}(R)$ to generate a synthetic table $\hat{R} = \{\hat{\boldsymbol{x}}\}$ with high-quality and diversity.*

1) Generic Diffusion and Flow Matching Models for Single Table Synthesis: We use "generic" to refer to methods intended as broadly applicable single-table generators, rather than models tied to a specific application domain. This does not imply that one model works uniformly well on every dataset; instead, these methods are formulated before adding assumptions such as EHR temporal structure or financial transaction rules. The main questions are therefore representational and procedural: how to encode numerical and categorical columns, whether to generate in data space or latent space, how to preserve dependencies and handle missing entries, and how flow matching changes probability paths and sampling cost. Following Table I and the taxonomy in Figure 2, we first discuss diffusion-based methods, then flow-matching-based methods, and finally compare their shared design choices and limitations.

TABLE I
OVERVIEW OF DIFFUSION AND FLOW MATCHING MODELS FOR SINGLE TABLE SYNTHESIS. '*' REFERS TO A NAME CREATED BY US FOR CONVENIENCE.

Paradigm	Model Name	Year	Venue	Feature Type	Domain
Diffusion	**SOS** [64]	2022	KDD	Num by Default	Generic
	STaSy [32]	2023	ICLR	Num+Cat	Generic
	TabDDPM [8]	2023	ICML	Num+Cat	Generic
	CoDi [34]	2023	ICML	Num+Cat	Generic
	MissDiff [76]	2023	ICMLW	Num+Cat	Generic
	AutoDiff [33]	2023	NeurIPSW	Num+Cat+Mixed	Generic
	TabSyn [35]	2024	ICLR	Num+Cat	Generic
	Forest-Diffusion [36]	2024	AISTATS	Num+Cat	Generic
	TabDiff [70]	2025	ICLR	Num+Cat	Generic
	TabUnite [77]	2024	OpenReview	Num+Cat	Generic
	CDTD [78]	2025	ICLR	Num+Cat	Generic
	TabRep [79]	2026	TMLR	Num+Cat	Generic
	TabNAT [80]	2025	ICML	Num+Cat	Generic
	TabuSDE [81]	2025	ICDMW	Num+Cat	Generic
Flow Matching	**TabbyFlow** [37]	2025	ICML	Num+Cat	Generic
	TabSynFlow [38]	2026	TMLR	Num+Cat	Generic
	TabFlowM [39]	2026	TMLR Sub.	Num+Cat	Generic
	TabCascade [40]	2026	ArXiv	Num+Cat+Mixed	Generic
Diffusion	**MedDiff** [82]	2023	ArXiv	Num by default	Healthcare
	EHR-TabDDPM [83]	2023	ArXiv	Num+Cat	Healthcare
	DPM-EHR* [84]	2023	NeurIPSW	Num+Cat	Healthcare
	FlexGen-EHR [85]	2024	ICLR	Cat+TS	Healthcare
	EHRDiff [86]	2024	TMLR	Cat+Num+TS	Healthcare
	EHR-D3PM [87]	2025	TMLR	Cat	Healthcare
Flow Matching	**PatientFlow** [41]	2026	AI Med.	Num+Cat+TS	Healthcare
Diffusion	**FinDiff** [88]	2023	ICAIF	Num+Cat	Finance
	EntTabDiff [89]	2024	ICAIF	Num+Cat	Finance
	Imb-FinDiff [90]	2024	ICAIF	Num+Cat	Finance

a) Diffusion-based generic synthesis: **SOS** [64] is the first to apply score-based generative models to tabular data oversampling. It trains class-specific score networks and generates minority-class samples either from random noise or from noised majority-class samples through a minority-class reverse process. This makes SOS mainly an oversampling method, but it shows that score-based generation can be useful for tabular data. Moving from oversampling to general data synthesis, **STaSy** [32] proposes a score-based tabular synthesis method based on stochastic differential equations. It improves score learning with self-paced training and fine-tunes the pre-trained score network with a probability-flow ODE. However, STaSy still converts categorical features into one-hot representations and rounds them back after generation, which may introduce invalid categories and weaken dependency preservation.

Unlike STaSy, which follows a continuous diffusion framework, **TabDDPM** [8] adapts diffusion models to mixed-type tabular data by modeling numerical and categorical features separately. It uses Gaussian diffusion for numerical features and multinomial diffusion for categorical features, which makes it a simple and influential baseline for generating mixed-type tables. However, because the two feature types are mostly modeled through separate corruption processes, cross-type dependency modeling remains a challenge. To better capture correlations between different feature types, **CoDi** [34] proposes a co-evolving diffusion framework with two coupled diffusion models: one for numerical attributes and the other for categorical attributes. Each reverse process is conditioned on the current noisy representation of the other feature type, and contrastive learning is used to align numerical and categorical representations. CoDi therefore improves dependency modeling over a purely separate design, but it still depends on preprocessing and postprocessing choices such as scaling, one-hot encoding, and rounding.

After TabDDPM and CoDi, a natural question is whether

tabular diffusion models can better handle practical issues beyond the basic numerical/categorical split. **AutoDiff** [33] addresses *heterogeneous features* by integrating an autoencoder with a score-based diffusion model. The autoencoder maps heterogeneous features into a continuous latent space, where diffusion is performed, and the decoder reconstructs samples in the original feature space. To handle *mixed-type features*, AutoDiff introduces a dummy variable encoding the frequency of repeated values, and it learns *feature correlations* through the joint distribution of latent embeddings. However, its performance and privacy depend strongly on the autoencoder: the authors observe weaker privacy guarantees, partly because a sophisticated autoencoder may overfit the input data and diffusion models may memorize training samples [91].

Vanilla diffusion models typically require complete training data, while tabular data often suffers from missing values. Ouyang et al. [76] observe that an "impute-then-generate" pipeline may introduce learning bias because single imputation cannot capture data variability. To address this issue, **MissDiff** trains diffusion models directly on incomplete mixed-type tables without prior imputation. Its main idea is to mask the denoising score-matching loss over missing entries, while using simple placeholders for missing numerical and categorical values so that the score function remains well-defined. This makes MissDiff an early tabular diffusion model for learning from incomplete data, but it still follows common preprocessing and postprocessing choices such as min-max scaling, one-hot encoding, softmax, and rounding, so categorical validity and dependency preservation remain concerns.

Following the same motivation of avoiding direct mixed-type diffusion, **TabSyn** [35] explores latent diffusion for tabular data synthesis, although AutoDiff [33] addressed this idea earlier. The authors emphasize that simple encoding strategies, such as one-hot encoding and analog bit encoding, often lead to suboptimal performance, while employing separate models for different feature types hampers the ability to capture cross-feature correlations. To overcome these limitations, TabSyn uses a VAE with feature tokenizers to transform raw numerical and categorical features into continuous embeddings, trains a score-based diffusion model in the latent space, and then uses a detokenizer to recover real feature values from the generated embeddings. This joint latent space enables the model to handle different feature types while preserving their correlations. Ablation studies further show that replacing the VAE with simple one-hot encoding gives the poorest performance, indicating that treating categorical features as continuous without learned representations is inadequate. However, TabSyn cannot directly handle missing values during training; missing-value processing and feature transformations must be done in preprocessing, which may introduce bias or noise.

Rather than relying on neural networks as universal function approximators [92] to estimate the score or noise function, **Forest-Diffusion** [36] employs XGBoost [93], leveraging its strength in tabular data prediction and classification. The method duplicates the training data across discretized noise levels, adds noise or interpolates between data and noise, and trains XGBoost models to predict the score at different time points. An important advantage is that XGBoost can naturally handle missing data by learning the best split, allowing Forest-Diffusion to be trained directly on incomplete datasets. For categorical data, it still uses dummy encoding during preprocessing and rounds dummy variables to the nearest class during postprocessing. Forest-Diffusion can also be adapted to data imputation and conditional synthesis, but its main message is more general: tabular diffusion models do not have to rely only on neural denoisers, and strong tabular learners can be inserted into the diffusion framework. At the same time, categorical postprocessing and the need to train models over many noise levels remain practical concerns.

After latent-space and tree-based alternatives, another line of work revisits how mixed features should be represented and noised within diffusion itself. **TabDiff** [70] identifies several limitations in prior methods, including the encoding overhead of latent diffusion models, the imperfect handling of discrete-time diffusion in mixed-type tables, and the insufficient treatment of feature-wise distribution heterogeneity. To address these issues, TabDiff adopts a continuous-time diffusion framework with learnable noise schedules for individual features. Numerical features are modeled with an SDE-based diffusion process, while categorical features are handled with masking diffusion. Although the paper describes this as a joint diffusion process, the numerical and categorical parts are still modeled differently; its main contribution is therefore feature-adaptive noise scheduling rather than a fully unified mixed-type diffusion process. **TabUnite** [77] pushes the representation question further. It argues that separate generative processes and simple encodings such as one-hot representations can weaken cross-feature correlation modeling, while learned latent embeddings may be parameter inefficient. It therefore uses a Quantile Transformer for continuous features and compact encodings such as analog bits, PSK encoding, or dictionary encoding for categorical features, then applies a single diffusion or flow model in the unified data space. Together, these works show that mixed-type tabular generation depends not only on the denoising model, but also on how feature types are encoded and how noise schedules are assigned across heterogeneous columns.

Along the same line, **CDTD** [78] points out that existing diffusion models for numerical and categorical features are mostly built upon advances from image or text domains, so their noise schedules are not specifically designed for mixed tabular features. This may lead to unbalanced losses across feature types, poor scalability, and weak performance on categorical features with many categories. CDTD addresses these challenges by embedding categorical features into a continuous space and applying a unified Gaussian diffusion process. It further explores learnable noise schedules at different granularities, including a single schedule for all features, separate schedules for numerical and categorical features, and individual schedules per feature. This design improves mixed-feature handling and high-cardinality categorical modeling, but it also shows that schedule design and loss balancing remain central issues for tabular diffusion.

TabRep [79] further emphasizes that representation design is not merely a preprocessing detail. Instead of using sparse one-hot encodings or an additional latent model, TabRep in-

troduces CatConverter, which maps each categorical value to a two-dimensional cosine-sine representation on a unit circle and concatenates it with quantile-transformed numerical features. A DDPM or flow matching model can then be trained in this unified continuous space, and generated categorical values are recovered by projecting the output to the nearest valid category. This compact representation avoids some drawbacks of sparse categorical encodings, but it also introduces a new assumption: categorical values must be ordered before placement on the unit circle. Different orderings can affect performance, suggesting that future tabular diffusion models may need semantically informed or learnable categorical representations.

Recent work also tries to combine diffusion with more structured dependency modeling. **TabNAT** [80] approaches mixed-type tabular synthesis by combining masked generative modeling with diffusion. Its motivation is that autoregressive models can explicitly model column dependencies but struggle with continuous values and fixed generation orders, while diffusion models handle continuous variables naturally but may underuse column dependency structure. TabNAT embeds continuous features and discrete columns into a shared sequence, randomly masks target positions, and uses a bidirectional Transformer to produce conditional embeddings. Discrete columns are generated with categorical prediction, while continuous columns are generated with a conditional diffusion model. This design supports order-agnostic generation and can also be used for missing-value imputation, although its imputation setting assumes complete training data and evaluates artificially injected missing entries.

TabuSDE [81] takes a more system-level route for joint tabular synthesis and completion. It uses learnable per-feature drift and diffusion coefficients to adapt SDEs to heterogeneous columns, combines categorical embeddings with transformations for numerical variables, and uses cross-feature attention to capture inter-column dependencies. It also predicts feature values and missingness masks, so that structured missing-data patterns can be modeled during generation. The framework further includes privacy accounting and accelerated sampling, but these many components make the source of performance gains harder to isolate. Therefore, TabuSDE is best viewed as a broad design that combines feature-specific SDEs, missingness modeling, attention, privacy accounting, and sampling acceleration, rather than as evidence that any single component alone solves tabular synthesis and completion.

b) Flow-matching-based generic synthesis: Flow matching extends this discussion by replacing denoising-based generation with learned transport paths. **TabbyFlow** [37] is a dedicated flow-matching framework for mixed-type tabular synthesis. Instead of forcing all columns into a single Gaussian score-matching objective, it uses exponential-family variational posteriors so that numerical variables and categorical variables can be handled with feature-type-specific losses under a common variational flow matching view. This makes TabbyFlow important not only as a new generator, but also as an example of how flow matching can be adapted to heterogeneous tabular likelihoods.

Beyond direct data-space flow matching, another question is how flow matching should be used when the representation or feature type is more specialized. **TabFlowM** [39] studies whether a lightweight latent-space flow-matching generator can replace diffusion-style score learning after mixed-type records are mapped into a decoder-compatible continuous token space. It trains a time-conditioned velocity field on a simple coupling path and samples by integrating the learned ODE. This makes TabFlowM useful evidence that latent flow matching can reduce training or sampling cost, while also showing that performance still depends on the quality of the learned latent representation.

The same representation issue becomes sharper when numerical columns contain discrete states rather than ordinary continuous values. In many tables, missing entries, censored observations, or zero-inflated outcomes behave like special states, while the remaining values still require continuous modeling. **TabCascade** [40] addresses this setting with a coarse-to-fine flow-matching pipeline: it first generates categorical features and discretized numerical states at low resolution, and then conditions a high-resolution flow model on this coarse output to recover numerical details. This design is useful for difficult mixed-type features, although its missingness setting is simulated and it does not provide formal privacy guarantees.

A related question is how much of flow matching's behavior comes from the transport formulation itself, and how much comes from the representation on which it is applied. **TabSynFlow** [38] studies this question by replacing TabSyn's latent diffusion model with a flow-matching velocity field, and comparing this latent-space design with data-space variants such as TabbyFlow under different paths and samplers. Its results suggest that flow matching can often reach competitive utility with fewer function evaluations, but privacy risk remains a post-hoc quantity that depends on the dataset, representation, and path choice.

c) Comparative Analysis: Overall, generic single-table methods are best compared along three design axes. The first is representation. STaSy treats categorical variables through one-hot encodings in a continuous score/SDE framework, while TabDDPM and CoDi separate numerical and categorical corruption channels. AutoDiff, TabSyn, TabUnite, and TabRep instead map heterogeneous columns into latent or unified continuous spaces before applying diffusion or flow matching. This makes the generative dynamics easier to define, but shifts part of the burden to encoding and decoding: category validity, rare values, and semantic distances depend on the representation. The second axis is where tabular heterogeneity is handled. TabSyn and TabRep mainly adapt the feature space to standard diffusion or flow matching, whereas TabDiff, CDTD, and TabuSDE adapt the process itself through schedules, losses, masks, or feature-wise dynamics. This process-level direction is useful because tabular columns have unequal scales, cardinalities, missingness patterns, and marginal distributions. However, it also adds hyperparameters and makes attribution harder: when performance improves, the gain may come from the schedule, backbone, missingness model, representation, or tuning budget. The third axis is path design and efficiency. Flow matching expands the choices of probability path and sampler: TabbyFlow uses feature-type-aware variational objectives for mixed variables, while TabFlowM and TabCascade

use latent or cascaded transport to reduce sampling cost or handle difficult mixed-type features. These results suggest that faster generation is possible, but efficiency is not a proxy for quality or safety. Conclusions still depend on downstream utility, fidelity, disclosure risk, rare-event behavior, and tuning protocol. Thus, the practical question is not simply whether diffusion or flow matching models can synthesize tables, but which representation, generative process, and evaluation protocol fit the target application.

2) Diffusion and Flow Matching Models for Single Table Synthesis in Healthcare Domain: Here, we review diffusion and flow matching models for single-table synthesis in the healthcare domain, with a primary focus on electronic health records (EHRs). EHR data often combines demographics, diagnoses, medications, laboratory measurements, and treatment outcomes, making it valuable for medical research but difficult to share directly because of privacy risks [94]. Synthetic EHR generation therefore aims to preserve useful clinical patterns while reducing the need to release patient records [87].

a) Diffusion-based healthcare synthesis: Early diffusion-based EHR generators mainly adapt general tabular diffusion ideas to clinical data. **MedDiff** [82] uses a DDIM-style generator with accelerated sampling and classifier guidance for conditional generation, but it is limited to continuous features. **EHR-TabDDPM** [83] instead applies TabDDPM [8] to mixed-type EHR tables, making it a more direct baseline for numerical and categorical clinical attributes. Both works evaluate synthetic quality through fidelity and downstream utility, with EHR-TabDDPM also considering privacy.

After early static EHR generators, later work turns to longitudinal and heterogeneous clinical records. **DPM-EHR** [84] extends diffusion-based synthesis to longitudinal mixed-type EHRs, using one-hot categorical representations together with Gaussian diffusion. This makes it an early attempt to move beyond static EHR tables, although discrete medical codes are still treated through a continuous representation. **FlexGen-EHR** [85] further considers a more realistic setting where static and temporal modalities may be missing not completely at random. It encodes temporal and static features separately, aligns their latent spaces with an optimal-transport module, applies latent diffusion, and decodes both modalities back to EHR records. This design is better matched to heterogeneous hospital data, but the added encoders, alignment module, and decoders also make it harder to isolate which component drives the improvement.

Another design choice is whether medical codes should be converted into a continuous space or modeled as discrete variables. **EHRDiff** [86] uses a real-valued representation for both categorical and continuous EHR features and applies an SDE-based diffusion model with probability-flow ODE sampling. This gives a simple unconditional generator, but the meaning of categorical codes still depends on the chosen encoding and normalization. **EHR-D3PM** [87] takes the opposite route by modeling categorical medical codes directly with a D3PM-style discrete diffusion process [29] and adding training-free classifier/energy guidance for conditional generation. This is better aligned with code-based EHR data, although continuous measurements and richer temporal structures still require additional modeling.

b) Flow-matching-based healthcare synthesis: Flow matching in healthcare is represented by **PatientFlow** [41], which moves beyond static EHR tables to mixed-type longitudinal clinical data. It learns latent representations for static patient information and temporal assessments, then uses a static latent flow and a temporal flow conditioned on the generated static representation to synthesize patient trajectories. Its evaluation is broader than generic fidelity, covering longitudinal similarity, train-on-synthetic/test-on-real prognosis, clinical rules, expert discrimination, and privacy risk. At the same time, reported violations of onset and ALSFRS-R progression rules show that longitudinal clinical synthesis still needs domain-semantic validation, not just distributional similarity or downstream utility.

c) Comparative Analysis: Healthcare synthesis is constrained by clinical meaning rather than table format alone. MedDiff and EHR-TabDDPM show how general tabular diffusion can be adapted to EHR records, whereas DPM-EHR, FlexGen-EHR, and PatientFlow move toward longitudinal or heterogeneous patient trajectories. EHRDiff and EHR-D3PM highlight a complementary representation issue: continuous encodings are convenient, but discrete diagnosis or procedure codes may require discrete generative processes. Evaluation therefore needs to go beyond distributional fidelity and downstream utility by also checking privacy risk, clinical rules, and temporal consistency. Otherwise, a synthetic record may look statistically plausible while violating medical semantics.

3) Diffusion Models for Single Table Synthesis in Finance Domain: In finance, diffusion models are mainly used to adapt mixed-type tabular synthesis to sensitive transactions, entity identifiers, and class imbalance, which are central to risk modeling, fraud detection, and privacy-preserving data sharing [12]. **FinDiff** [88] targets mixed-type financial tables by replacing categorical values with continuous embeddings, concatenating them with normalized numerical features and conditioning embeddings, and training a DDPM in the resulting representation space. The generated embeddings are then mapped back to valid categories by nearest-neighbor decoding. This design makes diffusion easier to apply to financial tables with categorical attributes, but its quality depends on whether the learned embeddings preserve useful category semantics.

However, representing each categorical column with embeddings is still not enough when a column denotes an entity, such as a company or an individual. **EntTabDiff** [89] therefore treats entity columns as a special part of the financial table rather than as ordinary categories. Instead of only reproducing records for entities observed in the training data, it learns an entity distribution and generates tabular records conditioned on synthetic entities through cross-attention. This makes the model more suitable for entity-aware financial synthesis, although the generated records still depend on how well the entity distribution and entity-feature relationships are learned.

A second finance-specific issue is that the most important classes are often rare, as in fraud or high-risk detection. **ImbFinDiff** [90] extends FinDiff in this direction by adding class conditioning and an auxiliary class-label prediction objective, encouraging the diffusion model to generate minority-

class records more effectively. Taken together, these finance-oriented methods show that the main issue is not a new diffusion backbone, but how to inject financial structure, including categorical semantics, entity identity, and class imbalance, into representation and conditioning design.

B. Diffusion Models for Multi-relational Data Synthesis

Most tabular data synthesis methods focus on single-table generation (see Sec. IV-A); however, real-world datasets often consist of multiple interconnected tables, highlighting the importance of multi-relational data synthesis—a challenge that remains largely underexplored. In this subsection, we first present the task-specific setup and problem definition (see Problem 2). Then, we review related works in chronological order, with a summary provided in Table II.

In addition to capturing the characteristics of individual tables, multi-relational data synthesis requires modeling relationships across multiple tables that are connected (i.e., constrained by foreign keys). A multi-relational database $\mathcal{R}$ consists of P tables $\{R_1, ..., R_P\}$. Each table R_p has *primary key* which is the unique identifiers of rows. Given two tables R_p and R_q, we say R_p refers to R_q (or R_p has a *foreign key constraint* with R_q, or R_p has a *parent-child relationship* with R_q where R_q is the *parent* and R_p is the *child*) if R_p has a feature (called *foreign key*) that refers to the primary key of R_q. With these notations, the *multi-relational data synthesis problem* can be defined as follows [95]:

Problem 2 (Multi-relational Data Synthesis)**.** *Given $\mathcal{R} = \{R_1, ..., R_P\}$, we aim to generate a synthetic database $\tilde{\mathcal{R}} = \{\tilde{R}_1, ..., \tilde{R}_P\}$ that preserves the structure, foreign-key constraints, as well as correlations among features, including any inter-column correlations within the same table, inter-table correlations, and intra-group correlations.*

TABLE II
OVERVIEW OF DIFFUSION MODELS FOR MULTI-RELATIONAL DATA SYNTHESIS

Paradigm	Model Name	Year	Venue	Feature Type	Domain
Diffusion	**ClavaDDPM** [95]	2024	NeurIPS	Num by default	Generic
	GNN-TabSyn [96]	2024	NeurIPSW	Num+Cat	Generic

Diffusion-based multi-relational synthesis still builds on single-table generators, but the key additional challenge is how to preserve dependencies across tables connected by foreign keys. **ClavaDDPM** [95] addresses this by introducing clustering labels as relational intermediaries: instead of synthesizing each table independently, it uses these labels to summarize cross-table dependencies and then applies a TabDDPM-style generator [8] in a unified continuous feature space. This design is relatively simple and scalable, and it can capture group-level dependencies, but its success depends on whether the clustering labels preserve the relational information that matters for downstream queries.

GNN-TabSyn [96] takes a more structural view. It represents a relational database as a heterogeneous graph, where rows are nodes and foreign-key links are edges, and then uses GNN embeddings as conditions for a latent TabSyn generator [35]. This makes inter-table topology explicit and is better aligned with the relational nature of the data. However, the method also inherits the cost and assumptions of graph representation learning, and it does not naturally synthesize unseen graph structures. Overall, these two methods show that multi-relational synthesis is less about generating realistic rows in isolation, and more about preserving join behavior, foreign-key structure, and cross-table dependencies.

Compared with single-table synthesis, this area remains underdeveloped: there are only a few diffusion-based systems, no mature flow-matching counterpart, and limited agreement on benchmarks or metrics. A promising direction is to combine transport-based generation with explicit relational constraints, so that probability paths or noising processes operate over both row attributes and relational structure. However, such methods would need to handle schema validity, unseen keys, rare relationship patterns, and privacy leakage caused by linkability across tables.

V. DIFFUSION MODELS FOR DATA IMPUTATION

Missing values imputation is a long-standing research problem in data mining and machine learning community [97]. These approaches can be divided into two categories [98]: 1) *iterative approaches* that estimate the conditional distribution of one feature based on other features (e.g., MICE[99]); and 2) *deep generative approaches* that train a generative model to generate values in missing parts based on observed parts (e.g., MIDA [100] and MIWAE [9] based on denoising AEs, HIVAE [101] based on VAEs, and GAIN [15] based on GANs). We formally define the missing data imputation problem below (see Problem 3) and summarize representative diffusion-based imputation methods in Table III. We focus on diffusion here because dedicated flow-matching methods for tabular imputation have not yet formed a comparable literature; existing tabular flow-matching work mainly studies synthesis or anomaly scoring rather than missing-value recovery.

Problem 3 (Tabular Data Imputation)**.** *Given a D-dimensional training dataset $\mathbf{X} = \{{}_i\mathbf{x}\}_{i=1}^N$, a feature $j \in \{1, ..., D\}$ of ${}_i\mathbf{x}$ is denoted as ${}_i^j\mathbf{x}$, where the j-th feature can be numerical or categorical. Besides, any feature j can suffer from missing values. Let $\mathcal{X} = (\mathbb{R} \cup \emptyset)^D$ be the input space, where $\mathbb{R}$ denotes the real number space for a numerical feature. For brevity, we also utilize $\mathbb{R}$ to denote the corresponding range if it is a categorical feature. Data imputation aims to find a function $f(\cdot)$: $\mathcal{X} \to \mathbb{R}^D$ which can replace the missing values with reasonable values.*

Early diffusion-based imputation methods formulate missing-value recovery as conditional denoising. **TabCSDI** [65] extends CSDI [108] from time-series imputation to tabular data by splitting a record into observed and target missing entries, and then training a conditional diffusion model to denoise the target part given the observed part. To support mixed-type tables, categorical features are converted by one-hot encoding, analog bits [109], or feature tokenization [110], and the imputed outputs are decoded back

TABLE III
OVERVIEW OF DIFFUSION MODELS FOR DATA IMPUTATION

Paradigm	Model Name	Year	Venue	Feature Type	Domain
Diffusion	**TabCSDI** [65]	2022	NeurIPSW	Num+Cat	Generic
	TabDiff [70]	2025	ICLR	Num+Cat	Generic
	SimpDM [102]	2024	CIKM	Num+Cat	Generic
	MTabGen [103]	2025	TKDD	Num+Cat	Generic
	DDPM-Perlin [104]	2024	KBS	Num	Generic
	NewImp [105]	2024	NeurIPS	Num	Generic
	DiffPuter [106]	2025	ICLR	Num+Cat	Generic
	HARPOON [107]	2026	ICLR	Num+Cat	Generic
	TabNAT [80]	2025	ICML	Num+Cat	Generic
	TabuSDE [81]	2025	ICDMW	Num+Cat	Generic

to the original feature space. This provides a simple template for diffusion-based imputation, but categorical validity and dependency preservation still depend on the encoding and decoding choices.

TabDiff [70], originally proposed for data synthesis (see Sec. IV-A), follows a similar conditional-denoising view for imputation. Observed features are fixed during the reverse process, while missing numerical and categorical features are generated with classifier-free guidance. Compared with TabCSDI, TabDiff benefits from a diffusion design already tailored to mixed-type tabular data, but its imputation setting is still mainly an adaptation of a synthesis model rather than a method designed specifically for incomplete-data learning.

The next group of methods tries to make conditional diffusion more suitable for the accuracy-driven nature of imputation. **SimpDM** [102] starts from the observation that generation and imputation have different goals: diffusion models are designed to produce diverse samples from noise, whereas imputation usually requires stable and accurate values for a fixed observed context. It therefore adds a self-supervised alignment mechanism so that imputations from the same observed entries remain consistent, and uses perturbation-based data augmentation to reduce the effect of small tabular training sets. For mixed-type features, however, it still follows the TabDDPM-style separation of Gaussian diffusion for numerical features and multinomial diffusion for categorical features [8], so it inherits similar mixed-type modeling limitations.

MTabGen [103] takes a broader route by turning imputation and synthesis into two cases of the same masked-generation framework. It normalizes numerical features, ordinally encodes categorical features, projects heterogeneous columns into a continuous latent space, and uses a Transformer encoder-decoder with attention-based conditioning to model the relation between observed and missing features. Dynamic masking allows the model to handle different visible-feature patterns; when all features are masked, the same framework becomes data synthesis. This makes MTabGen more flexible than a pure imputation model, but the unified latent representation and ordinal categorical encoding still make feature semantics and decoding quality important.

After conditional-denoising methods, another line asks whether the standard diffusion process itself is suited to imputation. **DDPM-Perlin** [104] follows the TabCSDI-style observed/target split, but replaces Gaussian corruption with Perlin noise when imputing numerical features. This explores whether structured noise can be more natural for imputation than independent Gaussian perturbations. However, the replacement is not well supported by the standard DDPM derivation, since Perlin noise does not have the same additive Gaussian properties.

NewImp [105] revisits the objective side of the same problem. It argues that diffusion models may generate unnecessarily diverse samples for a task that often needs accurate point estimates, and that mask-based training can be mismatched with the missingness mechanism at test time. Under a Wasserstein-gradient-flow view [111], NewImp introduces a cost functional with negative entropy regularization to reduce sample diversity and uses a joint-distribution formulation to avoid relying on a masking-based conditional objective. This gives a more imputation-oriented formulation, but it currently handles only numerical features.

A related issue is whether diffusion imputation should be treated as ordinary conditional sampling or as an incomplete-data optimization problem. **DiffPuter** [106] argues that predictive imputers often outperform generative imputers because estimating a joint distribution is difficult when missing entries are unknown. It addresses this by integrating an EM-style procedure [112] with diffusion: the model alternates between learning a joint distribution over observed entries and current missing-value estimates, and updating the missing entries through reverse denoising. This makes diffusion imputation closer to incomplete-data learning, but mixed-type handling still relies on one-hot encoding followed by Gaussian diffusion over the encoded features.

HARPOON [107] extends inference-time conditioning beyond missing-value imputation. It trains one unconditional data-space DDPM on standardized numerical and one-hot categorical features, and then guides sampling with differentiable constraints. The denoiser acts as a projection toward the learned data manifold, while constraint gradients move samples along this manifold, so missing-value constraints, ranges, categorical conditions, conjunctions, and disjunctions can be handled without retraining. Experiments on eight datasets show strong imputation performance and much lower violation rates than DiffPuter and GReaT under inequality constraints. However, HARPOON still works in a continuous encoded space and trains after filtering incomplete rows; it is therefore best viewed as a reusable constraint-guided sampler rather than a model that learns directly from naturally incomplete tables.

TabNAT [80] represents a different setting. It treats imputation as flexible conditional generation from a masked bidirectional Transformer trained on complete data. Missing target columns are sampled in a random order conditioned on observed columns; discrete targets are sampled from predicted categorical distributions, while continuous targets use a RePaint-style denoising procedure. This is useful for heterogeneous conditional inference, but its imputation experiment injects MCAR missingness into the test set after complete-data training, so it should not be read as a method that learns directly from incomplete training tables.

TabuSDE [81] moves one step further by modeling the missingness pattern together with the feature values. During conditional reverse diffusion, observed entries are clamped and

only missing entries are updated. Its denoiser predicts both feature values and missingness probabilities, and a VAE-learned missingness prior helps model structured MAR/MNAR masks rather than independent random masks. This makes TabuSDE important because it treats missingness as part of the data-generating process, not merely as an inference-time mask. At the same time, it is a large integrated system, combining feature-specific SDEs, hybrid encoders, cross-feature attention, mask modeling, privacy accounting, hyperparameter search, and accelerated sampling, so its results support the promise of joint value–mask diffusion but do not isolate a single mechanism as the universal solution for tabular imputation.

Comparative Analysis: Taken together, the methods in Table III turn diffusion into an imputer in several related ways. One line treats imputation as conditional generation from a joint model trained on complete data, as in TabDiff, MTabGen, and TabNAT; this is suitable when a clean reference table is available, but it mainly evaluates test-time masking rather than learning from incomplete data. A second line changes the objective or sampling process to better match incomplete observations, as in TabCSDI, DDPM-Perlin, NewImp, and DiffPuter; these methods are closer to observational databases, but remain sensitive to the missingness mechanism, feature encoding, and observed-entry clamping. HARPOON adds an inference-time guidance view: missing entries become one type of condition, so the same sampler can also enforce range, categorical, conjunctive, or disjunctive constraints, although it still trains on complete rows. TabuSDE adds a missingness-aware direction by modeling values and masks jointly, which is important when missingness itself carries signal.

These distinctions make the evaluation protocol as important as the model design. MCAR masking on complete test data, MAR/MNAR missingness during training, and naturally incomplete benchmark tables test different abilities. Entry-wise RMSE, categorical accuracy, downstream utility, distributional distance, correlation preservation, and uncertainty calibration also capture different failure modes. A useful imputation study should therefore report how missingness is created or observed, whether the training table is complete, how observed entries are clamped during sampling, and whether multiple-imputation uncertainty is preserved. From this perspective, conditional denoising is reasonable for complete-training and simple test-time missingness, while mask-aware, incomplete-data, or value–mask objectives are better aligned with structured or informative missingness. Flow matching could in principle support similar conditional generation, but current tabular flow-matching studies have not yet established imputation-specific paths, mask handling, or uncertainty protocols.

VI. Diffusion Models for Trustworthy Data Synthesis

Trustworthy data synthesis asks whether synthetic tables can be released and used without exposing individuals, reinforcing group bias, or reproducing training records. We begin with privacy-preserving synthesis. Two deployment settings are formalized below: cross-silo tabular synthesis, where different parties hold different feature subsets of the same records and want to synthesize a complete table without exposing raw feature values (Problem 4), and federated tabular synthesis, where clients hold different records and collaboratively train a generator without centralizing local data (Problem 5). We also discuss centralized differential privacy, where data are held by one curator but each record's influence on the learned generator must be bounded. We then review fairness-preserving synthesis (Problem 6) and memorization auditing. The reviewed methods are diffusion-based because dedicated flow-matching methods for privacy-preserving, fairness-aware, or memorization-aware tabular synthesis have not yet formed a comparable literature. Recent tabular flow-matching generators may report empirical disclosure-risk metrics in generic synthesis settings, but they do not yet provide explicit threat models, formal privacy mechanisms, fairness interventions, or memorization-mitigation protocols for trustworthy data release. We summarize representative methods in Table IV.

Problem 4 (Cross-Silo Tabular Synthesis)**.** *Consider there are L distinct parties $\{P_1, ..., P_L\}$, each party stores a subset of features $\mathbf{X}_l \in \mathbb{R}^{N \times D_l}$ with D_l the number of features stored at party P_l. Altogether, we have $\mathbf{X} = \mathbf{X}_1||\mathbf{X}_2||...||\mathbf{X}_L$ ("$||$" means column-wise concatenation) and $\mathbf{X} \in \mathbb{R}^{N \times D}$ with $D = \sum_{l=1}^{L} D_l$. The goal of cross-silo tabular synthesis is to generate a synthetic dataset $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}_1||\tilde{\mathbf{X}}_2||...||\tilde{\mathbf{X}}_L$ that are distributionally similar to the original dataset $\mathbf{X}$ while maintaining the private information of the actual values.*

Problem 5 (Federated Tabular Synthesis)**.** *Consider there are M distinct clients $\{Q_1, ..., Q_M\}$, each client stores a subset of samples $\mathbf{X}_m \in \mathbb{R}^{N_m \times D}$ with N_m the number of samples stored at client Q_m. Altogether, we have $\mathbf{X} = \mathbf{X}_1 \oplus \mathbf{X}_2 \oplus \cdots \oplus \mathbf{X}_M$ ("$\oplus$" denotes row-wise concatenation), and $\mathbf{X} \in \mathbb{R}^{N \times D}$ with with $N = \sum_{m=1}^{M} N_m$. The goal of federated learning based tabular synthesis is to generate a synthetic dataset $\tilde{\mathbf{X}} = \tilde{\mathbf{X}}_1 \oplus \tilde{\mathbf{X}}_2 \oplus ... \oplus \tilde{\mathbf{X}}_M$ that are distributionally similar to the original dataset $\mathbf{X}$ while maintaining the private information of the actual values.*

Meanwhile, *fairness-preserving data synthesis* aims to use diffusion models to generate synthetic tabular data that maintains fairness w.r.t. sensitive features (e.g., age, gender, race, etc.) while preserving data utility. Formally, it is defined as:

Problem 6 (Fairness-Preserving Data Synthesis)**.** *Given a D-dimensional dataset $\mathbf{X} = \{_i\mathbf{x}\}_{i=1}^{N}$, where each data point $_i\mathbf{x} \in \mathbb{R}^D$ represents a record with D features, let $\mathbf{S} \subseteq \{1, \ldots, D\}$ denote the set of indices corresponding to sensitive features. The goal is to learn a generative function $g(\cdot)$ to generate synthetic data $\hat{\mathbf{X}} = g(\mathbf{X})$ such that: 1) The synthetic data $\hat{\mathbf{X}} = \{_i\hat{\mathbf{x}}\}_{i=1}^{N}$ preserves fairness with respect to sensitive features in $\mathbf{S}$, ensuring no unfair bias is introduced; and 2) The synthetic data $\hat{\mathbf{X}}$ maintains the statistical properties and utility of the original dataset $\mathbf{X}$ for downstream tasks.*

Cross-silo tabular synthesis is the first privacy-preserving setting. **SiloFuse** [113] targets vertically partitioned tables, where different silos store different feature subsets of the same records. It uses local autoencoders to map silo-side features, including categorical features, into continuous embeddings,

TABLE IV
OVERVIEW OF DIFFUSION MODELS FOR TRUSTWORTHY DATA SYNTHESIS.'*' REFERS TO A NAME CREATED BY US FOR CONVENIENCE.

Paradigm	Model Name	Year	Venue	Feature Type	Domain
Diffusion	**SiloFuse** [113]	2024	ICDE	Num+Cat	Generic
	FedTabDiff [114]	2024	ArXiv	Num+Cat	Generic
	FairTabDDPM* [115]	2025	TMLR	Num+Cat	Generic
	DP-Fed-FinDiff [116]	2024	ArXiv	Num+Cat	Finance
	DP-FinDiff [117]	2025	ArXiv	Num+Cat	Generic
	TabCutMix/TabCutMixPlus [118]	2025	ICML	Num+Cat	Generic
	DynamicCut [119]	2026	TMLR	Num+Cat	Generic

and then trains a central latent DDPM over the aggregated embeddings. This allows the generator to model cross-silo feature correlations without directly sharing raw feature values. The design is especially useful for sparse or high-cardinality tables, although TabDDPM-style Gaussian/multinomial diffusion can still be preferable when the number of features is small.

Federated tabular synthesis arises when clients hold different records rather than different features. **FedTabDiff** [114] adapts FinDiff to this horizontally partitioned setting by training local diffusion models at clients and aggregating model updates through federated learning. This reduces raw-data sharing, but federated optimization alone does not provide a formal record-level privacy guarantee. **DP-Fed-FinDiff** [116] therefore keeps the same basic federated diffusion setup but adds differential privacy to parameter updates through clipping and Gaussian noise. Its main lesson is that data locality and individual-level privacy are different requirements: federated learning controls where data are stored, whereas differential privacy controls how much one record can influence the learned generator.

Centralized differentially private synthesis raises a different difficulty. **DP-FinDiff** [117] studies differentially private diffusion for mixed-type tables when data are centrally available but record-level influence must still be bounded. Directly applying differentially private stochastic gradient descent (DP-SGD) is difficult because one-hot categorical representations increase dimensionality and sparsity, while clipping and Gaussian noise can distort the denoising signal unevenly across timesteps and features. DP-FinDiff therefore follows FinDiff by using dense categorical embeddings and training the denoiser in the embedding space under per-example clipping, Gaussian noise, and privacy random variable (PRV) accounting. It further uses adaptive timestep sampling and a feature-aggregated loss to reduce clipping-induced bias. This gives a practical differentially private variant of FinDiff, but it still faces the usual privacy–utility tradeoff, and formal differential privacy does not by itself address minority coverage, fairness, or bias inherited from the training data.

Fairness-preserving synthesis broadens trustworthiness beyond privacy. **FairTabDDPM** [115] starts from the observation that tabular diffusion models may reproduce group bias in the training data. To mitigate this, it builds balanced joint distributions of sensitive attributes and target labels, and conditions a TabDDPM-style Gaussian/multinomial diffusion model on both label and sensitive columns. This makes fairness control part of the generation process rather than a purely post-hoc filter. However, the guarantee is tied to the chosen sensitive attributes, target labels, and fairness metrics; reducing measured disparities does not necessarily remove other biased correlations or ensure fairness for every downstream use.

Memorization auditing is another part of trustworthy synthesis, because a high-fidelity generator may reproduce individual training records too closely even without an explicit privacy failure. Fang et al. [118] show that tabular diffusion models can memorize training samples and propose a mixed-type relative distance ratio to identify generated records that are much closer to their nearest training neighbor than to other training records. They further introduce **TabCutMix**, which swaps feature segments between same-class samples, and **TabCutMixPlus**, which swaps correlated feature groups to better preserve feature coherence. This line of work is useful because common distance-to-closest-record diagnostics can miss sample-level overfitting, but feature swapping may also create semantically invalid combinations in sensitive domains.

A follow-up study shifts the focus from model-level memorization to data-centric auditing [119]. It finds that memorized generations are often driven by a small subset of high-risk training records, and proposes **DynamicCut** to identify such records during a warm-up phase before retraining; **DynamicCutMix** combines this filtering with TabCutMix. The broader lesson is that privacy in tabular diffusion should be studied at three levels: formal mechanisms such as differential privacy, model-level attacks or memorization metrics, and data-centric auditing of repeatedly reproduced records. The remaining tradeoff is that removing high-risk records can also remove rare but analytically important cases, so memorization mitigation should be evaluated together with minority coverage, fairness, and domain validity.

Comparative Analysis: The methods in Table IV are best distinguished by their trust assumptions before considering the diffusion backbone. Cross-silo and federated synthesis reduce direct data sharing, but neither setting by itself gives a record-level privacy guarantee. Differentially private variants make individual influence explicit through a privacy budget, at the cost of utility, fidelity, or minority coverage. Fairness-aware synthesis changes the conditioning distribution to reduce measured group disparities, but it does not remove all biased correlations. Memorization auditing targets another failure mode: a non-private generator may reproduce particular training records even when aggregate fidelity and downstream utility look strong. These differences imply that trustworthy tabular generation cannot be evaluated by a single fidelity or utility score. Studies should report the deployment setting, adversary or privacy definition, privacy budget or attack protocol, fairness metrics, minority-group utility, and sample-level memorization diagnostics together. Flow matching remains largely unexplored in this setting. Its faster sampling may affect utility, privacy leakage, or memorization behavior, but without explicit threat models and attack evaluations, efficiency should not be taken as evidence of trustworthiness.

VII. DIFFUSION AND FLOW MATCHING MODELS FOR ANOMALY DETECTION

Anomaly detection aims to train diffusion or flow-matching models to learn the "normal" distribution of data from the

training set and identify anomalies as deviations from this learned distribution or learned transport behavior in the test data. Formally, it is defined below (see Problem 7). Because these models can either score deviations directly or synthesize auxiliary samples for downstream fraud classifiers, we summarize representative methods in Table V and organize the discussion into diffusion-based scoring (see Sec. VII-1), flow-matching-based scoring (see Sec. VII-2), auxiliary data generation for fraud detection (see Sec. VII-3), and a final comparative analysis (see Sec. VII-4).

Problem 7 (Tabular Anomaly Detection)**.** *Given a D-dimensional training dataset $\mathbf{X}_{train} = \{{}_i\mathbf{x}\}_{i=1}^{N_{train}}$, where each data point ${}_i\mathbf{x} \in \mathbb{R}^D$ represents a record with D features, the objective is to learn a function $f(\cdot) : \mathbb{R}^D \to \{0, 1\}$ that distinguishes between normal and anomalous data points. During testing, given a D-dimensional test dataset $\mathbf{X}_{test} = \{{}_j\mathbf{x}\}_{j=1}^{N_{test}}$, the function $f(\cdot)$ predicts $f({}_j\mathbf{x}) = 1$ if ${}_j\mathbf{x}$ is an anomaly and $f({}_j\mathbf{x}) = 0$ if it is normal. Anomalies are identified as data points in $\mathbf{X}_{test}$ that significantly deviate from the normal patterns learned from $\mathbf{X}_{train}$.*

TABLE V
OVERVIEW OF DIFFUSION AND FLOW MATCHING MODELS FOR ANOMALY DETECTION

Paradigm	Model Name	Year	Venue	Feature Type	Domain
Diffusion	**TabADM** [120]	2023	ArXiv	Num by default	Generic
	DTE [121]	2024	ICLR	Num by default	Generic
	SDAD [122]	2024	Inf. Sci.	Num by default	Generic
	NSCBAD [123]	2024	OpenReview	Num by default	Generic
	FraudDiffuse [124]	2024	ICAIF	Num+Cat	Finance
	FraudDDPM [125]	2024	ISIJ	Num+Cat	Finance
	DDAE [126]	2025	KDD	Num by default	Generic
Flow Matching	**TCCM** [71]	2025	NeurIPS	Num by default	Generic

1) Diffusion-Based Scoring: Early diffusion-based anomaly detectors use the denoising process as a scoring device rather than as a data-release mechanism. **TabADM** [120] is designed for unsupervised tabular anomaly detection when the training set may already contain anomalies. It trains a robust DDPM with a sample-rejection step that downweights high-loss samples, and then scores a test record by its denoising or reconstruction loss across timesteps. Samples far from the learned high-density region tend to produce larger losses. This makes TabADM a direct diffusion-based scoring method, but it inherits the cost of repeated denoising and provides limited explainability or missing-value support.

DTE [121] starts from the same concern about computational cost. Instead of learning a full reverse denoising model, it estimates the diffusion time associated with a noisy input sample. The intuition is that anomalous samples are farther from the normal data manifold and therefore correspond to larger estimated diffusion times. DTE can use non-parametric or neural estimators and also supports reconstruction-based interpretation through deterministic ODE flows. It is more efficient than full DDPM-style scoring, but it is mainly designed for numerical features and still depends on how well diffusion time separates normal and abnormal regions.

After TabADM and DTE, later work changes what is scored. **SDAD** [122] argues that generative anomaly detectors should not only reconstruct data, but also learn discriminative representations. It first uses self-supervised pretext tasks to train an encoder that separates normal and anomalous samples in latent space, and then fits a denoising diffusion model to the normal latent distribution. At inference, samples with large latent reconstruction errors are treated as anomalies. This makes SDAD more representation-aware than direct input-space reconstruction, but it is designed for the semi-supervised setting with normal-only training data, and its pseudo-anomaly construction still relies on Gaussian noise rather than real abnormal patterns. **NSCBAD** [123] further shifts the score from reconstruction to the learned score function. Instead of estimating the full data density or running a full reverse denoising chain, it trains a noising conditional score network and scores a test sample by how well the predicted score matches injected noise at a fixed timestep. This avoids the long sequential inference of DDPM-based detectors and is therefore faster, but it still assumes that score mismatch under numerical noising is a reliable proxy for abnormality.

DDAE [126] takes a more lightweight reconstruction route. Instead of training a full diffusion generator, it imports diffusion-style scheduled noise into a denoising autoencoder: inputs are corrupted at different timesteps, a timestep-conditioned encoder–decoder reconstructs the clean sample, and the cumulative reconstruction error becomes the anomaly score. Its contrastive variant further shapes the latent space by separating clean and noised samples. This design keeps the useful idea of timestep-dependent corruption while avoiding iterative reverse sampling, but its behavior depends on the noise schedule and it mainly treats tabular inputs as standardized numerical inputs.

2) Flow-Matching-Based Scoring: **TCCM** [71] pushes the simplification further from a flow-matching perspective. It trains only on normal samples and learns a time-conditioned contraction field toward the origin, using a fixed contraction target rather than a full probability-flow trajectory. At inference, it measures how far the predicted vector deviates from the expected contraction, so the anomaly score is computed in one forward pass instead of through multi-step denoising or ODE sampling. Because the residual remains in the input feature space, it also gives feature-wise attributions. The main caveat is the semi-supervised mismatch assumption: anomalous samples should fail to follow the contraction dynamics learned from normal data. Thus, its behavior still needs validation for mixed-type tables and high-stakes domains beyond standardized numerical benchmarks.

3) Auxiliary Data Generation for Fraud Detection: Auxiliary data generation for fraud detection uses diffusion models as supervised oversamplers rather than as direct anomaly scorers. **FraudDiffuse** [124] targets labeled transaction data with severe class imbalance. It embeds categorical features into a continuous space and trains a latent DDPM to generate fraudulent transactions, using the normal-transaction distribution to guide the diffusion process toward boundary-like fraud samples and adding a contrastive loss to keep synthetic fraud close to real fraud. This makes diffusion useful for improving downstream fraud classifiers, but it still requires labeled fraudulent data and may miss new fraud patterns outside training.

TABLE VI
OVERVIEW OF DIFFUSION AND FLOW MATCHING MODELS FOR TABULAR DATA. FROM TOP TO BOTTOM, ROW SHADING MARKS DATA AUGMENTATION, DATA IMPUTATION, TRUSTWORTHY DATA SYNTHESIS, AND ANOMALY DETECTION. "NUM" AND "CAT" REPORT NUMERICAL/CATEGORICAL SUPPORT AND THE MAIN NORMALIZATION OR ENCODING: MM=MIN–MAX, QT=QUANTILE TRANSFORM, ZS=Z-SCORE, OH=ONE-HOT, AB=ANALOG BITS, DC=DICTIONARY, DE=DUMMY, IE=INTEGER, OE=ORDINAL, PSK=PSK, FT=FEATURE TOKENIZATION OR LEARNED EMBEDDING, CC=CATCONVERTER, EMB=LEARNED EMBEDDING. "#DATASETS" IS THE NUMBER OF BENCHMARK DATASETS; "?/57 A" DENOTES THE SUBSET USED FROM ADBENCH [127]. GITHUB ICONS LINK TO CODE; "COMPLETE" INDICATES COMPLETE-DATA TRAINING.

Paradigm	Model Name	Year	Venue	Tasks	Num	Cat	Backbone	#Datasets	Metrics	Code	Complete	Domain
Diffusion	**SOS** [64]	2022	KDD	Synthesis (single, generic)	✓	✗	SDEs	6	Utility	○	✓	Generic
	STaSy [32]	2023	ICLR	Synthesis (single, generic)	✓(MM)	✓(OH)	SDEs	15	Fidelity, Utility, Diversity	○	✓	Generic
	TabDDPM [8]	2023	ICML	Synthesis (single, generic)	✓(QT)	✓(OH)	DDPM+MLD	16	Fidelity, Utility, Privacy	○	✓	Generic
	CoDi [34]	2023	ICML	Synthesis (single, generic)	✓(MM)	✓(OH)	DDPM+MLD	15	Utility, Diversity	○	✓	Generic
	AutoDiff [33]	2023	NeurIPSW	Synthesis (single, generic)	✓	✓	Any	15	Fidelity, Utility, Privacy	○	✓	Generic
	MissDiff [76]	2023	ICMLW	Synthesis (single, generic)	✓(MM)	✓(OH)	SDEs	3	Fidelity, Utility	✗	✗	Generic
	TabSyn [35]	2024	ICLR	Synthesis (single, generic), Imputation	✓	✓(OH)	SDEs	6	Fidelity, Utility, Diversity, Privacy	○	✓	Generic
	Forest-Diffusion [36]	2024	AISTATS	Synthesis (single, generic), Imputation	✓	✓(DE)	SDEs	27	Fidelity, Diversity, Utility	○	✗	Generic
	TabDiff [70]	2025	ICLR	Synthesis (single, generic), Imputation	✓(MM)	✓(OH)	SDEs+MSD	7	Fidelity, Diversity, Utility	○	✓	Generic
	TabUnite [77]	2024	OpenReview	Synthesis (single, generic)	✓(QT)	✓(AB, PSK, DC)	SDEs+MSD	10	Fidelity, Diversity, Utility	○	✓	Generic
	CDTD [78]	2025	ICLR	Synthesis (single, generic)	✓	✓(FT)	Latent SDEs	11	Fidelity, Utility, Privacy	○	✓	Generic
	TabRep [79]	2026	TMLR	Synthesis (single, generic)	✓(QT)	✓(CC)	DDPM/FM	7	Fidelity, Utility, Privacy	○	✓	Generic
	TabNAT [80]	2025	ICML	Synthesis (single, generic), Imputation	✓	✓(Emb)	Conditional DDPM	10	Fidelity, Utility, Privacy, Runtime	○	✓	Generic
	TabuSDE [81]	2025	ICDMW	Synthesis (single, generic), Imputation	✓	✓(Emb)	SDE	25	Fidelity, Imputation, Privacy	✗	✗	Generic
Flow Matching	**TabbyFlow** [37]	2025	ICML	Synthesis (single, generic)	✓	✓	VFM	6	Fidelity, Diversity, Utility, Privacy	○	✓	Generic
	TabSynFlow [38]	2026	TMLR	Synthesis (single, generic)	✓	✓	Latent Flow Matching	6	Utility, Privacy, Runtime	○	✓	Generic
	TabFlowM [39]	2026	TMLR Sub.	Synthesis (single, generic)	✓	✓(FT)	Latent Flow Matching	12	Fidelity, Utility, Privacy, Runtime	✗	✓	Generic
	TabCascade [40]	2026	ArXiv	Synthesis (single, generic)	✓	✓	Conditional Flow Matching	12	Fidelity, Utility, Privacy	○	✗	Generic
Diffusion	**MedDiff** [82]	2023	ArXiv	Synthesis (single, healthcare)	✓	✗	DDIM	2	Fidelity, Utility	✗	✓	Healthcare
	EHR-TabDDPM [83]	2023	ArXiv	Synthesis (single, healthcare)	✓(QT)	✓(OH)	DDPM	4	Fidelity, Utility, Privacy	✗	✓	Healthcare
	DPM-EHR* [84]	2023	NeurIPSW	Synthesis (single, healthcare)	✓	✓(OH)	DDPM	2	Fidelity, Diversity, Utility, Privacy	✗	✓	Healthcare
	FlexGen-EHR [85]	2024	ICLR	Synthesis (single, healthcare)	✓+ TS (FT)	✓(FT)	DDPM	2	Fidelity, Utility, Privacy	✗	✗	Healthcare
	EHRDiff [86]	2024	TMLR	Synthesis (single, healthcare)	✓(MM)	✓(OH)	SDEs	3	Fidelity, Utility, Privacy	○	✓	Healthcare
	EHR-D3PM [87]	2025	TMLR	Synthesis (single, healthcare)	✗	✓(OH)	D3PM	3	Fidelity, Utility, Privacy	✗	✓	Healthcare
Flow Matching	**PatientFlow** [41]	2026	AI Med.	Synthesis (single, healthcare)	✓+ TS	✓(Emb)	Latent Flow Matching	1	Fidelity, Utility, Clinical Rules, Privacy	○	✗	Healthcare
Diffusion	**FinDiff** [88]	2023	ICAIF	Synthesis (single, finance)	✓(ZS)	✓(OH)	DDPM	3	Fidelity, Utility, Privacy	○	✓	Finance
	EntTabDiff [89]	2024	ICAIF	Synthesis (single, finance)	✓(ZS)	✓(OH)	DDPM	3	Fidelity, Utility, Privacy	✗	✓	Finance
	Imb-FinDiff [90]	2024	ICAIF	Synthesis (single, finance)	✓(ZS)	✓(OH)	DDPM	4	Fidelity, Utility	✗	✓	Finance
Diffusion	**ClavaDDPM** [95]	2024	NeurIPS	Synthesis (multi-relational)	✓	✓(IE)	DDPM	5	Fidelity, Diversity, Utility, Dependency	○	✓	Generic
	GNN-TabSyn [96]	2024	NeurIPSW	Synthesis (multi-relational)	✓	✓(IE)	DDPM	6	Fidelity, Utility, Privacy	○	✓	Generic
Diffusion	**TabCSDI** [65]	2022	NeurIPSW	Imputation	✓	✓(OH, AB, FT)	Conditional DDPM	7	Accuracy	○	✗	Generic
	TabDiff [70]	2025	ICLR	Imputation	✓	✓	Conditional DDPM	7	Accuracy	○	✗	Generic
	SimpDM [102]	2024	CIKM	Imputation	✓(QT)	✓(OH)	DDPM+MLD	17	Accuracy	○	✗	Generic
	MTabGen [103]	2025	TKDD	Imputation	✓(QT)	✓(OE)	DDPM+MLD	10	Utility	✗	✗	Generic
	DDPM-Perlin [104]	2024	KBS	Imputation	✓	✗	DDPM	10	Accuracy	✗	✗	Generic
	NewImp [105]	2024	NeurIPS	Imputation	✓	✗	Wasserstein GF	8	MAE, WASS	○	✗	Generic
	DiffPuter [106]	2025	ICLR	Imputation	✓	✓(OH)	DDPM	10	Accuracy	○	✗	Generic
	HARPOON [107]	2026	ICLR	Imputation, Conditional Generation	✓(ZS)	✓(OH)	Conditional DDPM	8	MSE, Accuracy, Violation, Fidelity, Utility, Runtime	○	✓	Generic
	TabNAT [80]	2025	ICML	Imputation	✓	✓(Emb)	Conditional DDPM	5	MAE, Accuracy	○	✓	Generic
	TabuSDE [81]	2025	ICDMW	Imputation	✓	✓(Emb)	SDE	25	MSE, Accuracy, CPS	✗	✗	Generic
Diffusion	**SiloFuse** [113]	2024	ICDE	Trustworthy Synthesis	✓(FT)	✓(FT)	Latent DDPM	9	Fidelity, Utility, Privacy	✗	✓	Generic
	FedTabDiff [114]	2024	ArXiv	Trustworthy Synthesis	✓(QT)	✓(FT)	DDPM	2	Fidelity, Utility, Privacy	○	✓	Generic
	FairTabDDPM* [115]	2025	TMLR	Trustworthy Synthesis	✓(QT)	✓(OH)	DDPM	3	Fidelity, Diversity, Utility, Privacy, Fairness	○	✓	Generic
	DP-Fed-FinDiff [116]	2024	ArXiv	Trustworthy Synthesis	✓(QT)	✓(FT)	DDPM	4	Fidelity, Utility, Privacy	✗	✓	Finance
	DP-FinDiff [117]	2025	ArXiv	Trustworthy Synthesis	✓	✓(Emb)	DP-DDPM	5	Fidelity, Utility, Privacy	○	✓	Generic
	TabCutMix/TabCutMixPlus [118]	2025	ICML	Trustworthy Synthesis	✓	✓	–	7	Memorization, Fidelity, Utility	○	✓	Generic
	DynamicCut [119]	2026	TMLR	Trustworthy Synthesis	✓	✓	–	5	Memorization, Fidelity, Utility	○	✓	Generic
Diffusion	**TabADM** [120]	2023	ArXiv	Anomaly Detection	✓	✗	DDPM	32/57 A	Accuracy	✗	✓	Generic
	DTE [121]	2024	ICLR	Anomaly Detection	✓(ZS)	✗	DDPM	57/57 A	Accuracy	○	✓	Generic
	SDAD [122]	2024	Inf. Sci.	Anomaly Detection	✓	✗	Latent DDPM	10/57 A	Accuracy	✗	✓	Generic
	NSCBAD [123]	2024	OpenReview	Anomaly Detection	✓	✗	SDEs	57/57 A+15	Accuracy	○	✓	Generic
	FraudDiffuse [124]	2024	ICAIF	Anomaly Detection, Synthesis (single, finance)	✓	✓(FT)	Latent DDPM	2	Utility	✗	✓	Finance
	FraudDDPM [125]	2024	ISIJ	Anomaly Detection, Synthesis (single, finance)	✓	✓(OH, FT)	DDPM	4	Utility	✗	✓	Finance
	DDAE [126]	2025	KDD	Anomaly Detection	✓(ZS)	✗	Denoising Autoencoder	47/57 A	PR-AUC, ROC-AUC	○	✓	Generic
Flow Matching	**TCCM** [71]	2025	NeurIPS	Anomaly Detection	✓(ZS)	✗	Flow Matching	47/57 A	PR-AUC, ROC-AUC, Runtime	○	✓	Generic

FraudDDPM [125] follows the same supervised augmentation goal but uses a simpler class-conditional strategy: one diffusion model generates normal transactions and another generates fraudulent transactions, after which synthetic samples are merged with real data to build a balanced classifier-training set. This turns diffusion into an oversampling tool for fraud detection. Its key evaluation question is therefore not whether the generated samples look realistic in isolation, but whether they improve downstream detection without leakage, unrealistic transactions, or overfitting to known fraud patterns.

4) Comparative Analysis: Overall, the methods in Table V use diffusion or flow matching in two different ways. The first use is normality scoring. TabADM asks whether a record can be reconstructed from a robust diffusion model even when training data may be contaminated, while SDAD, DDAE, DTE, NSCBAD, and TCCM mainly assume normal-only or mostly normal training data and turn different learned quantities into anomaly scores: latent reconstruction, scheduled denoising, diffusion time, score mismatch, or contraction mismatch. The key design difference is therefore not simply diffusion versus flow matching, but which notion of normality is exposed by the learned dynamics and how strongly that score depends on the chosen feature representation.

The second use is supervised auxiliary generation, where FraudDiffuse and FraudDDPM synthesize minority fraud samples to improve a downstream classifier. Here, the generative model is not itself the detector; it reshapes the training distribution so that the classifier can see more rare-event patterns. The central question is therefore whether synthetic fraud samples add useful variation while preserving transaction semantics, temporal plausibility, and the boundary between normal and fraudulent behavior. A useful direction is to combine the two roles more deliberately: normality scores can provide feature-level evidence at inference time, while controlled synthesis can cover rare patterns that are underrepresented during training.

VIII. DISCUSSION AND CONCLUSIONS

The methods summarized in Table VI show that diffusion and flow matching models have begun to address several challenges discussed in Sec. II-B, but the area is still developing. Recent benchmark evidence shows that conclusions about tabular generators can change once feature encoding, hyperparameter search space, tuning budget, sampling cost, and carbon footprint are controlled [50]. Broader studies of synthetic tabular data also show that generation quality cannot be separated from post-processing, evaluation, privacy pro-

tection, application setting, and deployment constraints [49]. Progress in diffusion and flow matching models therefore depends not only on stronger architectures, but also on clearer data assumptions, reproducible evaluation protocols, and explicit reporting of utility–privacy–cost tradeoffs. We close by summarizing several open directions.

(1) **Scalability and Sampling Cost**: Diffusion models are typically computationally intensive, especially for high-dimensional or large-scale tabular datasets. Recent work on scalable Forest-Diffusion [128] and flow-matching-based tabular synthesis, including TabSynFlow [38], TabFlowM [39], TabCascade [40], and PatientFlow [41], shows that sampling efficiency should be reported alongside fidelity and utility. Future studies should include training time, sampling time, number of function evaluations, memory footprint, and performance under limited-budget tuning, rather than only final utility or fidelity.

(2) **Task Coverage Gaps for Flow Matching**: Current tabular flow-matching studies are concentrated in single-table synthesis, longitudinal healthcare synthesis, and anomaly detection. By contrast, multi-relational synthesis, missing-value imputation under incomplete training data, differentially private generation, fairness-aware generation, and memorization mitigation remain largely diffusion-dominated or unexplored from a flow-matching perspective. It remains unclear whether path design, low-cost sampling, and velocity-field residuals can be adapted to missingness mechanisms, relational constraints, formal privacy guarantees, and fairness or memorization objectives.

(3) **Evaluation, Benchmarking, and Standardization**: Unlike images, tabular data lacks a single perceptual quality metric, so evaluation should jointly cover downstream utility, marginal fidelity, higher-order dependencies, privacy leakage, constraint satisfaction, fairness, and domain validity. Recent benchmark evidence shows that rankings can depend on preprocessing, encoding choices, hyperparameter search space, tuning budget, sampling cost, and carbon footprint [50]. Future benchmarks should include missing-data settings, imbalanced classes, high-cardinality categorical features, relational constraints, rare-event domains, and privacy-sensitive settings, while reporting code, preprocessing pipelines, tuning budgets, and failure cases.

(4) **Privacy, Memorization, and Fairness**: Formal differential privacy has begun to appear in tabular diffusion models [116], [117], but most works still rely on empirical privacy metrics such as distance-to-closest-record, attribute inference, or membership inference. Recent memorization studies [118], [119] indicate that a small subset of records may dominate memorized generations, so future research should combine formal privacy mechanisms, empirical attack evaluations, and data-centric memorization audits. These analyses should also be connected to fairness, because removing memorization-prone or rare records can change minority-group coverage.

(5) **Interpretability and Feature-Level Explanation**: Many diffusion and flow matching models operate as black-box generative systems. Anomaly detection methods such as TCCM [71] show that feature-level residuals can support interpretability, but similar explanation tools are underdeveloped for synthesis and imputation. Future methods should expose which features or feature interactions drive generated samples, imputed values, privacy risk, or anomalous scores. These explanations should be connected to tabular preprocessing choices, because scaling, encoding, and inverse transformations can change how feature-level evidence is interpreted.

(6) **Cross-Modality and Longitudinal Data**: Real data-engineering systems often mix static tables, event sequences, text, images, and time-series measurements. Healthcare models such as FlexGen-EHR [85] and PatientFlow [41] point toward multimodal and longitudinal tabular generation, but most current benchmarks remain single-table and static. Extending diffusion and flow matching models to multi-table, temporal, and cross-modal records is therefore an important next step. The key challenge is to preserve alignment between static attributes, temporal events, measurements, and domain-specific validity rules.

(7) **Hybrid and Representation-Centric Models**: Several recent methods combine diffusion or flow matching with autoencoders, transformers, tree models, or feature-token representations [33], [35], [79], [80], [39]. This trend suggests that the representation layer may be as important as the generative objective. Future work should disentangle representation learning, probability-path design, sampler choice, and backbone effects through controlled ablations. Otherwise, an apparent gain from a new generative model may actually come from better preprocessing, a stronger encoder, or a more favorable inverse mapping.

(8) **Modeling Feature Correlations and Constraints**: Many methods still handle categorical attributes through one-hot encodings or embeddings and rely on neural networks to recover dependencies implicitly. This is fragile for high-cardinality categories, mixed-type features, relational constraints, and domain rules. Future tabular diffusion and flow matching models should more explicitly model numerical–categorical dependencies, constraint satisfaction, schema-level structure, and rare feature combinations. Otherwise, a generated table may look realistic under marginal statistics while violating valid ranges, business rules, medical logic, or foreign-key consistency.

In summary, diffusion and flow matching models have become promising tools for tabular data modeling, but their advantages depend on representation choices, missingness assumptions, privacy requirements, computational budgets, and evaluation protocols. This survey has reviewed the literature from a task-driven and design-oriented perspective, covering mathematical preliminaries, synthesis and augmentation, imputation, trustworthy data synthesis, anomaly detection, benchmark practice, and open research directions. We hope it helps clarify where the field has made progress and where careful work is still needed.

REFERENCES

[1] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang, and L. Hua, "Data mining in healthcare and biomedicine: a survey of the literature," *Journal of medical systems*, vol. 36, pp. 2431–2448, 2012.

[2] M. F. Dixon, I. Halperin, and P. Bilokon, *Machine learning in finance*. Springer, 2020, vol. 1170.

[3] A. Algarni, "Data mining in education," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 6, pp. 456–461, 2016.

[4] S. Anand, P. Padmanabham, A. Govardhan, and R. H. Kulkarni, "An extensive review on data mining methods and clustering models for intelligent transportation system," *Journal of Intelligent Systems*, vol. 27, no. 2, pp. 263–273, 2018.

[5] M. W. King and P. A. Resick, "Data mining in psychological treatment research: a primer on classification and regression trees." *Journal of consulting and clinical psychology*, vol. 82, no. 5, p. 895, 2014.

[6] G. GDPR, "General data protection regulation," *Regulation (EU)*, vol. 679, 2016.

[7] C. S. Legislature, "California consumer privacy act of 2018 (ccpa)," 2018, accessed: 2024-12-27.

[8] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "Tabddpm: Modelling tabular data with diffusion models," in *International Conference on Machine Learning*. PMLR, 2023, pp. 17 564–17 579.

[9] P.-A. Mattei and J. Frellsen, "Miwae: Deep generative modelling and imputation of incomplete data sets," in *International conference on machine learning*. PMLR, 2019, pp. 4413–4423.

[10] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM computing surveys (CSUR)*, vol. 52, no. 4, pp. 1–36, 2019.

[11] V. A. Fajardo, D. Findlay, C. Jaiswal, X. Yin, R. Houmanfar, H. Xie, J. Liang, X. She, and D. B. Emerson, "On oversampling imbalanced data with deep conditional generative models," *Expert Systems with Applications*, vol. 169, p. 114463, 2021.

[12] S. A. Assefa, D. Dervovic, M. Mahfouz, R. E. Tillman, P. Reddy, and M. Veloso, "Generating synthetic data in finance: opportunities, challenges and pitfalls," in *Proceedings of the First ACM International Conference on AI in Finance*, 2020, pp. 1–8.

[13] M. Hernandez, G. Epelde, A. Alberdi, R. Cilla, and D. Rankin, "Synthetic data generation for tabular health records: A systematic review," *Neurocomputing*, vol. 493, pp. 28–45, 2022.

[14] J. You, X. Ma, Y. Ding, M. J. Kochenderfer, and J. Leskovec, "Handling missing data with graph representation learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 075–19 087, 2020.

[15] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International conference on machine learning*. PMLR, 2018, pp. 5689–5698.

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

[17] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang *et al.*, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.

[18] D. P. Kingma, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[21] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.

[22] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. PMLR, 2015, pp. 2256–2265.

[23] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *International Conference on Learning Representations*, 2023.

[24] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[25] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*.

[26] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *International Conference on Learning Representations*, 2020.

[27] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *International Conference on Learning Representations*, 2020.

[28] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, "Argmax flows and multinomial diffusion: Learning categorical distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 454–12 465, 2021.

[29] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. Van Den Berg, "Structured denoising diffusion models in discrete state-spaces," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 981–17 993, 2021.

[30] Z. Xing, Q. Feng, H. Chen, Q. Dai, H. Hu, H. Xu, Z. Wu, and Y.-G. Jiang, "A survey on video diffusion models," *ACM Computing Surveys*, vol. 57, no. 2, pp. 1–42, 2024.

[31] C. Liu, W. Fan, Y. Liu, J. Li, H. Li, H. Liu, J. Tang, and Q. Li, "Generative diffusion models on graphs: methods and applications," in *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2023, pp. 6702–6711.

[32] J. Kim, C. Lee, and N. Park, "Stasy: Score-based tabular data synthesis," in *The Eleventh International Conference on Learning Representations*, 2023.

[33] N. Suh, X. Lin, D.-Y. Hsieh, M. Honarkhah, and G. Cheng, "Autodiff: combining auto-encoder and diffusion model for tabular data synthesizing," in *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*.

[34] C. Lee, J. Kim, and N. Park, "Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis," in *International Conference on Machine Learning*. PMLR, 2023, pp. 18 940–18 956.

[35] H. Zhang, J. Zhang, Z. Shen, B. Srinivasan, X. Qin, C. Faloutsos, H. Rangwala, and G. Karypis, "Mixed-type tabular data synthesis with score-based diffusion in latent space," in *The Twelfth International Conference on Learning Representations*, 2024.

[36] A. Jolicoeur-Martineau, K. Fatras, and T. Kachman, "Generating and imputing tabular data via diffusion and flow-based gradient-boosted trees," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2024, pp. 1288–1296.

[37] A. Guzmán-Cordero, F. Eijkelboom, and J.-W. van de Meent, "Exponential family variational flow matching for tabular data generation," in *Forty-second International Conference on Machine Learning*, 2025.

[38] B. I. Nasution, F. Eijkelboom, M. Elliot, R. Allmendinger, and C. A. Naesseth, "Flow matching for tabular data synthesis," *Transactions on Machine Learning Research*, 2026.

[39] Anonymous, "Tabflowm: Lightweight flow matching for mixed-type tabular data synthesis in latent space," *Submitted to Transactions on Machine Learning Research*, 2026, under review.

[40] M. Mueller, K. Gruber, and D. Fok, "Cascaded flow matching for heterogeneous tabular data with mixed-type features," *arXiv preprint arXiv:2601.22816*, 2026.

[41] R. Branco, M. Gromicho, M. de Carvalho, P. Fariselli, and S. C. Madeira, "Patientflow: Learning to generate mixed-type longitudinal clinical data with flow matching," *Artificial Intelligence in Medicine*, p. 103392, 2026.

[42] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.

[43] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng, and S. Z. Li, "A survey on generative diffusion models," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[44] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 850–10 869, 2023.

[45] Y. Zhu and Y. Zhao, "Diffusion models in nlp: A survey," *arXiv preprint arXiv:2303.07576*, 2023.

[46] L. Lin, Z. Li, R. Li, X. Li, and J. Gao, "Diffusion models for time-series applications: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 25, no. 1, pp. 19–41, 2024.

[47] A. X. Wang, S. S. Chukova, C. R. Simpson, and B. P. Nguyen, "Challenges and opportunities of generative models on tabular data," *Applied Soft Computing*, p. 112223, 2024.

[48] D.-K. Kim, D. Ryu, Y. Lee, and D.-H. Choi, "Generative models for tabular data: A review," *Journal of Mechanical Science and Technology*, vol. 38, no. 9, pp. 4989–5005, 2024.

[49] R. Shi, Y. Wang, M. Du, X. Shen, Y. Chang, and X. Wang, "A comprehensive survey of synthetic tabular data generation," *arXiv preprint arXiv:2504.16506*, 2025.

[50] G. C. N. Kindji, L. M. Rojas-Barahona, E. Fromont, and T. Urvoy, "Tabular data generation models: An in-depth survey and performance benchmarks with extensive tuning," *Neurocomputing*, p. 131655, 2025.

[51] A. Sklar, "Random variables, joint distribution functions, and copulas," *Kybernetika*, vol. 9, no. 6, pp. 449–460, 1973.

[52] D. A. Reynolds *et al.*, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, no. 659-663, 2009.

[53] P. Rabaey, J. Deleu, S. Heytens, and T. Demeester, "Clinical reasoning over tabular data and text with bayesian networks," in *International Conference on Artificial Intelligence in Medicine*. Springer, 2024, pp. 229–250.

[54] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[55] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee, 2008, pp. 1322–1328.

[56] B. Nowok, G. M. Raab, and C. Dibben, "synthpop: Bespoke creation of synthetic data in r," *Journal of statistical software*, vol. 74, pp. 1–26, 2016.

[57] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[58] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in neural information processing systems*, vol. 32, 2019.

[59] T. Liu, Z. Qian, J. Berrevoets, and M. van der Schaar, "Goggle: Generative modelling for tabular data by learning relational structure," in *The Eleventh International Conference on Learning Representations*, 2023.

[60] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, "Large language models: A survey," *arXiv preprint arXiv:2402.06196*, 2024.

[61] V. Borisov, K. Sessler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language models are realistic tabular data generators," in *The Eleventh International Conference on Learning Representations*, 2023.

[62] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[63] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[64] J. Kim, C. Lee, Y. Shin, S. Park, M. Kim, N. Park, and J. Cho, "Sos: Score-based oversampling for tabular data," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 762–772.

[65] S. Zheng and N. Charoenphakdee, "Diffusion models for missing value imputation in tabular data," in *NeurIPS 2022 First Table Representation Workshop*.

[66] L. Lillard, J. P. Smith, and F. Welch, "What do we really know about wages? the importance of nonreporting and census imputation," *Journal of Political Economy*, vol. 94, no. 3, Part 1, pp. 489–506, 1986.

[67] B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan, "Strategies for handling missing data in electronic health record derived data," *Egems*, vol. 1, no. 3, 2013.

[68] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, "A survey on missing data in machine learning," *Journal of Big data*, vol. 8, pp. 1–37, 2021.

[69] D. B. Rubin, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[70] J. Shi, M. Xu, H. Hua, H. Zhang, S. Ermon, and J. Leskovec, "Tabdiff: a unified diffusion model for multi-modal tabular data generation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[71] Z. Li, Q. Huang, Y. Zhu, L. Yang, M. Mohammadi Amiri, N. van Stein, and M. van Leeuwen, "Scalable, explainable and provably robust anomaly detection with one-step flow matching," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[72] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[73] P. E. Kloeden, E. Platen, P. E. Kloeden, and E. Platen, *Stochastic differential equations*. Springer, 1992.

[74] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[75] L. Cui, H. Li, K. Chen, L. Shou, and G. Chen, "Tabular data augmentation for machine learning: Progress and prospects of embracing generative ai," *arXiv preprint arXiv:2407.21523*, 2024.

[76] Y. Ouyang, L. Xie, C. Li, and G. Cheng, "Missdiff: Training diffusion models on tabular data with missing values," in *ICML 2023 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2023.

[77] J. Si, Z. Ou, M. Qu, and Y. Li, "Tabunite: Efficient encoding schemes for flow and diffusion tabular generative models," 2024.

[78] M. Mueller, K. Gruber, and D. Fok, "Continuous diffusion for mixed-type tabular data," 2024.

[79] J. Si, Z. Ou, M. Qu, Z. Xiang, and Y. Li, "Tabrep: Training tabular diffusion models with a simple and effective continuous representation," *Transactions on Machine Learning Research*, 2026.

[80] H. Zhang, L. Fang, Q. Wu, and P. S. Yu, "Tabnat: A continuous-discrete joint generative framework for tabular data," in *Forty-second International Conference on Machine Learning*, 2025.

[81] I. Malashin, I. Masich, D. Martysyuk, A. Gantimurov, A. Borodulin, V. Nelyub, and V. Tynchenko, "A stochastic diffusion framework for tabular data synthesis and completion," in *2025 IEEE International Conference on Data Mining Workshops*. IEEE, 2025, pp. 2227–2236.

[82] H. He, S. Zhao, Y. Xi, and J. C. Ho, "Meddiff: Generating electronic health records using accelerated denoising diffusion model," *arXiv preprint arXiv:2302.04355*, 2023.

[83] T. Ceritli, G. O. Ghosheh, V. K. Chauhan, T. Zhu, A. P. Creagh, and D. A. Clifton, "Synthesizing mixed-type electronic health records using diffusion models," *arXiv preprint arXiv:2302.14679*, 2023.

[84] I. Nicholas, H. Kuo, F. Garcia, A. Sonnerborg, M. Bohm, R. Kaiser, M. Zazzi, L. Jorm, and S. Barbieri, "Synthetic health-related longitudinal data with mixed-type variables generated using diffusion models," in *NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI*, 2023.

[85] H. He, Y. Xi, Y. Chen, B. Malin, J. Ho *et al.*, "A flexible generative model for heterogeneous tabular ehr with missing modality," in *The Twelfth International Conference on Learning Representations*, 2024.

[86] H. Yuan, S. Zhou, and S. Yu, "Ehrdiff: Exploring realistic ehr synthesis with diffusion models," *Transactions on Machine Learning Research*, 2024.

[87] J. Han, Z. Chen, Y. Li, Y. Kou, E. Halperin, R. E. Tillman, and Q. Gu, "Guided discrete diffusion for electronic health record generation," *Transactions on Machine Learning Research*, 2025.

[88] T. Sattarov, M. Schreyer, and D. Borth, "Findiff: Diffusion models for financial tabular data generation," in *Proceedings of the Fourth ACM International Conference on AI in Finance*, 2023, pp. 64–72.

[89] C. Liu and C. Liu, "Entity-based financial tabular data synthesis with diffusion models," in *Proceedings of the 5th ACM International Conference on AI in Finance*, 2024, pp. 547–554.

[90] M. Schreyer, T. Sattarov, A. Sim, and K. Wu, "Imb-findiff: Conditional diffusion models for class imbalance synthesis of financial tabular data," in *Proceedings of the 5th ACM International Conference on AI in Finance*, 2024, pp. 617–625.

[91] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramer, B. Balle, D. Ippolito, and E. Wallace, "Extracting training data from diffusion models," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5253–5270.

[92] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[93] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[94] P. Yadav, M. Steinbach, V. Kumar, and G. Simon, "Mining electronic health records (ehrs) a survey," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–40, 2018.

[95] W. Pang, M. Shafieinejad, L. Liu, and X. He, "Clavaddpm: Multi-relational data synthesis with cluster-guided diffusion models," *Advances in Neural Information Processing Systems*, 2024.

[96] V. Hudovernik, "Relational data generation with graph neural networks and latent diffusion models," in *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024.

[97] W.-C. Lin and C.-F. Tsai, "Missing value imputation: a review and analysis of the literature (2006–2017)," *Artificial Intelligence Review*, vol. 53, pp. 1487–1509, 2020.

[98] D. Jarrett, B. C. Cebere, T. Liu, A. Curth, and M. van der Schaar, "Hyperimpute: Generalized iterative imputation with automatic model selection," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9916–9937.

[99] S. Van Buuren and C. G. Oudshoorn, "Multivariate imputation by chained equations," 2000.
[100] L. Gondara and K. Wang, "Mida: Multiple imputation using denoising autoencoders," in *Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, PAKDD 2018, Melbourne, VIC, Australia, June 3-6, 2018, Proceedings, Part III 22*. Springer, 2018, pp. 260–272.
[101] A. Nazabal, P. M. Olmos, Z. Ghahramani, and I. Valera, "Handling incomplete heterogeneous data using vaes," *Pattern Recognition*, vol. 107, p. 107501, 2020.
[102] Y. Liu, T. Ajanthan, H. Husain, and V. Nguyen, "Self-supervision improves diffusion models for tabular data imputation," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 1513–1522.
[103] M. Villaizán-Vallelado, M. Salvatori, C. Segura, and I. Arapakis, "Diffusion models for tabular data imputation and synthetic data generation," *ACM Transactions on Knowledge Discovery from Data*, vol. 19, no. 6, pp. 1–32, 2025.
[104] A. Wibisono, P. Mursanto, S. See *et al.*, "Natural generative noise diffusion model imputation," *Knowledge-Based Systems*, vol. 301, p. 112310, 2024.
[105] Z. Chen, H. Li, F. Wang, O. Zhang, H. Xu, X. Jiang, Z. Song, and H. Wang, "Rethinking the diffusion models for missing data imputation: A gradient flow perspective," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
[106] H. Zhang, L. Fang, Q. Wu, and P. S. Yu, "Diffputer: An EM-driven diffusion model for missing data imputation," in *The Thirteenth International Conference on Learning Representations*, 2025.
[107] A. Shankar, Y. Wang, R. Hai, and L. Y. Chen, "Harpoon: Generalised manifold guidance for conditional tabular diffusion," in *The Fourteenth International Conference on Learning Representations*, 2026.
[108] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "Csdi: Conditional score-based diffusion models for probabilistic time series imputation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24 804–24 816, 2021.
[109] T. Chen, R. ZHANG, and G. Hinton, "Analog bits: Generating discrete data using diffusion models with self-conditioning," in *The Eleventh International Conference on Learning Representations*.
[110] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, "Revisiting deep learning models for tabular data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021.
[111] P. Mokrov, A. Korotin, L. Li, A. Genevay, J. M. Solomon, and E. Burnaev, "Large-scale wasserstein gradient flows," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 243–15 256, 2021.
[112] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.
[113] A. Shankar, H. Brouwer, R. Hai, and L. Chen, " SiloFuse: Cross-silo Synthetic Data Generation with Latent Tabular Diffusion Models ," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 110–123.
[114] T. Sattarov, M. Schreyer, and D. Borth, "Fedtabdiff: Federated learning of diffusion probabilistic models for synthetic mixed-type tabular data generation," *arXiv preprint arXiv:2401.06263*, 2024.
[115] Z. Yang, H. Yu, P. Guo, K. Zanna, X. Yang, and A. Sano, "Balanced mixed-type tabular data synthesis with diffusion models," *Transactions on Machine Learning Research*, 2025.
[116] T. Sattarov, M. Schreyer, and D. Borth, "Differentially private federated learning of diffusion models for synthetic tabular data generation," *arXiv preprint arXiv:2412.16083*, 2024.
[117] ——, "Privacy preserving diffusion models for mixed-type tabular data generation," *arXiv preprint arXiv:2512.00638*, 2025.
[118] Z. Fang, Z. Jiang, H. Chen, X. Li, and J. Li, "Understanding and mitigating memorization in diffusion models for tabular data," in *Forty-second International Conference on Machine Learning*, 2025.
[119] Z. Fang, Z. Jiang, H. Chen, X. Zhang, K. Tang, X. Li, and J. Li, "A closer look on memorization in tabular diffusion model: A data-centric perspective," *Transactions on Machine Learning Research*, 2026.
[120] G. Zamberg, M. Salhov, O. Lindenbaum, and A. Averbuch, "Tabadm: Unsupervised tabular anomaly detection with diffusion models," *arXiv preprint arXiv:2307.12336*, 2023.
[121] V. Livernoche, V. Jain, Y. Hezaveh, and S. Ravanbakhsh, "On diffusion modeling for anomaly detection," in *The Twelfth International Conference on Learning Representations*, 2024.
[122] S. Li, J. Yu, Y. Lu, G. Yang, X. Du, and S. Liu, "Self-supervised enhanced denoising diffusion for anomaly detection," *Information Sciences*, vol. 669, p. 120612, 2024.
[123] M. Hirth and E. Kasneci, "Anomaly detection by estimating gradients of the tabular data distribution," in *Submitted to The Thirteenth International Conference on Learning Representations*, 2024, under review.
[124] R. Roy, D. Tiwari, and A. Pandey, "Frauddiffuse: Diffusion-aided synthetic fraud augmentation for improved fraud detection," in *Proceedings of the 5th ACM International Conference on AI in Finance*, 2024, pp. 90–98.
[125] Y. Pushkarenko and V. Zaslavskyi, "Synthetic data generation for fraud detection using diffusion models," *Information & Security: An International Journal*, vol. 55, no. 2, pp. 185–198, 2024.
[126] T. Sattarov, M. Schreyer, and D. Borth, "Diffusion-scheduled denoising autoencoders for anomaly detection in tabular data," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 2478–2489.
[127] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, "Adbench: Anomaly detection benchmark," in *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.
[128] J. C. Cresswell and T. Kim, "Scaling up diffusion and flow-based xgboost models," *arXiv preprint arXiv:2408.16046*, 2024.