

Creation begins with understanding: LLMs as strategy designers for privacy-preserving tabular data synthesis

Dandan Guo

guodandan@jlu.edu.cn

Jilin University <https://orcid.org/0009-0005-2278-498X>

Jinmeng Li

Jilin University

Quan Zhang

Michigan State University <https://orcid.org/0000-0001-9260-4618>

Hangting Ye

Jilin University

He Zhao

Commonwealth Scientific and Industrial Research Organisation (CSIRO)

Firas Laakom

King Abdullah University of Science and Technology (KAUST)

Juergen Schmidhuber

King Abdullah University of Science and Technology (KAUST)

Article

Keywords:

Posted Date: October 15th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10869884/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: There is **NO** Competing Interest.

Creation begins with understanding: LLMs as strategy designers for privacy-preserving tabular data synthesis

Jinmeng Li¹, Quan Zhang², Hangting Ye¹, He Zhao³,
Firas Laakom⁴, Dandan Guo^{1,4,*} and Jürgen Schmidhuber^{4,5}

¹School of Artificial Intelligence, Jilin University, Changchun, China

²Broad College of Business, Michigan State University, East Lansing, USA

³Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia

⁴Center of Excellence for Generative AI, King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

⁵The Swiss AI Lab, IDSIA-USI/SUPSI, Lugano, Switzerland

*Corresponding author. E-mail: guodandan@jlu.edu.cn

Abstract

Sharing tabular data in high-stakes domains is constrained by privacy regulations. Synthetic data offer a promising alternative, but deep generative models are costly to train and difficult to audit, while LLM-based methods often serialize records as text, obscuring tabular structure and exposing sensitive data. We introduce Tabular Synthesis Strategy Designer (TabSSD), which uses an LLM to design synthesis procedures rather than directly generate records. TabSSD provides the LLM with tree-derived summaries of variable dependence rather than raw records, which produces Python programs for local execution and evaluation. Across twelve datasets, TabSSD strikes a favourable balance among statistical fidelity, predictive utility, and empirical privacy risk, achieving the best average rank across six metrics among ten methods. Moreover, it substantially reduces local computation and token consumption relative to the compared methods. By enabling human-guided refinement and eliminating user-side model tuning, TabSSD lowers the expertise and infrastructure barriers to transparent tabular data synthesis.

Introduction

Tabular data, which combine heterogeneous (e.g., numerical and categorical) variables, underpin applications in various domains such as healthcare and physical systems [1, 2]. Collaboration and data sharing in these settings are increasingly constrained by stringent data-protection regulations, including the European Union’s General Data Protection Regulation [3]. Tabular data synthesis offers a promising solution for data use without releasing sensitive records. High-quality synthetic data must balance statistical fidelity, downstream utility, and privacy: they should preserve the distributions and variable relationships of the original data so that models trained on them remain effective and generalizable, while minimizing memorization and re-identification risks [4, 5]. In safety-critical applications, the synthesis process should also be transparent and auditable, enabling domain experts to identify potential failure modes and determine whether the resulting data are suitable for downstream use.

Existing synthesis approaches, however, face limitations arising from their underlying modelling paradigms. Classical statistical methods [6–8] provide interpretable and principled mechanisms for tabular synthesis, but the reliance on distributional assumptions can limit their effectiveness for high-dimensional data with strong variable dependence [9, 10]. Deep generative models, including generative adversarial networks [11–13] and diffusion models [14–16], offer greater modelling flexibility but encode data distributions implicitly within neural network parameters (Fig. 1a). Their generation mechanisms are therefore difficult to inspect or audit. In high-stakes domains, this opacity limits experts’ ability to understand the generation logic, diagnose failure modes, and correct undesirable behaviours [17, 18]. Moreover, adapting these models to new data typically requires computationally intensive retraining.

Large language models (LLMs) have recently been adapted for tabular tasks [19, 20], but using them directly as record-level generators for tabular synthesis creates a modality mismatch: LLMs are optimized

for sequential text, whereas tabular records are heterogeneous, non-sequential, and order-invariant [21, 22]. Existing approaches commonly address this mismatch by serializing rows into text sequences (Fig. 1b). Such representations obscure variable dependence and impose an artificial order that does not reflect the intrinsic structure of tabular data [21]. Methods based on serialized records generally rely on either task-specific fine-tuning [23, 24] or in-context learning [25, 26]. Fine-tuning is data-intensive and computationally expensive, and in-context learning must infer the joint distribution from a limited set of examples and incurs increasing context-window and token costs as more examples are provided. Record serialization can also expose raw data during LLM adaptation or prompting, introducing additional privacy risks [27, 28]. These limitations motivate a different paradigm in which the LLM reasons over summarized data structure rather than generating records directly from raw examples.

We propose the Tabular Synthesis Strategy Designer (TabSSD), which shifts the role of the LLM from a record-level generator to a high-level strategy designer. TabSSD draws on conditional modelling in statistical practice (Fig. 1c), where a complex joint distribution is decomposed into a sequence of conditional models. To represent tabular structure without exposing raw records, TabSSD uses chained classification and regression trees (chained trees, hereafter) as a language-compatible bridge. These trees capture variable dependence through decision rules [29, 30], compressing variable-dependence structure into compact prompts. Guided by these prompts, the LLM identifies redundant predictive dependence, determines sampling orders, selects appropriate conditional samplers, and expresses the resulting synthesis strategy as executable Python code (Fig. 1d). Because the synthesis logic is represented as inspectable, executable code, domain experts can audit its rules, diagnose empirical anomalies, and make targeted revisions without computationally expensive retraining. By automating dependence extraction, strategy generation, local validation, and candidate selection, TabSSD lowers the expertise and computational requirements of tabular synthesis. Through a standardized interface, it produces locally executable and inspectable programs without requiring users to design deep generative models, fine-tune LLMs, perform extensive hyperparameter tuning, or rely on local GPU acceleration.

We evaluate TabSSD on twelve heterogeneous datasets spanning a range of dimensionalities and sample sizes. Compared with representative deep generative models and LLM-based methods, TabSSD achieves a favourable balance among statistical fidelity, downstream utility, and privacy while substantially reducing computational and token costs. A case study on a severely imbalanced clinical dataset further demonstrates its practical use for minority-outcome augmentation and assesses whether the generated data preserve clinically meaningful phenotype distributions beyond conventional statistical metrics. Overall, these results show that strategy-level LLM reasoning provides a transparent and efficient approach to privacy-preserving tabular data synthesis and may extend to other heterogeneous data modalities.

Results

Overview of TabSSD

As shown in Fig. 1d, TabSSD is an end-to-end framework that converts tabular data into executable synthesis strategies. It first extracts predictive dependence among variables using chained trees and encodes this information in a compact prompt. The LLM then analyses the dependence structure and designs multiple candidate strategies, each implemented as a Python program. These programs run locally and undergo evaluation for statistical fidelity, empirical privacy risk, and computational efficiency. The selected program synthesizes the data and provides an inspectable generation procedure. This design shifts the role of the LLM from record-level generation to strategy design. Because the LLM receives summaries of the variable-dependence structure rather than training records, the original data remain in the local environment throughout synthesis and evaluation. TabSSD runs on CPU only and is summarized as Algorithm 1.

Datasets, baselines, evaluation, and settings

We assess TabSSD on 12 real-world datasets and in a clinical data-augmentation experiment. Supplementary Table 1 provides the data descriptions. We compare TabSSD with methods that span several synthesis paradigms: tabular deep generative models (CTGAN [31] and TVAE [31]), diffusion-based models (TabDPM [32], TabDiff [33], and TABSYN [34]), a fine-tuned LLM using record serialization (GReaT [23]), tree- and energy-based models (ARF [35] and NRGBoost [36]), and the interpolation method SMOTE [7]. Each method uses the same train-test splits and generates as many synthetic samples as the corresponding training set contains. When required, validation uses only the training data. In the main experiments, TabSSD queries Gemini-2.5-Pro 10 times per dataset to generate candidate strategies. All local stages of TabSSD, including tree fitting, candidate execution, strategy evaluation, and final synthesis, run on CPU. Among the baselines, the tree-based methods ARF and NRGBoost run on CPU, and others use GPU acceleration

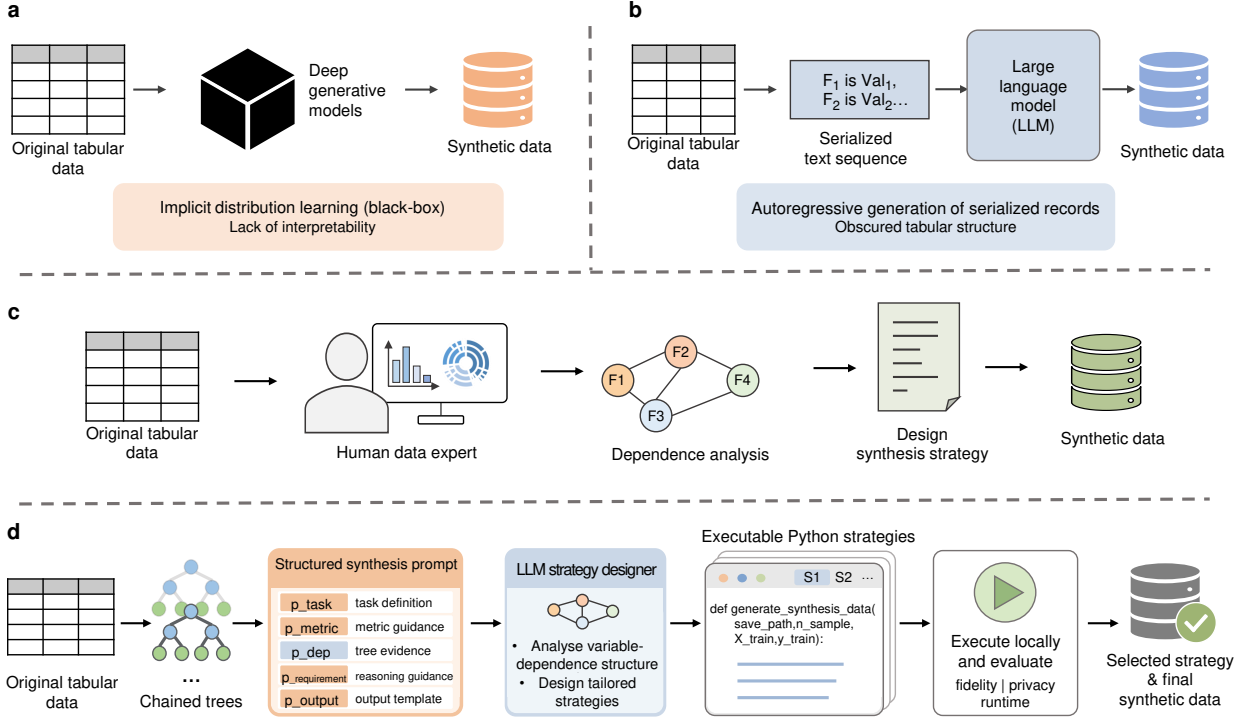


Fig. 1 Comparison of tabular data synthesis paradigms and overview of the proposed TabSSD framework. **a**, Conventional deep generative models encode data distributions implicitly within trained parameters, resulting in opaque generation mechanisms that are difficult to inspect or audit. **b**, Existing LLM-based methods serialize tabular records into one-dimensional text sequences, obscuring variable dependence, increasing computational overhead, and exposing raw records in prompts. **c**, Human experts instead analyse variable dependence and domain constraints to design tailored synthesis strategies. **d**, TabSSD similarly positions the LLM as a strategy designer. Chained trees extract the variable-dependence structure from the original data. This information is distilled and combined with the task definition, evaluation criteria, reasoning instructions, and output specification, forming a prompt. Without accessing raw records, the LLM analyses the variable-dependence structure and produces multiple executable Python strategies. These candidates are executed and evaluated locally for statistical fidelity, empirical privacy risk, and runtime. Finally, the selected strategy is used to generate synthetic data.

because their running on CPU is prohibitively slow. We follow the scheme of TABSYN [34] to report results averaged over 20 randomly sampled synthetic data. Supplementary Note A.2 provides the implementation details. Across all datasets, we use the same prompt template, validation procedure, and selection criteria. No dataset-specific synthesis has been designed, tuned, or selected manually.

We evaluate the synthetic data from four perspectives: statistical fidelity, downstream predictive utility, empirical privacy risk, and computational efficiency. We additionally examine inspectability and editability through code-level case studies. Statistical fidelity is measured by marginal distribution error, pairwise correlation error, and α -precision, quantifying how well the synthetic data preserve the empirical properties of the real training data. Under the train-on-synthetic, test-on-real protocol, AdaBoost [37], Random Forest [38], and XGBoost [39] are used as downstream models to measure predictive utility, and we report their mean performance. Distance to closest record (DCR) measures empirical memorization risk, and δ -presence measures empirical re-identification risk. Computational efficiency is reflected by the time required for the entire data synthesis process. The code-level case studies examine whether the resulting procedures are inspectable and editable. Supplementary Note A.3 defines the evaluation metrics.

Synthetic data fidelity, utility, and privacy risks

We compare TabSSD with the baseline synthesis methods on 12 real-world datasets and examine statistical fidelity, downstream predictive utility, and empirical privacy risk. Fig. 2 summarizes the trade-offs: TabSSD combines low statistical errors and competitive predictive utility with a favourable empirical privacy profile. Supplementary Tables 2–7 report the complete numerical results.

Statistical fidelity and variable dependence. SMOTE attains high statistical fidelity and predictive utility through linear interpolation between two data records. This mechanism preserves local geometry but produces the least favourable DCR result, indicating higher memorization risk and limiting its applicability for privacy-preserving data sharing [40]. We therefore focus the performance analysis and comparison on the deep generative and LLM-based methods. Among these methods, TabSSD has the second-lowest mean marginal distribution error (3.80%), close to TabDiff (3.74%; Fig. 2a). TabSSD also has the lowest mean

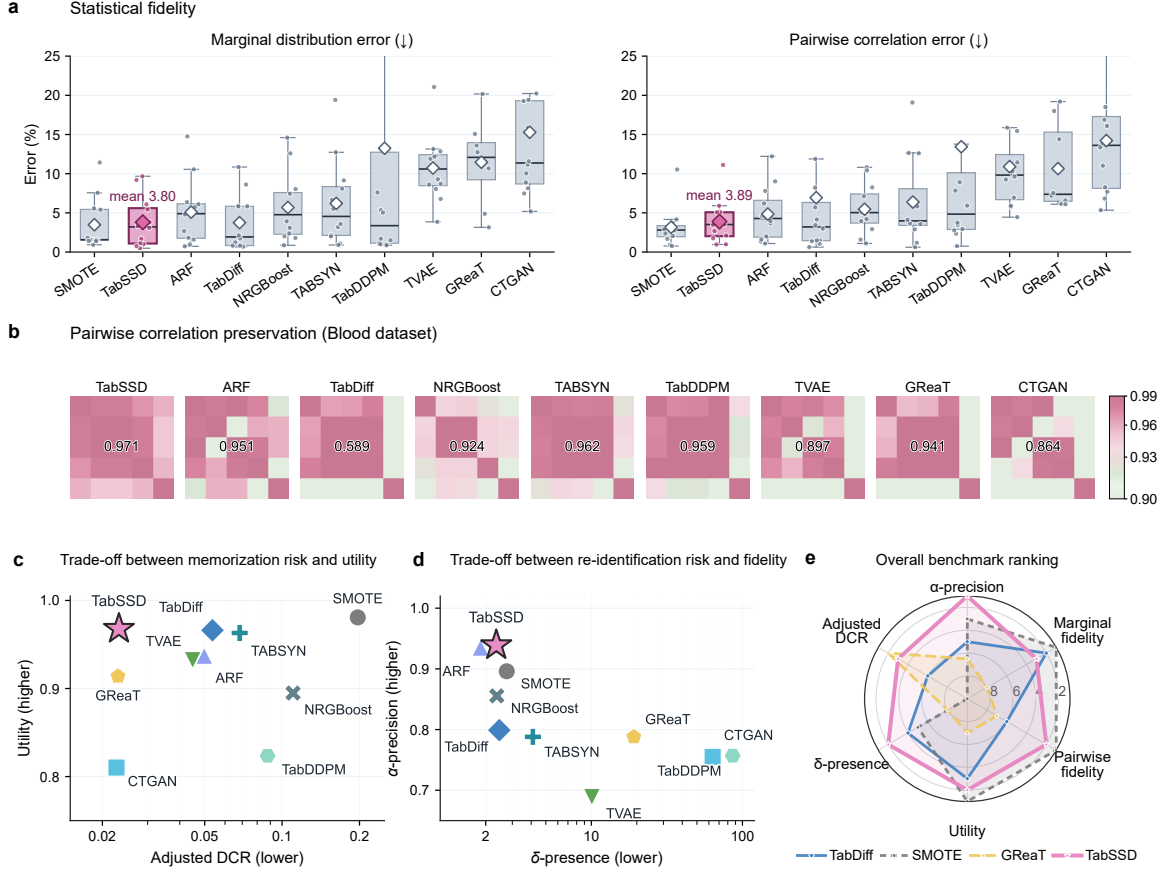


Fig. 2 Statistical fidelity, predictive utility, and empirical privacy risk of representative tabular data synthesis methods. **a**, Statistical fidelity on 12 datasets, measured by marginal distribution error and pairwise correlation error. Lower values indicate higher fidelity. Boxes span the interquartile range, centre lines show the median, whiskers extend to 1.5 times this range, diamonds show the arithmetic mean, and dots denote performance on individual datasets. Because GReaT serializes tabular records as a text sequence, inputs from four high-dimensional datasets (*SpeedDating*, *Jasmine*, *DNA*, and *DARWIN*) exceed the LLM’s context window; the GReaT results cover the remaining eight datasets. TabSSD is comparable to the best-performing SMOTE and better than the generative baselines. **b**, Pairwise correlation preservation on the *Blood* dataset. Each cell reports the preservation score for a variable pair, and the value in the centre reports the mean across all pairs. Higher values indicate better performance. **c**, Trade-off between downstream utility and DCR. A lower adjusted DCR indicates less proximity to training records and lower potential memorization risk. **d**, Trade-off between δ -presence and α -precision. Lower δ -presence indicates lower empirical re-identification risk, and higher α -precision indicates higher sample-level fidelity. Panels c and d report the performance averaged across datasets with available results. Methods on the top left are preferred, and TabSSD is on the Pareto front in both comparisons. **e**, Ranking profiles of TabSSD, TabDiff, SMOTE, and GReaT across marginal fidelity, pairwise fidelity, utility, adjusted DCR, α -precision, and δ -presence. For each metric, all ten evaluated methods are ranked in the preferred direction, with rank 1 denoting the best performance. The ranks are mapped such that better performance appears closer to the outer ring. Across the six metrics, TabSSD ranks third, second, second, third, first, and second, respectively, yielding an average rank of 2.17. Among all methods compared, TabSSD exhibits the strongest and most balanced overall ranking profile.

pairwise correlation error (3.89%), a relative reduction of 19.79% from ARF (4.85%), the second-best method compared, and 43.95% from TabDiff (6.94%). On the *Blood* dataset, TabSSD achieves a mean pairwise correlation preservation score of 0.971, better recovering variable correlations than the other generative methods (Fig. 2b).

Fidelity–utility–privacy trade-offs. High statistical fidelity and predictive utility can be achieved by sacrificing privacy protection. Fig. 2c and Fig. 2d show that TabSSD is on the Pareto front of the utility–memorization and fidelity–re-identification comparisons, respectively. TabSSD combines competitive predictive utility and leading α -precision with memorization and re-identification risks that are comparable to or lower than those of the strongest baselines. Among other well-performing methods, GReaT and CTGAN have a favourable adjusted DCR but lower predictive utility and α -precision, whereas ARF and TabDiff have favourable δ -presence but a lower utility and/or a less favourable adjusted DCR than TabSSD. Fig. 2e summarizes the six metrics in a rank-based radar plot. Each metric is ranked across the evaluated methods, with better ranks plotted closer to the outer boundary. TabSSD achieves the best average rank of 2.17 across the six metrics and is near the outer boundary, indicating its balanced performance.

More importantly, TabSSD mitigates a source of exposure not captured by these empirical metrics: raw records are not transmitted to the LLM during strategy generation. The LLM receives abstract partition rules rather than raw records, while tree extraction and synthesis execution remain within the data holder’s environment. Ablation studies support the contributions of tree-based dependence extraction, compact dependence encoding, and explicit prompt guidance (see Supplementary Tables 8–10). In addition, the framework remains applicable across the evaluated LLM backbones, although synthesis performance varies across models and datasets (see Supplementary Table 11).

Scalability and token efficiency

Computing efficiency and scalability. We assess computational scalability across 12 datasets that vary in training data size N and variable dimension $J + 1$ (J features and 1 label) and report method efficiency in Fig. 3. TabSSD requires a 199.6-second end-to-end runtime averaged across datasets. ARF, the fastest generative baseline compared, requires 332.6 seconds, or 1.7 times the TabSSD runtime (Fig. 3a). The other baselines require 3.4 to 38.4 times the TabSSD runtime. Notably, the reported runtimes of the deep generative baselines are obtained with GPU acceleration; on CPUs, these methods are prohibitively slow. By contrast, TabSSD’s local execution uses CPU only. The runtime decomposition in Fig. 3b attributes 77.1% of TabSSD runtime to LLM querying and 22.9% to tree fitting and synthesis execution. Fig. 3c presents dataset-level runtime ratios, calculated as baseline method runtime divided by TabSSD runtime. Values above 1 indicate that TabSSD is faster. Overall, TabSSD is competitive in terms of computing speed, and its largest runtime advantages occur on high-dimensional data. Following established empirical computational-complexity analyses [41], we fit a log-linear model, $\log T = \alpha + \beta_N \log N + \beta_J \log(J + 1)$, to TabSSD runtime and the geometric mean runtime of the baselines. The baseline runtime has a significant positive association with training data size ($\beta_N = 0.48$), whereas TabSSD exhibits an insignificant, near-zero association ($\beta_N = 0.04$; Fig. 3d). Within the evaluated datasets, TabSSD runtime shows no detectable association with training-data size ($\beta_N = 0.04$, $P = 0.51$). This finding is consistent with the use of compact tree summaries instead of serializing records. Variable dimensionality has similar associations with TabSSD and baseline runtime ($\beta_J = 0.30$ and 0.37 , respectively), but TabSSD retains a lower absolute runtime (Fig. 3e). Supplementary Table 12 reports the results for each dataset.

Token efficiency and context management. We compare TabSSD with record serialization, a common approach used by LLM-based tabular synthesis methods. On low-dimensional datasets, TabSSD and serialization have comparable token consumption. As the variable dimension increases, however, serialization becomes increasingly costly and requires up to 130.3 times as many tokens as TabSSD (Fig. 3f). TabSSD’s token efficiency stems from its use of compact chained-tree rules rather than raw record and variable values. For high-dimensional data, a greedy set-cover algorithm selects a subset of trees that covers all the variables (Algorithm 1). Thus, increasing the data dimension adds only the rules needed to include previously uncovered variables, rather than replicating the entire dataset in the prompt. For example, a coverage of *DARWIN*’s 451 variables requires rules from only 54 trees. By separating dependence extraction from prompt construction, TabSSD limits token growth and avoids the context-window failures that are often observed with record serialization.

Transparent data synthesis and human-in-the-loop auditing

From variable dependence to transparent code synthesis. TabSSD produces inspectable and executable Python programs. We illustrate this procedure using *Adult*, a census dataset commonly used to predict whether annual income exceeds 50,000 U.S. dollars from demographic, educational, employment, and financial variables [42]. To reduce redundancy across the chained trees (Supplementary Fig. 1), the LLM distills the extracted variable dependence into a directed graph. As shown in Fig. 4a, the graph designates **education.num**, **relationship**, and **age** as root variables, maps **education** deterministically from **education.num**, and assigns an appropriate estimator to each remaining variable—for example, a regression tree for **capital.gain**. TabSSD renders the resulting strategy as executable Python code, which is shown in Fig. 4b with details provided in Supplementary Note D. This enables domain experts to inspect, audit, and modify the synthesis logic. Notably, the LLM does not receive the original variable names; the variable names restored in the figure are solely for illustration.

Human-in-the-loop interventions. Generative models can inherit biases in their training data, like sex-related disparities in *Adult*. TabSSD’s generated scripts support direct code-level interventions without tree retraining or prompt editing. For example, if a user wants to reduce gender bias in their synthetic data to be shared, they can remove **sex** from the predictors for **occupation** and add group-wise probability calibration for **income** (Fig. 4c) without regenerating the data synthesis strategy. We assess the original and edited strategies using Cramér’s V for the association between **sex** and **occupation**, the absolute demographic-parity difference ($|\text{DPD}|$) between female and male high-income rates, and the counterfactual

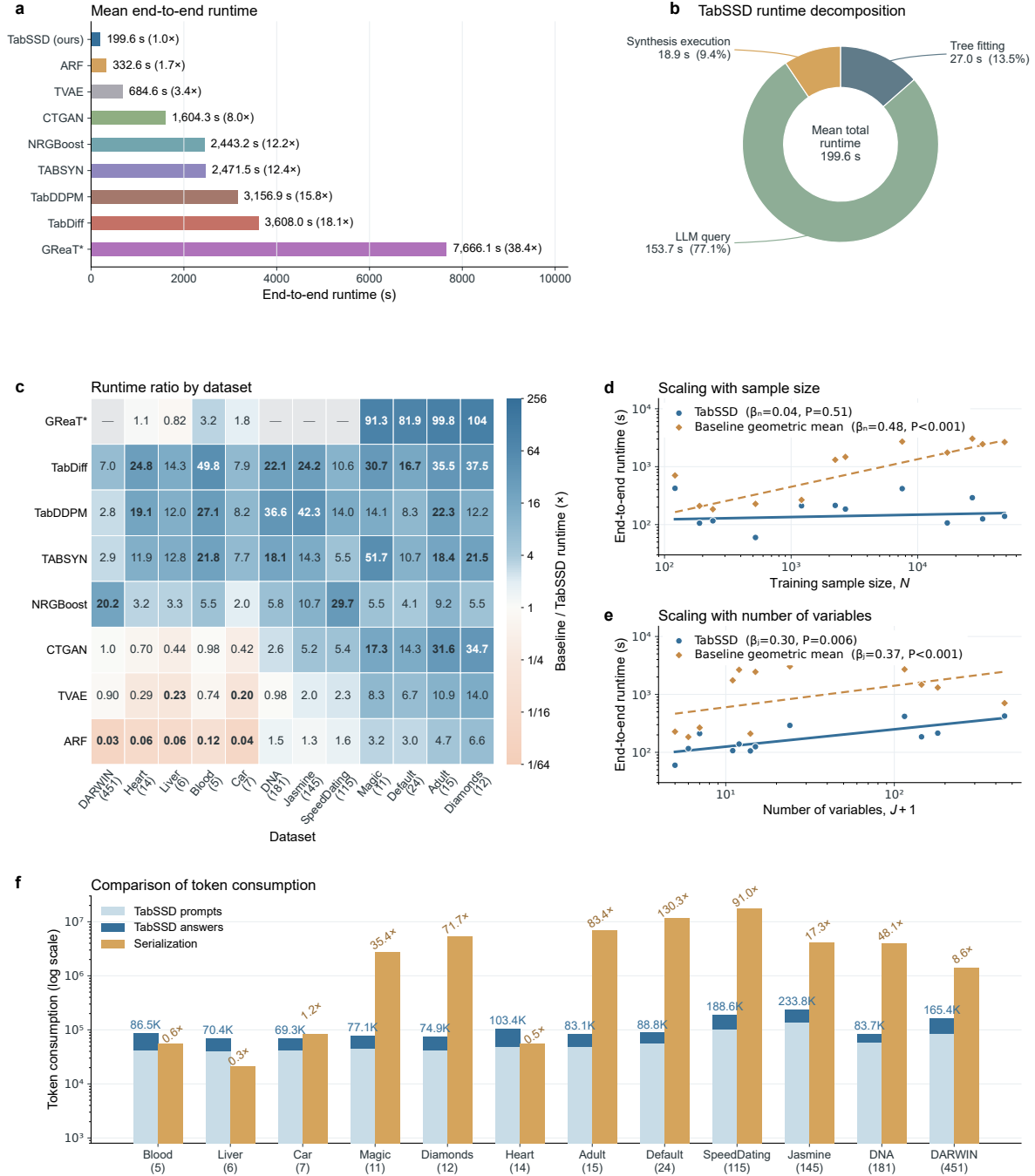
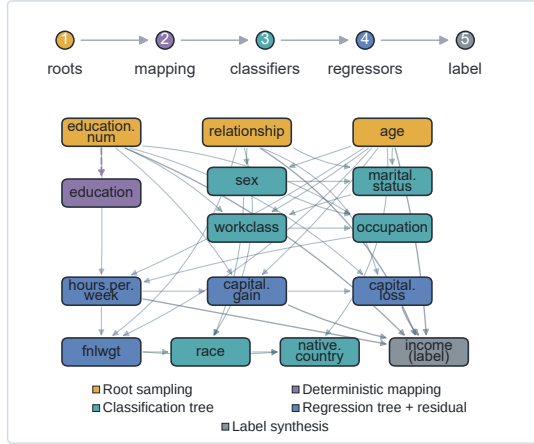


Fig. 3 Scalability and efficiency. Deep generative baselines and the serialization-based method GReaT use GPU acceleration. The runtime of TabSSD reported in **a** to **e** is calculated as the sum of the tree-fitting time and the parallel execution time of ten LLM queries and local syntheses. **a**, End-to-end runtime of data synthesis averaged across datasets. Parenthetical values indicate runtime relative to TabSSD. **b**, Decomposition of TabSSD’s mean end-to-end runtime into tree fitting, LLM querying, and synthesis execution. **c**, Ratios of baseline runtime to TabSSD runtime on the 12 datasets. Values greater than 1 indicate that TabSSD is faster; dashes indicate unavailable results of GReaT due to the LLM’s context window constraint. Datasets are ordered by training-set size in ascending order. Parenthetical values below dataset names indicate the number of variables $J + 1$. **d**, End-to-end runtime as a function of training-data size N . Each point represents one dataset, with TabSSD compared against the geometric mean of the compared baselines. **e**, End-to-end runtime as a function of the number of variables $J + 1$. In **d** and **e**, lines show trends estimated using a log-linear model, $\log T = \alpha + \beta_N \log N + \beta_J \log(J + 1)$. The coefficients β_N and β_J quantify the associations of runtime with training-set size and variable dimension, respectively. **f**, Token consumption on a logarithmic scale. Stacked blue bars show the ten-query token consumption of TabSSD prompts and answers. Orange bars show the tokens required for one-time training data serialization and new data generation. Values above the TabSSD bars report total prompt-and-answer token consumption, and values above the serialization bars report token consumption relative to TabSSD. Parenthetical values below dataset names in **c** and **f** indicate the number of variables $J + 1$. In **a** and **c**, asterisks indicate that GReaT results are based on the eight datasets that fit within the LLM context window.

a Dependence-guided synthesis graph



b Executable strategy snippet

```
01 def generate_synthesis_data(...):
02     df['subspace'] = route_by_label_tree(df)
03     for sub_id in range(4):
04         sub_df = df[df.subspace == sub_id]
05         # root sampling
06         roots = sample(sub_df, root_features)
07         # deterministic mapping
08         education = map_model(roots['education.num'])
09         # categorical classifiers
10         sex = fit_sample_cat(['relationship', 'age'])
11         occupation = fit_sample_cat(
12             ['workclass', 'education.num', 'sex'])
13         # regression + residual sampling
14         hours = fit_sample_cont(
15             ['age', 'education.num', 'occupation'])
16         capital_gain = fit_sample_cont(
17             ['education.num', 'age', 'hours'])
18         income = fit_sample_cat(label_predictors)
19         append(sub_gen)
```

c Strategy editing for improved fairness

Original strategy

```
01 occupation_predictors = [
02     "workclass", "education.num",
03     "sex",
04 ]
05 occupation = fit_sample_cat(
06     "occupation", occupation_predictors
07 )
08
09 income_p = label_model.predict_proba(
10     income_predictors
11 )[:, positive_col]
12 income = sample(income_p)
```

Revised strategy

```
01 occupation_predictors = [
02     "workclass", "education.num",
03 ]
04 occupation = fit_sample_cat(
05     "occupation", occupation_predictors
06 )
07
08 income_p = label_model.predict_proba(
09     income_predictors
10 )[:, positive_col]
11 income_p = calibrate_by_group(
12     income_p, sex, strength=0.50
13 )
14 income = sample(income_p)
```

d Improved parity with slight fidelity change

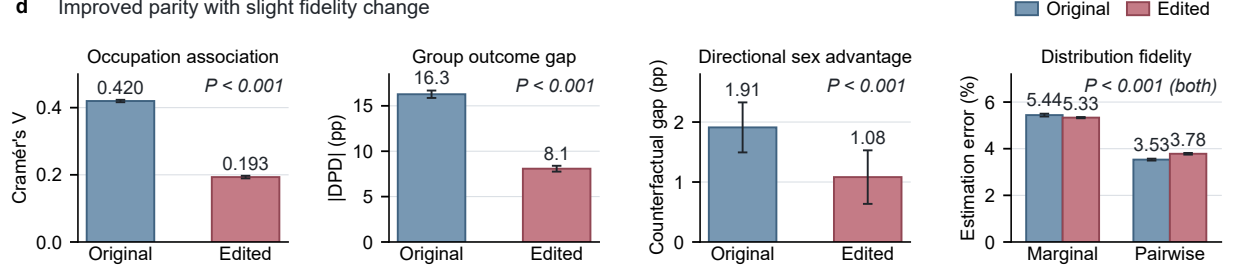


Fig. 4 Distilling variable dependence into executable synthesis strategies with human-in-the-loop fairness editing. **a**, Variable-dependence graph distilled by TabSSD for the *Adult* dataset. Nodes denote variables, directed edges encode the predictive conditional dependence used by the synthesis program, and node colours indicate the models assigned to variables. **b**, An excerpt from the executable synthesis strategy generated by TabSSD. Each record is routed to a tree-induced subspace, within which variables are synthesized with the models and predictors specified by the distilled dependence graph. **c**, Human-in-the-loop editing of the executable strategy. **Sex** is removed from the predictors of **Occupation** and sex-aware calibration is used in the generated probability of high income. The edited strategy executes locally and requires no additional LLM query. **d**, Fairness and fidelity comparison between the original and edited strategies. Fairness is quantified using Cramér's *V* for the association between **Sex** and **Occupation**, the absolute demographic-parity difference in the generated high-income outcome, and the directional counterfactual sex gap estimated by a random-forest income auditor. Fidelity is quantified using marginal-distribution and pairwise-correlation errors. Lower values indicate better parity or higher fidelity. Bars and error bars show the mean and s.d., respectively. *P* values are reported for two-sided paired *t*-tests.

sex gap obtained by changing **sex** while holding all other attributes fixed (Fig. 4d). Lower values indicate better parity. The edit reduces the mean sex–occupation association from 0.420 to 0.193, the mean absolute DPD from 16.3 to 8.1 percentage points, and the mean counterfactual sex gap from 1.91 to 1.08 percentage points, while largely preserving overall statistical fidelity. This experiment illustrates code-level editability rather than prescribing a general fairness intervention.

Code diagnosis and domain-knowledge injection. The executable representation further enables targeted diagnosis and the direct incorporation of domain knowledge. Users can correct empirically identified errors at

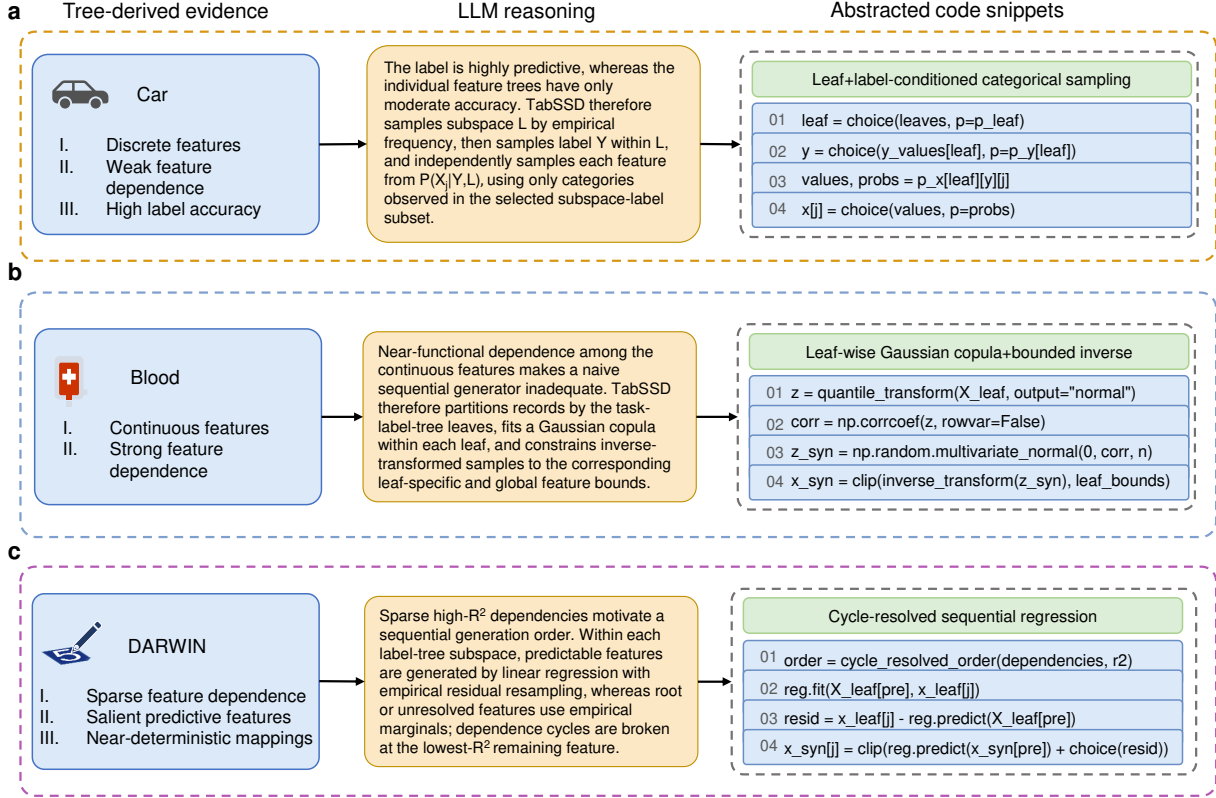


Fig. 5 Dataset-adaptive synthesis strategies guided by variable types and dependence from chained trees. **a**, For *Car*, weak inter-feature dependence coupled with strong label predictability leads to categorical sampling conditioned on both the label-tree leaf and class label. **b**, For *Blood*, strong dependence among continuous variables motivates a leaf-wise Gaussian copula followed by a bounded inverse transformation. **c**, For *DARWIN*, sparse dependence combined with near-deterministic relationships motivates cycle-resolved sequential regression with empirical residual resampling.

specific lines of Python code (Supplementary Table 13), impose logical constraints such as mutually exclusive medical diagnoses, and insert custom `if-else` statements to construct targeted stress-test scenarios. By combining variable dependence learned from the data with explicit expert rules, TabSSD makes the synthesis process transparent, auditable, and readily editable.

Tailored data synthesis strategies

We examine three datasets with different structures: *Car*, which contains only categorical features; *Blood*, which contains only continuous features; and *DARWIN*, a high-dimensional, small-sample dataset. The same prompting and validation pipeline is used to produce synthesis strategies. Fig. 5 shows that TabSSD adapts its synthesis procedure to each dataset rather than applying a fixed template. Specifically, TabSSD combines empirical sampling, node-conditional categorical sampling, Gaussian copulas, and cycle-resolved sequential regression.

For *Car*, the chained trees show weak feature dependence but high label predictability. TabSSD first samples a label within a selected leaf according to its empirical distribution and then samples each feature conditional on the leaf and label. For *Blood*, the variables exhibit near-deterministic functional dependence. TabSSD fits a Gaussian copula within each label-tree leaf, transforms the samples back to the original scale, and constrains them to the corresponding variable ranges. For *DARWIN*, variable dependence is sparse. TabSSD uses predictive variable dependence and the associated R^2 values to establish a sequential generation order, breaking a dependence cycle at the feature with the smallest R^2 . Within each label-tree leaf, ordinary least squares regression with empirical residual resampling generates predictable features. Leaf-level or global empirical distributions supply root and unmodelled features, and the leaf-level empirical distribution supplies labels. Supplementary Note I provides the corresponding Python implementations.

Clinically robust augmentation under class imbalance

We examine TabSSD for data augmentation to address imbalanced classification in a clinically motivated setting. Specifically, we investigate whether TabSSD-generated data can improve minority identification

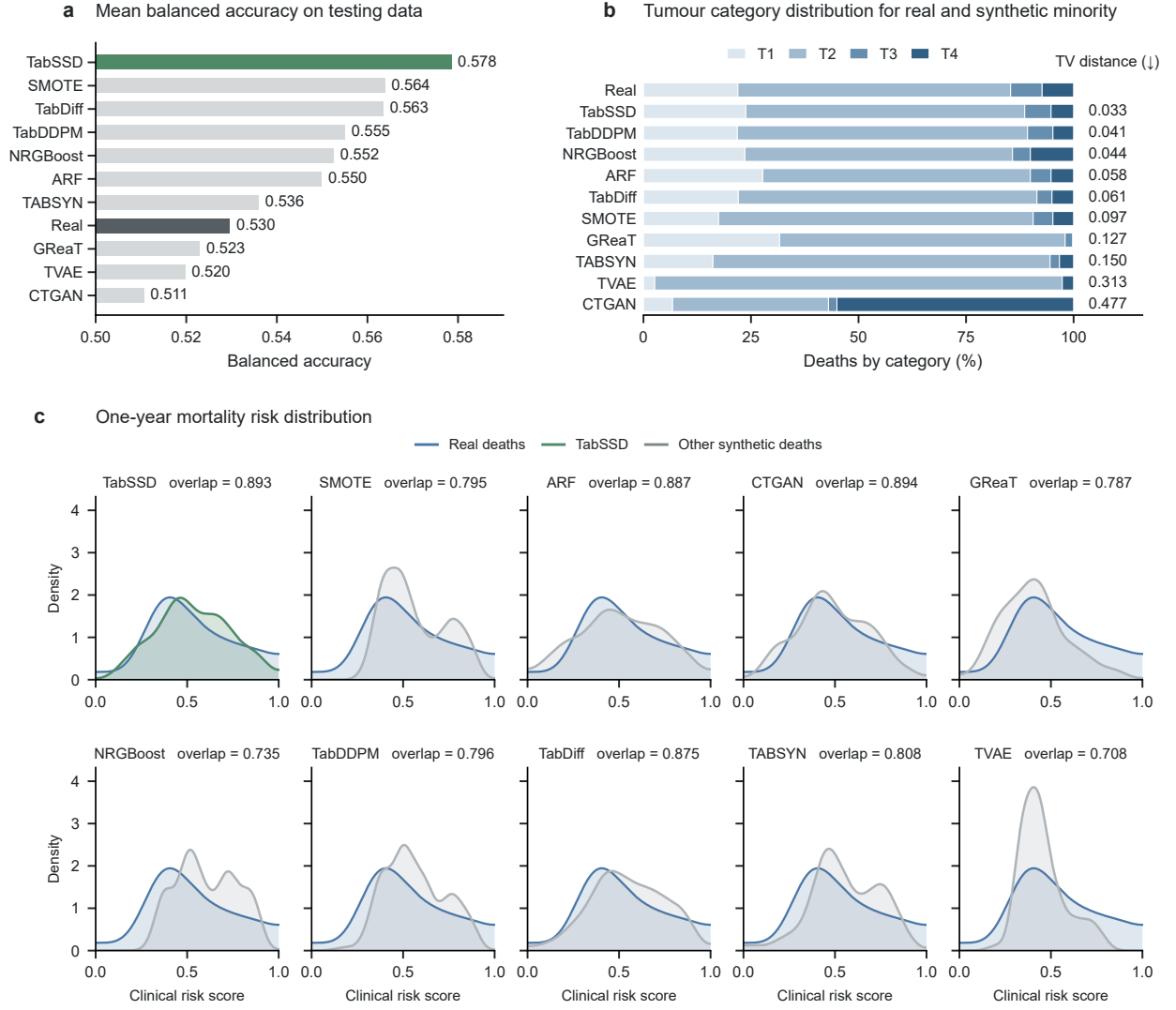


Fig. 6 Clinical phenotype preservation and mortality-prediction utility under 2x augmentation in the Thoracic Surgery case study. **a**, Mean balanced accuracy on test data. AdaBoost, random forest, and XGBoost are trained on augmented training data, and the results are averaged across the three classifiers. **b**, Tumour category distributions among 41 real deaths in training data and 458 synthetic deaths for each synthesis method. Lower total-variation (TV) distance indicates better recovery of the real tumour phenotype distribution among the minority. **c**, Cross-fitted one-year mortality-risk distributions for the real and synthetic data. Five class-weighted logistic-regression models are cross-fitted on the real training data; real records receive out-of-fold scores, and synthetic records receive scores averaged across the five models. Higher density overlap indicates better recovery of the minority risk profile.

under class imbalance while preserving the clinical phenotype distribution. We conduct this evaluation using the *Thoracic Surgery* dataset [43] that comprises 470 patients, including 70 deaths within one year, and 16 predictors. Death within one year serves as the binary outcome, with deceased patients forming the minority class. We use 270 samples, including 41 deaths, for training, and the other 200, including 29 deaths, for testing. Each data synthesis method generates 458 synthetic samples per class, corresponding to twice the number of the majority in the training data, and data synthesis augments the training data. We show representative results in Fig. 6.

Fig. 6a shows that TabSSD achieves the highest mean balanced accuracy (0.578) among the evaluated methods, using AdaBoost, random forest, and XGBoost as classifiers. This result indicates that TabSSD is effective for data augmentation under class imbalance, improving minority-class detection. We defer detailed results to Supplementary Table 14. We next assess synthetic data’s preservation of the tumour category distribution. A tumour category (T1, T2, T3, or T4) represents the local extent of invasion and relates directly to disease severity. As shown in Fig. 6b, TabSSD most closely reproduces the observed tumour-category distribution with a total-variation distance of 0.033, achieving the best distribution recovery among the compared methods. It retains the predominance of T2 while preserving the less frequent T3 and T4.

Finally, we evaluate whether the synthetic data generated by TabSSD and the baseline methods preserve the predictive characteristics of the minority class. Specifically, we examine how closely the synthetic records

corresponding to patients who die within one year recover the distribution of one-year mortality risk observed in real data. We estimate these risks using class-weighted logistic regression trained on the real data. To obtain out-of-sample predictions for the real records, we perform five-fold cross-fitting; four folds are used for model training and one fold for prediction. A synthetic record’s prediction is averaged over the five models. We then compare the risk-score distributions of the real and synthetic minority-class records using the overlap coefficient, which measures the shared area between their density curves; values closer to 1 indicate better recovery of the real data. Fig. 6c shows that TabSSD achieves an overlap coefficient of 0.893, very close to the best-performing CTGAN (0.894), demonstrating that TabSSD well preserves the data distribution and delivers strong predictive utility.

Discussion

In this work, we introduce TabSSD, which repositions LLMs from record-level data generators to synthesis strategy designers. TabSSD separates the local extraction of variable-dependence structure from strategy design and synthesis execution: chained trees summarize variable dependence in the training data, the LLM translates the summary into executable synthesis programs, and candidate programs are executed and validated within the data holder’s environment. Across 12 benchmark datasets, this strategy-level formulation strikes a favourable balance among statistical fidelity, downstream predictive utility, empirical privacy risk, and computational efficiency. The case studies of fairness editing and clinical data augmentation further illustrate how exposing the synthesis logic as executable code enables targeted intervention and application-specific validation. Unlike extant methods that encode the generator in learned parameters or read and generate records token by token, TabSSD produces an inspectable computational procedure while avoiding the transmission of raw data to the LLM during strategy generation.

Our findings have practical and broader implications. TabSSD lowers the expertise and infrastructure barriers to tabular synthesis by automating dependence extraction, strategy design, local validation, and candidate selection. Through a standardized interface, it produces executable programs without requiring users to design deep generative models, fine-tune LLMs, perform extensive hyperparameter tuning, or undertake local GPU-accelerated neural network training. More broadly, TabSSD suggests that, for structured data, LLMs may be more valuable as designers of synthesis procedures than as direct record generators. Tree-derived summaries bridge private, order-invariant tables and general-purpose LLMs, while local statistical tools handle data access, model fitting, sampling, and evaluation. Representing the synthesis mechanism as executable code also enables failures to be traced, domain constraints to be incorporated, and strategies to be revised without retraining. This modular separation of data structure, LLM reasoning, and local verification provides a basis for human-governed generative AI.

Several limitations delineate the scope of the conclusions, although some reflect broader challenges common to current LLM-based systems rather than TabSSD alone. TabSSD depends on both the adequacy of tree-derived summaries and the reliability of the backbone LLM. Chained trees may miss weak or higher-order variable dependence, and stochastic LLM outputs may yield invalid, inefficient, or low-fidelity synthesis routines. We mitigate these risks through multi-candidate generation, local execution, and code inspection (Supplementary Table 13). Future work could further incorporate execution feedback, iterative program repair, and verifier-guided search over alternative strategies [44–48]. Scalability to ultra-high-dimensional tables can be limited by LLM context windows. Step-specific retrieval-augmented context construction [49] and graph-based decomposition [50] could restrict reasoning to the dependence relevant to each synthesis decision. Moreover, future work can further reduce information leakage by integrating formal privacy mechanisms into dependence extraction and strategy validation. Human-centred studies are also needed to assess whether practitioners can reliably understand, audit, and modify generated strategies in realistic settings. Taken together, our findings suggest that LLMs may be particularly valuable as designers of transparent, locally executable procedures rather than as unconstrained generators of sensitive structured data.

Methods

Problem setup

Let $D_{\text{train}} = \{(\mathbf{X}_i, Y_i)\}_{i=1}^N$ denote an original tabular dataset with N samples, where $\mathbf{X}_i = (X_{i1}, \dots, X_{iJ})$ and Y_i denote the feature vector and target value of the i -th sample, respectively. The dataset comprises $J+1$ variables: J feature variables $X_1, \dots, X_J$ and one target variable, or label, Y . Here, X_{ij} denotes the value of feature variable X_j for the i -th sample. The goal of tabular data synthesis is to construct a generator G that synthesizes table $\hat{D}$ from the original data D_{train} , $\hat{D} \leftarrow G(D_{\text{train}})$. The synthetic data should preserve statistical properties of the original dataset, such as marginal distributions and variable dependence, without compromising privacy. For supervised tasks, we distinguish between the features $\mathbf{X}_i$ and the target Y_i , such as a continuous target for regression tasks or a categorical target for classification tasks. If the data holder does not specify a task for the tabular data, Y_i is omitted, and the workflow of TabSSD remains unchanged.

TabSSD separates the learning of variable-dependence structure from synthesis execution. The framework first extracts variable-dependence structure from D_{train} using chained trees. It then encodes this information into a compact prompt p , which is provided to an LLM to generate K candidate synthesis strategies $\{S_k\}_{k=1}^K$. Each candidate strategy is executed locally to produce a candidate synthetic table $\hat{D}_k$. The final strategy is selected based on fidelity, empirical privacy risk, runtime criteria, and potentially other criteria. This design changes the role of the LLM from a record-level generator to a synthesis strategy designer. The LLM receives summaries of the variable-dependence structure rather than raw records, and the original data are used only within the local environment for chained tree training, strategy execution, and candidate evaluation. The workflow is summarized in Algorithm 1.

Variable dependence extraction, encoding, and prompt design

Tabular synthesis requires preserving variables' marginal distributions and dependence to maintain statistical fidelity and downstream utility. Because it is inherently difficult for an LLM to reason about variable dependence given tabular data, we use chained trees to capture variable relationships without imposing parametric assumptions. Concretely, we construct chained trees $T_{\text{chain}} = \{\text{Tree}_{X_1}, \dots, \text{Tree}_{X_J}, \text{Tree}_Y\}$ and train a feature tree Tree_{X_j} to predict each feature X_j from all other features, denoted by $\mathbf{X}_{\setminus j}$, and the label tree Tree_Y to predict Y from $\mathbf{X}$:

$$\text{Tree}_{X_j} : \mathbf{X}_{\setminus j} \rightarrow X_j, \quad \text{Tree}_Y : \mathbf{X} \rightarrow Y. \quad (1)$$

For each Tree_{X_j} , we denote by $A(X_j)$ the set of split variables. The variables in $A(X_j)$ are ordered according to the minimum tree depth where they are first split, as variables splitting closer to the root define broader partitions of the data and are more influential in predicting X_j . We denote the goodness of fit of Tree_{X_j} by $\phi(\text{Tree}_{X_j})$, which contains the coefficient of determination (R^2) and root-mean-square error (RMSE) for continuous variables and prediction accuracy for categorical variables:

$$\phi(\text{Tree}_{X_j}) = \begin{cases} (R^2(\text{Tree}_{X_j}), \text{RMSE}(\text{Tree}_{X_j})), & \text{if } X_j \text{ is continuous,} \\ \text{Acc}(\text{Tree}_{X_j}), & \text{if } X_j \text{ is categorical.} \end{cases} \quad (2)$$

Directly incorporating all decision rules into the LLM prompt would introduce substantial overhead and rapidly inflate the prompt length, particularly for high-dimensional datasets. We encode each Tree_{X_j} 's decision rule into compact dependence signals, including variable types and ranges, split variables $A(X_j)$, and goodness-of-fit scores $\phi(\text{Tree}_{X_j})$. For Tree_Y , we retain its split rules and the distribution of Y in leaf nodes. For high-dimensional datasets, even the compact dependence encoding can exceed the LLM's context-window limit. To enhance scalability, we apply a greedy set-cover procedure. At each iteration, we select the Tree_{X_j} whose variable set $A(X_j) \cup \{X_j\}$ covers the largest number of previously uncovered features, and repeat this process until all features are covered. We always retain Tree_Y when Y is defined.

Given the variable-dependence structure extracted by the chained trees, we construct a structured prompt that guides the LLM to reason about variable relationships and generate an executable synthesis strategy. The prompt p is composed of five components:

$$p = p_{\text{task}} \oplus p_{\text{metric}} \oplus p_{\text{requirement}} \oplus p_{\text{dep}} \oplus p_{\text{output}}, \quad (3)$$

where $\oplus$ denotes textual concatenation. p_{task} is the task definition that informs the LLM that the goal is to generate an executable tabular data synthesis strategy rather than directly populating data records. p_{metric} is the guidance on evaluation metrics. It tells the LLM that the generated synthetic data will be evaluated

according to statistical fidelity, downstream machine-learning utility, and empirical privacy-related risk. This component does not directly optimize these metrics but encourages the LLM to reason about trade-offs among the metrics when designing the synthesis strategy. $p_{\text{requirement}}$ instructs the LLM to analyse p_{dep} to infer variable dependence and distributional characteristics. It also encourages dividing heterogeneous data into more homogeneous subspaces when appropriate, thereby simplifying the modelling of each subspace. p_{dep} contains information about the variable-dependence structure extracted by the chained trees. p_{output} provides both an output template and a worked-in-context example. The template requires each candidate synthesis strategy to be expressed as an executable Python program with the following standardized interface:

```
generate_synthesis_data(save_path, n_sample, X_train, y_train).
```

During local execution, `X_train` and `y_train` provide the training data, `n_sample` specifies the number of records, and `save_path` specifies the output location. The function saves `n_sample` records matching the training schema. An example prompt is provided in Supplementary Note D.1.

Synthesis strategy validation and selection

Given the prompt p , we query the LLM multiple times to generate a set of candidate synthesis strategies. Because LLM outputs are stochastic and the extracted dependence information can support multiple valid synthesis approaches, these candidates often vary in their variable-generation order, subspace partitioning, and choice of statistical or machine learning models. Consequently, the candidates exhibit disparate statistical fidelity, empirical utility, privacy risk, and execution time. Generating multiple candidates increases strategy diversity, allowing subsequent validation and selection to filter out suboptimal or non-executable pipelines. The impact of the number of LLM queries on overall synthesis performance is detailed in Supplementary Fig. 2.

Each candidate strategy is executed locally to generate a synthetic table; any strategy that fails to execute is discarded. We then select a synthesis strategy through a filtering step prioritizing privacy and computational efficiency, followed by an evaluation step based on statistical fidelity. During filtering, we discard strategies whose execution time exceeds a predefined limit or whose resulting synthetic table exhibits a δ -presence privacy risk above an acceptable threshold. The remaining strategies constitute the feasible candidate set Ω_{feas} . We then compute the marginal distribution error and pairwise correlation error for the remaining synthetic tables, ultimately selecting the strategy that minimizes the combined errors:

$$S^* = \arg \min_{S \in \Omega_{\text{feas}}} E(\hat{D}(S), D_{\text{train}}), \quad (4)$$

where E is the average of the marginal distribution error and pairwise correlation error. Finally, S^* or its synthetic table can be published or used for downstream tasks.

Algorithm 1 LLM-guided strategy-level tabular data synthesis

Notation: $D_{\text{train}} = \{(\mathbf{X}_i, Y_i)\}_{i=1}^N$, where $\mathbf{X}_i = (X_{i1}, \dots, X_{iJ})$. Tree_{X_j} predicts X_j from $\mathbf{X}_{\setminus j}$, whereas Tree_Y predicts Y from $\mathbf{X}$.
 $A(X_j)$ denotes the split-variable set of Tree_{X_j} ; $\phi(\text{Tree}_{X_j})$ denotes its goodness-of-fit statistics;
 $\text{Exec}(S; D, n)$ executes strategy S on D to generate n samples; and $E(\hat{D}, D)$ denotes the statistical error.
If no target is specified, Y and Tree_Y are omitted.

Require: Training data D_{train} ; number of synthetic samples n_{sample} ; number of LLM queries K ; δ -presence risk threshold $\tau_\delta = 10$; execution-time threshold $\tau_T = 100$ s

- 1: Define the synthesis task, evaluation objectives, and output requirements
- 2: Construct $\mathcal{T}_{\text{chain}} \leftarrow \{\text{Tree}_{X_1}, \dots, \text{Tree}_{X_J}, \text{Tree}_Y\}$ from D_{train}
- 3: Extract $\{A(X_j), \phi(\text{Tree}_{X_j})\}_{j=1}^J$ and, when present, the split structure and leaf-level target distributions of Tree_Y
- 4: $\mathcal{T}_{\text{enc}} \leftarrow \mathcal{T}_{\text{chain}}$
- 5: **if** D_{train} is high-dimensional **then**
- 6: $\mathcal{T}_{\text{sel}} \leftarrow \emptyset, \quad \mathcal{U} \leftarrow \emptyset$
- 7: **while** $\mathcal{U} \neq \{X_1, \dots, X_J\}$ **do**
- 8: Select the tree covering the most previously uncovered variables:
$$j^* \leftarrow \arg \max_{j: \text{Tree}_{X_j} \notin \mathcal{T}_{\text{sel}}} |(A(X_j) \cup \{X_j\}) \setminus \mathcal{U}|$$
- 9: $\mathcal{T}_{\text{sel}} \leftarrow \mathcal{T}_{\text{sel}} \cup \{\text{Tree}_{X_{j^*}}\}$
- 10: $\mathcal{U} \leftarrow \mathcal{U} \cup A(X_{j^*}) \cup \{X_{j^*}\}$
- 11: **end while**
- 12: $\mathcal{T}_{\text{enc}} \leftarrow \mathcal{T}_{\text{sel}} \cup \{\text{Tree}_Y\}$
- 13: **end if**
- 14: Encode the information retained in $\mathcal{T}_{\text{enc}}$ as p_{dep}
- 15: Construct the structured prompt
$$p \leftarrow p_{\text{task}} \oplus p_{\text{metric}} \oplus p_{\text{requirement}} \oplus p_{\text{dep}} \oplus p_{\text{output}}$$
- 16: initialize the feasible strategy set $\Omega_{\text{feas}} \leftarrow \emptyset$
- 17: **repeat**
- 18: **for** $k = 1, \dots, K$ **do**
- 19: Query the LLM with p to generate a candidate strategy S_k
- 20: Attempt $\hat{D}(S_k) \leftarrow \text{Exec}(S_k; D_{\text{train}}, n_{\text{sample}})$; skip non-executable or invalid candidates
- 21: Evaluate the statistical-fidelity error $E(\hat{D}(S_k), D_{\text{train}})$, δ -presence risk $R_\delta(\hat{D}(S_k), D_{\text{train}})$, and execution time $T(S_k)$
- 22: **if** $R_\delta(\hat{D}(S_k), D_{\text{train}}) \leq \tau_\delta$ **and** $T(S_k) \leq \tau_T$ **then**
- 23: $\Omega_{\text{feas}} \leftarrow \Omega_{\text{feas}} \cup \{S_k\}$
- 24: **end if**
- 25: **end for**
- 26: **until** $\Omega_{\text{feas}} \neq \emptyset$ Raw records are not transmitted
- 27: Select
$$S^* \leftarrow \arg \min_{S \in \Omega_{\text{feas}}} E(\hat{D}(S), D_{\text{train}})$$
- 28:
$$\hat{D}^* \leftarrow \hat{D}(S^*)$$
- 29: **return** S^* and $\hat{D}^*$

Data availability

All datasets analysed in this study are publicly available from the UCI Machine Learning Repository and OpenML, with links to the original data sources provided in Supplementary Table 1. The processed datasets, train-test splits, feature-type annotations, and random seeds used to reproduce the experiments are available at https://anonymous.4open.science/r/llm_designer-126C.

445 **Code availability**

446 The source code used to reproduce all experiments, including dependence extraction, prompt construction,
447 validation of LLM-generated synthesis strategies, baseline execution, and metric evaluation, is available at
448 https://anonymous.4open.science/r/llm_designer-126C.

449 **Author contributions**

450 J.L., H.Y., and D.G. conceived the study. J.L., H.Y., D.G., and Q.Z. contributed substantially to the design
451 of various aspects of the study. J.L. developed the code and performed the experiments. J.L., H.Y., D.G.,
452 and Q.Z. analysed and interpreted the experimental results. All authors contributed to the critical revision
453 of the manuscript and reviewed and approved the final version.

454 **Competing interests**

455 The authors declare no competing interests.

References

- [1] D'Sa, K., Evans, J.R., Viridi, G.S., Vecchi, G., Adam, A., Bertolli, O., Fleming, J., Chang, H., Leighton, C., Horrocks, M.H., Athauda, D., Choi, M.L., Gandhi, S.: Prediction of mechanistic subtypes of Parkinson's using patient-derived stem cell models. *Nature Machine Intelligence* **5**(8), 933–946 (2023) <https://doi.org/10.1038/s42256-023-00702-9>
- [2] Gamella, J.L., Peters, J., Bühlmann, P.: Causal chambers as a real-world physical testbed for AI methodology. *Nature Machine Intelligence* **7**(1), 107–118 (2025) <https://doi.org/10.1038/s42256-024-00964-x>
- [3] European Parliament, Council: General data protection regulation. *Off. J. Eur. Union* **L 119**, 1–88 (2016)
- [4] Yan, C., Yan, Y., Wan, Z., Zhang, Z., Omberg, L., Guinney, J., Mooney, S.D., Malin, B.A.: A multifaceted benchmarking of synthetic electronic health record generation models. *Nature Communications* **13**(1), 7609 (2022) <https://doi.org/10.1038/s41467-022-35295-1>
- [5] van Breugel, B., Liu, T., Oglic, D., van der Schaar, M.: Synthetic data in biomedicine via generative artificial intelligence. *Nature Reviews Bioengineering* **2**(12), 991–1004 (2024) <https://doi.org/10.1038/s44222-024-00245-7>
- [6] Reynolds, D.: Gaussian mixture models. In: Li, S.Z., Jain, A. (eds.) *Encyclopedia of Biometrics*, pp. 659–663. Springer, Boston, MA (2009). https://doi.org/10.1007/978-0-387-73003-5_196
- [7] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**, 321–357 (2002)
- [8] Nowok, B., Raab, G.M., Dibben, C.: synthpop: Bespoke creation of synthetic data in R. *Journal of Statistical Software* **74**, 1–26 (2016)
- [9] O'Hagan, A., Murphy, T.B., Gormley, I.C., McNicholas, P.D., Karlis, D.: Clustering with the multivariate normal inverse gaussian distribution. *Computational Statistics & Data Analysis* **93**, 18–30 (2016) <https://doi.org/10.1016/j.csda.2014.09.006>
- [10] Blagus, R., Lusa, L.: SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics* **14**(1), 106 (2013) <https://doi.org/10.1186/1471-2105-14-106>
- [11] Schmidhuber, J.: Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90, Institut für Informatik, Technische Universität München (1990)
- [12] Schmidhuber, J.: A possibility for implementing curiosity and boredom in model-building neural controllers. In: *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pp. 222–227 (1991)
- [13] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, vol. 27, pp. 2672–2680 (2014)
- [14] Jarzynski, C.: Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach. *Phys. Rev. E* **56**, 5018–5035 (1997) <https://doi.org/10.1103/PhysRevE.56.5018>
- [15] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: *Proceedings of the 32nd International Conference on Machine Learning. Proceedings of Machine Learning Research*, vol. 37, pp. 2256–2265 (2015)
- [16] Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851 (2020)
- [17] Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* **1**(5), 206–215 (2019)
- [18] Fang, Z., Jiang, Z., Chen, H., Li, X., Li, J.: Understanding and mitigating memorization in

diffusion models for tabular data. In: Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 15976–16005 (2025). <https://proceedings.mlr.press/v267/fang25f.html>

- [19] Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., Sontag, D.: TabLLM: Few-shot classification of tabular data with large language models. In: International Conference on Artificial Intelligence and Statistics, pp. 5549–5581 (2023). PMLR
- [20] Ye, H., Li, J., Zhao, H., Guo, D., Chang, Y.: LLM meeting decision trees on tabular data. In: Advances in Neural Information Processing Systems, vol. 38, pp. 118095–118131 (2025). <https://doi.org/10.52202/085713-3938>
- [21] Wu, X., Ritter, A., Xu, W.: Tabular data understanding with LLM: A survey of recent advances and challenges. arXiv preprint arXiv:2508.00217 (2025)
- [22] Fang, X., Xu, W., Tan, F.A., Hu, Z., Zhang, J., Qi, Y., Sengamedu, S.H., Faloutsos, C.: Large language models (LLMs) on tabular data: Prediction, generation, and understanding - a survey. Transactions on Machine Learning Research (2024)
- [23] Borisov, V., Sessler, K., Leemann, T., Pawelczyk, M., Kasneci, G.: Language models are realistic tabular data generators. In: The Eleventh International Conference on Learning Representations (2023)
- [24] Yang, S., Yuan, C., Rong, Y., Steinbauer, F., Kasneci, G.: P-ta: Using proximal policy optimization to enhance tabular data augmentation via large language models. In: Findings of the Association for Computational Linguistics: ACL 2024, pp. 248–264 (2024)
- [25] Seedat, N., Huynh, N., Breugel, B., Schaar, M.: Curated LLM: Synergy of LLMs and data curation for tabular augmentation in low-data regimes. In: Proceedings of the 41st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 235, pp. 44060–44092 (2024). <https://proceedings.mlr.press/v235/seedat24a.html>
- [26] Kim, J., Kim, T., Choo, J.: Epic: Effective prompting for imbalanced-class data synthesis in tabular data classification via large language models. In: Advances in Neural Information Processing Systems, vol. 37, pp. 31504–31542 (2024)
- [27] Ward, J., Gu, B., Wang, C.-H., Cheng, G.: When tables leak: Attacking string memorization in llm-based tabular data generation. arXiv preprint arXiv:2512.08875 (2025)
- [28] Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, U., *et al.*: Extracting training data from large language models. In: 30th USENIX Security Symposium (USENIX Security 21), pp. 2633–2650 (2021)
- [29] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth International Group, Belmont, CA (1984)
- [30] Grinsztajn, L., Oyallon, E., Varoquaux, G.: Why do tree-based models still outperform deep learning on typical tabular data? In: Advances in Neural Information Processing Systems, vol. 35, pp. 507–520 (2022)
- [31] Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling tabular data using Conditional GAN. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
- [32] Kotelnikov, A., Baranchuk, D., Rubachev, I., Babenko, A.: TabDDPM: Modelling tabular data with diffusion models. In: International Conference on Machine Learning, pp. 17564–17579 (2023). PMLR
- [33] Shi, J., Xu, M., Hua, H., Zhang, H., Ermon, S., Leskovec, J.: TabDiff: a mixed-type diffusion model for tabular data generation. In: The Thirteenth International Conference on Learning Representations (2025). <https://openreview.net/forum?id=swvURjrt8z>
- [34] Zhang, H., Zhang, J., Srinivasan, B., Shen, Z., Qin, X., Faloutsos, C., Rangwala, H., Karypis, G.: Mixed-type tabular data synthesis with score-based diffusion in latent space. In: The Twelfth International Conference on Learning Representations (2024)

- [35] Watson, D.S., Blesch, K., Kapar, J., Wright, M.N.: Adversarial random forests for density estimation and generative modeling. In: International Conference on Artificial Intelligence and Statistics, pp. 5357–5375 (2023). PMLR
- [36] Bravo, J.: NRGBoost: Energy-based generative boosted trees. In: The Thirteenth International Conference on Learning Representations (2025). <https://openreview.net/forum?id=wQHylZ1SH>
- [37] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**(1), 119–139 (1997)
- [38] Breiman, L.: Random forests. *Machine Learning* **45**(1), 5–32 (2001)
- [39] Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794 (2016)
- [40] Ganey, G., Nazari, M., Davison, R., Dizche, A.F., Wu, X., Abbey, R., Silva, J.G., Cristofaro, E.D.: SMOTE and mirrors: Exposing privacy leakage from synthetic minority oversampling. In: The Fourteenth International Conference on Learning Representations (2026). <https://openreview.net/forum?id=ZQSZMpsQKj>
- [41] Goldsmith, S.F., Aiken, A.S., Wilkerson, D.S.: Measuring empirical computational complexity. In: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, pp. 395–404 (2007)
- [42] Becker, B., Kohavi, R.: Adult. UCI Machine Learning Repository (1996). <https://doi.org/10.24432/C5XW20>
- [43] Lubicz, M., Pawelczyk, K., Rzechonek, A., Kolodziej, J.: Thoracic Surgery Data. UCI Machine Learning Repository (2014). <https://doi.org/10.24432/C5Z60N>
- [44] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhunoye, S., Yang, Y., Gupta, S., Majumder, B.P., Hermann, K., Welleck, S., Yazdanbakhsh, A., Clark, P.: Self-refine: Iterative refinement with self-feedback. In: Advances in Neural Information Processing Systems, vol. 36, pp. 46534–46594 (2023)
- [45] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. In: Advances in Neural Information Processing Systems, vol. 36, pp. 8634–8652 (2023)
- [46] Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. In: Advances in Neural Information Processing Systems, vol. 36, pp. 11809–11822 (2023)
- [47] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: ReAct: Synergizing reasoning and acting in language models. In: The Eleventh International Conference on Learning Representations (2023)
- [48] Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., Cobbe, K.: Let’s verify step by step. In: The Twelfth International Conference on Learning Representations (2024). <https://openreview.net/forum?id=v8L0pN6EOi>
- [49] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474 (2020)
- [50] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018). <https://openreview.net/forum?id=rJXmpikCZ>
- [51] Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025)

- 593 [52] Patki, N., Wedge, R., Veeramachaneni, K.: The synthetic data vault. In: IEEE International Conference
594 on Data Science and Advanced Analytics (DSAA), pp. 399–410 (2016). [https://doi.org/10.1109/DSAA.](https://doi.org/10.1109/DSAA.2016.49)
595 [2016.49](https://doi.org/10.1109/DSAA.2016.49)
- 596 [53] Qian, Z., Davis, R., van der Schaar, M.: Synthcity: A benchmark framework for diverse use cases of
597 tabular synthetic data. In: Advances in Neural Information Processing Systems, vol. 36, pp. 3173–3188
598 (2023). <https://doi.org/10.52202/075280-0140>
- 599 [54] DataCebo, Inc.: Synthetic Data Metrics. (2023). Version 0.9.3. <https://docs.sdv.dev/sdmetrics/>
- 600 [55] Zeng, A., Lv, X., Hou, Z., Du, Z., Zheng, Q., Chen, B., Yin, D., Ge, C., Huang, C., Xie, C., et al.:
601 Glm-5: from vibe coding to agentic engineering. arXiv preprint arXiv:2602.15763 (2026)
- 602 [56] Qwen Team: Qwen3-Max: Just Scale it (2025)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [Supplementary.pdf](#)