When Synthetic Data Is Enough: Calibration for Tabular Model Ranking

Gennadii Filatov
AI Institute, ITMO University
filatovgwork@gmail.com

Irina Deeva
AI Institute, ITMO University
iriny.deeva@gmail.com

Abstract

Model selection in tabular machine learning typically relies on held-out validation or cross-validation, which reduces effective training data and increases computational cost. Recent work suggests that synthetic data can serve as a validation surrogate if they preserve the *ranking* of candidate models, but generator imperfections in tabular domains often make uncalibrated synthetic evaluation unreliable. We propose a simple, task-agnostic calibration procedure for synthetic tabular validation: given a synthetic validation set and a small calibration subset of models, we learn sample weights by aligning weighted synthetic losses with cross-fitted real-data loss estimates, yielding a calibrated synthetic risk for ranking and selection. The method generalizes correctness-based schemes to arbitrary losses and supports both classification and regression. Across five classification and five regression benchmarks and multiple generators (CTGAN, TVAE, Gaussian Copula, TabDDPM, TabPFN-based generation), calibration consistently improves ranking fidelity measured by Spearman correlation of synthetic and real model orderings, with large gains when generators are weaker. Finally, we study weight interpretability via a surrogate regressor to predict weights from sample values and analyzing SHAP attributions, revealing that weight assignment is driven primarily by the synthetic target and a few task-relevant covariates, providing insight into when and why calibration succeeds.

Workshop "Towards Trustworthy Predictions: Theory and Applications of Calibration for Modern AI" at AISTATS 2026, Tangier, Morocco.

1 Introduction

Model selection is a central and costly step in tabular machine learning. Standard practice relies on held-out validation sets or cross-validation, which reduce the effective training sample size, multiply computational cost, and create a mismatch between the model instance selected during tuning and the model ultimately trained on the full dataset. These issues are particularly acute in tabular settings, where data are often limited, imbalanced, or heterogeneous, and where model performance can vary substantially across seeds and hyperparameters. Synthetic data generation offers an appealing alternative: if a synthetic dataset can reliably preserve the ranking of candidate models, it can serve as a validation surrogate without sacrificing real training data. However, in tabular domains, generator imperfection - missing interactions, poor tail behavior, or misrepresented subpopulations—often lead to biased and unstable synthetic evaluation.

In this work, we propose a calibration algorithm for synthetic tabular validation that aligns synthetic evaluation with real-data performance. Rather than scoring models by an unweighted average synthetic loss, we learn importance weights over synthetic samples such that, for a small set of calibration models, the weighted synthetic loss matches their real loss estimates computed via cross-fitting on the training data. Our formulation is loss-based and task-agnostic, supporting both classification and regression. This generalizes correctness-based calibration schemes and leverages the full magnitude of model disagreement, which is particularly informative in heterogeneous tabular settings. We empirically demonstrate that calibrated synthetic validation substantially improves model ranking fidelity and reduces selection regret across multiple tabular datasets and generator qualities. Moreover, we introduce an explainability layer that uses SHAP-based analysis of learned sample weights to reveal which synthetic regions drive model discrimination, providing insight into when and

why calibration succeeds. Together, our results show that synthetic data, when properly calibrated, can become a reliable and data-efficient tool for model selection in tabular learning. The code and data are available in the repository `https://github.com/ITMO-NSS-team/tabular-synth-calibration`.

2 Related Work

2.1 Synthetic data for validation

A recurring challenge in using synthetic data inside an ML pipeline is domain gap: even when a generator produces samples that look plausible (or match low-order statistics), the behavior of downstream models trained or evaluated on the synthetic distribution can deviate substantially from behavior on real data. [Shoshan et al., 2023] explicitly frame this distinction: for model selection, the key requirement is rank preservation across domains (synthetic evaluation should induce the same ordering as real evaluation), which is weaker-and different-than minimizing the usual synthetic-to-real generalization gap. They propose substituting a held-out real validation set with a synthetic one and then improving ranking via sample reweighting calibration of the synthetic set. [Hu et al., 2023] argue that synthetic samples can diversify the validation set and reduce overfitting in domains where real validation data are limited. In a different but related direction [Boyeau et al., 2024] propose statistically principled correction procedures (prediction-powered inference) that combine a small human-labeled set with larger synthetic-labeled sets to improve sample efficiency without introducing bias. Prior work in this area spans both domain-specific and general-purpose approaches. For example, [Shoshan et al., 2023] focus on visual domains, whereas [Boyeau et al., 2024] propose a task-agnostic evaluation framework applicable across a wide range of settings.

2.2 Calibrating synthetic tabular data

Research on calibrating and curating synthetic tabular data for predictive training and validation has developed along several complementary directions. One line of work modifies the generator itself to improve downstream utility, for example, by incorporating task-aware losses that preserve feature-label relationships or by guiding generation with auxiliary predictive models [Zhao et al., 2024, Jia et al., 2024]. A second line focuses on post-hoc refinement [Ghalebikesabi et al., 2022], where synthetic datasets are reweighted or resampled-often via density-ratio estimation using classifiers-to better align with real data and improve predictive performance. Constrained optimization approaches [Wang et al., 2023] further formalize calibration as projecting the synthetic distribution onto one that matches selected utility statistics, while iterative refinement schemes progressively improve sample quality using evaluation signals. Although these methods demonstrate that synthetic tabular data can be made more useful for predictive tasks, they typically target distributional fidelity or training utility rather than directly optimizing for reliable model ranking and validation, leaving a gap that motivates task-specific calibration for model selection.

3 Loss-Calibrated Synthetic Validation

Let $D_r = \{(x_j, y_j)\}_{j=1}^{N_r}$ denote the real training data and $D_s = \{(x_i^s, y_i^s)\}_{i=1}^{N_s}$ a labeled synthetic validation set generated from a model trained on D_r. Our objective is to rank a candidate model set $\mathcal{H}$ using D_s so that the induced ordering matches the performance under the real distribution. We introduce a small calibration subset $\mathcal{H}_{\text{cal}} = \{h_m\}_{m=1}^{M} \subset \mathcal{H}$ and fix a task-appropriate loss $\ell(\cdot,\cdot)$ (e.g. log loss for classification, absolute error for regression). Define the synthetic loss matrix

$$L_{i,m} = \ell(h_m(x_i^s),\, y_i^s)\,,$$

and let $\hat{\ell}_r(m)$ be the real-data loss estimate for h_m, obtained via cross-fitted evaluation on D_r. Calibration learns weights $w \in \mathbb{R}^{N_s}$ by solving

$$w^\star = \arg\min_{w} \left\|\hat{\ell}_r - L^\top w\right\|_2^2 + \lambda\|w\|_2^2 \quad \text{s.t. } w \succeq 0,\ \mathbf{1}^\top w = 1,$$

where $\lambda > 0$ controls regularization. The calibrated synthetic risk of any $h \in \mathcal{H}$ is then

$$\hat{\ell}_{s,\text{cal}}(h) = \sum_{i=1}^{N_s} w_i^\star\, \ell(h(x_i^s),\, y_i^s)\,.$$

Models are selected by minimizing $\hat{\ell}_{s,\text{cal}}(h)$.

Intuitively, the optimization redistributes probability mass over synthetic samples so that the weighted synthetic losses reproduce the real-data losses of calibration models, thereby emphasizing regions where models meaningfully disagree. The simplex constraints ensure that $\hat{\ell}_{s,\text{cal}}$ remains an expectation under a valid reweighted synthetic distribution, improving numerical stability and interpretability, while the ℓ_2 penalty prevents overfitting to $\mathcal{H}_{\text{cal}}$. When tabular heterogeneity is substantial, we optionally perform segment-wise calibration (e.g., per class or per target-quantile bin) and aggregate the resulting weighted risks, which improves robustness under imbalance and heavy-tailed

targets. Computationally, calibration requires only evaluating M models on N_s synthetic samples and solving a single constrained quadratic program, which is negligible relative to training the candidate model pool.

Computationally, we solve the resulting bound-constrained quadratic program using L-BFGS-B quasi-Newton method as implemented in SciPy's `minimize` method, initialized from uniform weights and run with analytic gradients for up to 15000 iterations with tight stopping tolerances ($f_{tol} = 10^{-10}$, $g_{tol} = 10^{-8}$), which in practice yield numerically stable and accurate convergence of the calibration weights.

4 Experimental Setup

We evaluate calibration for synthetic tabular validation on a suite of 10 tabular benchmarks: 5 for classification and 5 for regression. For downstream evaluation we consider a diverse pool of predictive models (50 for classification and 42 for regression), spanning linear baselines, tree-based methods, and neural tabular architectures. As synthetic-data sources we use five generators/priors representing different modeling paradigms: CTGAN, TabDDPM, Gaussian Copula, TVAE and a TabPFN-based baseline. For each dataset we apply a standard preprocessing pipeline: numerical features (and the target, when applicable) are scaled using `MinMaxScaler`, and categorical features are encoded. All generative models were subjected to hyperparameter tuning (details in the Appendix B). All datasets are publicly available benchmarks: datasets accessed via the UCI Machine Learning Repository [Dua and Graff, 2019] (mushroom, diabetes, abalone, wine quality, obesity) are distributed under the Creative Commons Attribution 4.0 International (CC BY 4.0) licence; datasets loaded via scikit-learn [Pedregosa et al., 2011] (California housing, diabetes regression) are in the public domain; and datasets retrieved through OpenML [Bischl et al., 2025] (German credit, heart disease, concrete strength) are likewise available under CC BY 4.0. All experiments were conducted on an internal university cluster equipped with an Intel Xeon CPU at 3.19 GHz and 24 GB of RAM.

4.1 Experiment 1: Calibration Improves Ranking Fidelity

Research question. Does loss-based calibration improve the ability of synthetic validation to preserve the ranking of predictive models with respect to real test performance?

Protocol. We perform K-fold cross-validation on the real dataset. For each fold, the training portion D_{train} is further divided into $(D_{\text{train}}^{\text{calib}}, D_{\text{test}}^{\text{calib}})$.

We partition the model pool $\mathcal{H}$ into calibration models $\mathcal{H}_{\text{cal}}$ and evaluation models $\mathcal{H}_{\text{eval}}$. On each fold, we randomly select $|\mathcal{H}_{\text{cal}}| = 15$ models for calibration, and all models are trained on $D_{\text{train}}^{\text{calib}}$. The tuned generator $G_{\theta_G^\star}$ is trained on the same data and used to sample

$$D_s^{\text{test}} \sim G_{\theta_G^\star}, \qquad |D_s^{\text{test}}| = |D_{\text{test}}|.$$

Calibration targets are computed as

$$\hat{\ell}_r(m) = \hat{\ell}_{D_{\text{test}}^{\text{calib}}}(h_m), \quad h_m \in \mathcal{H}_{\text{cal}},$$

and the synthetic loss matrix is defined as

$$L_{i,m} = \ell(h_m(x_i^s), y_i^s), \quad (x_i^s, y_i^s) \in D_s^{\text{test}}.$$

Weights $w^\star$ are obtained by solving the calibration problem described in Section 3. For each $h \in \mathcal{H}_{\text{eval}}$, we compute

$$\hat{\ell}_{s,\text{cal}}(h) = \sum_i w_i^\star \ell(h(x_i^s), y_i^s),$$

and compare rankings induced by calibrated synthetic loss, uncalibrated synthetic loss, and real test loss $\hat{\ell}_{D_{\text{test}}}(h)$ using Spearman correlation, averaged across folds.

4.2 Experiment 2: Explainability of Calibration Weights

Research question. Are the learned synthetic sample weights $w^\star$ interpretable in terms of meaningful tabular feature patterns?

Protocol. Using the calibrated weights obtained in Experiment 1, we construct a dataset

$$\mathcal{D}_w = \{(\phi(x_i^s, y_i^s),\, w_i^\star)\}_{i=1}^{N_s},$$

where $\phi(\cdot)$ denotes the encoded synthetic feature representation. A surrogate regressor g (e.g., XGBoost) is trained to predict $w_i^\star$ from $\phi(x_i^s, y_i^s)$. We then compute SHAP values for g to obtain (i) global feature importance for weight assignment and (ii) local explanations for high- and low-weight samples.

5 Experimental Results

The figure 1 presents the outcomes of the first experiment, in which a comparison was made between the ranking of models on synthetic and real data using Spearman's rank correlation. Detailed results for each

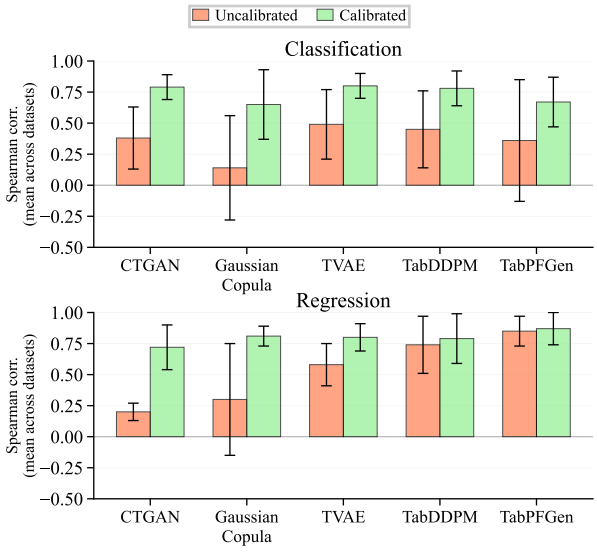

Figure 1: Comparison of Spearman's rank correlation for ranking models on synthetic data without calibration and with calibration (the higher the value, the closer the order of models is to the order obtained on real test data)

dataset are presented in table 1 in the Appendix A. Across the five classification and five regression benchmarks, calibration consistently improves the average agreement between synthetic and real model rankings for each generative model. On classification datasets, uncalibrated synthetic evaluation yields only moderate correlations overall (mean Spearman ≈ 0.37 across generators and datasets), and calibration produces substantial and stable gains for all generators (per-generator improvements of approximately 0.29–0.51), raising correlations into a significantly higher range (approximately 0.65–0.80 on average). On regression datasets, calibration also improves ranking fidelity on average (overall mean Spearman $\approx 0.53 \rightarrow 0.79$), but we observe a clear diminishing-returns trend: weaker generators such as CTGAN and Gaussian Copula benefit most ($\Delta\rho \approx +0.52$ and $+0.51$), while stronger generators that already exhibit high uncalibrated ranking fidelity show smaller gains (TabDDPM $\Delta\rho \approx +0.05$, TabPFN $\Delta\rho \approx +0.02$). These results indicate that the proposed calibration mechanism acts as a robust correction when synthetic-real mismatch would otherwise distort model ordering, while naturally providing limited additional benefit when synthetic rankings are already close to real. The figure 2 in Appendix A provides a clear illustration of the alterations in losses subsequent to calibration. Although the algorithm does not ensure equivalent absolute values of the loss to the actual data, following calibration, the losses demonstrate a stronger correspondence to the actual data (they align on the main diagonal).

In the second experiment, we first evaluate the coefficient of determination for a surrogate model that predicts calibration weights from data points (Figure 3, Appendix A). The findings reveal an intriguing correlation: as the strength of the generative model increases, the coefficient of determination concomitantly rises. This can be explained by how the calibration behaves as the synthetic distribution becomes closer to the real one. First, when the generator is strong (e.g., TabPFN or TabDDPM in our experiments), the synthetic distribution already captures most relevant structure of the real data. In this regime the weight function $w(x, y)$ acts as a smooth, low-complexity density-ratio correction that mildly re-balances over- and underrepresented regions. In contrast, weaker generators (e.g., TVAE, CTGAN, Gaussian Copula) exhibit larger structural deficiencies: missing modes, distorted dependencies, or synthetic artifacts. Calibration in this regime must compensate for more irregular mismatches, often assigning high leverage to specific samples that correct localized failures of the generator. Using the `abalone` regression dataset as a representative example, we analyze SHAP attributions of a surrogate model (figures 4, 5 in Appendix A). While the relative importance of secondary features varies somewhat by generator (with simpler priors spreading influence across more variables), the predominance of y_{synth} is repeated across most datasets in our benchmark suite, supporting the interpretation that loss-based calibration corrects synthetic evaluation largely through reweighting along the target dimension and its associated feature structure.

6 Conclusion and Discussion

We studied whether a lightweight calibration step can make synthetic tabular data reliable for *model ranking* - a prerequisite for model selection. Our approach learns importance weights over a synthetic validation set by aligning weighted synthetic losses with cross-fitted real-data estimates on a small calibration subset of predictors. This loss-based formulation is task-agnostic and computationally inexpensive. Across five classification and five regression benchmarks with diverse generators (CTGAN, TVAE, Gaussian Copula, TabDDPM, and TabPFN-based generation), calibration consistently improves ranking fidelity, with the largest gains where uncalibrated evaluation is least reliable. An explainability analysis via SHAP attributions on a surrogate regressor shows that calibration primarily reallocates mass across target regimes (y_{synth}), with secondary dependence on a compact covariate set-supporting an interpretation as implicit, task-driven reweighting.

Limitations. First, calibration still requires *real* performance estimates for a model subset (obtained here via cross-fitting), introducing additional computation and design choices (fold count, size and composition of $\mathcal{H}_{\text{cal}}$). Second, improvements hold *on average* but are not guaranteed per dataset-generator pair; where uncalibrated ranking is already strong, gains can be marginal or slightly negative. Third, severe distributional artifacts (support mismatch, missing modes) may exceed what reweighting alone can correct, causing weight concentration. Fourth, rank correlation, while well-aligned with model selection, is only a proxy for end-to-end selection regret and may not capture cost-sensitive or top-k objectives.

References

[Bischl et al., 2025] Bischl, B., Casalicchio, G., Das, T., Feurer, M., Fischer, S., Gijsbers, P., Mukherjee, S., Müller, A. C., Németh, L., Oala, L., Purucker, L., Ravi, S., van Rijn, J. N., Singh, P., Vanschoren, J., van der Velde, J., and Wever, M. (2025). Openml: Insights from 10 years and more than a thousand papers. *Patterns*, 6(7):101317.

[Boyeau et al., 2024] Boyeau, P., Angelopoulos, A. N., Yosef, N., Malik, J., and Jordan, M. I. (2024). Autoeval done right: Using synthetic data for model evaluation. *arXiv preprint arXiv:2403.07008*.

[Dua and Graff, 2019] Dua, D. and Graff, C. (2019). UCI machine learning repository. `https://archive.ics.uci.edu/ml`.

[Ghalebikesabi et al., 2022] Ghalebikesabi, S., Wilde, H., Jewson, J., Doucet, A., Vollmer, S., and Holmes, C. (2022). Mitigating statistical bias within differentially private synthetic data. In *Uncertainty in Artificial Intelligence*, pages 696–705. PMLR.

[Hu et al., 2023] Hu, Q., Yuille, A., and Zhou, Z. (2023). Synthetic data as validation. *arXiv preprint arXiv:2310.16052*.

[Jia et al., 2024] Jia, F., Zhu, H., Jia, F., Ren, X., Chen, S., Tan, H., and Chan, W. K. V. (2024). A tabular data generation framework guided by downstream tasks optimization. *Scientific Reports*, 14(1):15267.

[Kotelnikov et al., 2023] Kotelnikov, A., Baranchuk, D., Rubachev, I., and Babenko, A. (2023). Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pages 17564–17579. PMLR.

[Ma et al., 2024] Ma, J., Dankar, A., Stein, G., Yu, G., and Caterini, A. (2024). Tabpfgen–tabular data generation with tabpfn. *arXiv preprint arXiv:2406.05216*.

[Patki et al., 2016] Patki, N., Wedge, R., and Veeramachaneni, K. (2016). The synthetic data vault. In *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 399–410.

[Pedregosa et al., 2011] Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Qian et al., 2023] Qian, Z., Davis, R., and van der Schaar, M. (2023). Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. *Advances in Neural Information Processing Systems*, 36:3173–3188.

[Shoshan et al., 2023] Shoshan, A., Bhonker, N., Kviatkovsky, I., Fintz, M., and Medioni, G. (2023). Synthetic data for model selection. In *International Conference on Machine Learning*, pages 31633–31656. PMLR.

[Wang et al., 2023] Wang, H., Sudalairaj, S., Henning, J., Greenewald, K., and Srivastava, A. (2023). Post-processing private synthetic data for improving utility on selected measures. *Advances in Neural Information Processing Systems*, 36:64139–64154.

[Xu et al., 2019] Xu, L., Skoularidou, M., Cuesta-Infante, A., and Veeramachaneni, K. (2019). Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, volume 32.

[Zhao et al., 2024] Zhao, Z., Kunar, A., Birke, R., Van der Scheer, H., and Chen, L. Y. (2024). Ctab-gan+: Enhancing tabular data synthesis. *Frontiers in big Data*, 6:1296508.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **Yes.** The mathematical formulation of the calibration objective, loss matrix construction, constraints, and selection rule are presented, including notation and optimization problem.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **Yes.** Section 3 discusses computational complexity (model evaluations and quadratic program), and Section 6 outlines practical computational considerations.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries.
Yes. https://github.com/ITMO-NSS-team/tabular-synth-calibration.

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results.
Not Applicable. The paper focuses on empirical evaluation and algorithmic development rather than formal theoretical guarantees.

(b) Complete proofs of all theoretical results.
Not Applicable.

(c) Clear explanations of any assumptions.
Yes. Assumptions regarding cross-fitted loss estimation, calibration subset selection, and loss definitions are described in Sections 3 and 4.

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL).
Yes. https://github.com/ITMO-NSS-team/tabular-synth-calibration.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen).
Yes. Section 4 describes splits and evaluation protocol; hyperparameter search spaces are detailed in Appendix B.

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times).
Yes. Spearman correlation is explicitly defined as the primary metric; results are reported as mean±std across folds and datasets.

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider).
Yes. See Section 4.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator if your work uses existing assets.
Yes. All generative models and predictive baselines (CTGAN, TVAE, TabDDPM, TabPFN, Gaussian Copula) are cited appropriately.

(b) The license information of the assets, if applicable.
Yes. See Section 4.

(c) New assets either in the supplemental material or as a URL, if applicable.
Not Applicable. No new datasets are released.

(d) Information about consent from data providers/curators.
Not Applicable. All datasets are publicly available benchmarks.

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content.
Not Applicable. The datasets used are standard tabular benchmarks without personally identifiable or sensitive content.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots.
Not Applicable.

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable.
Not Applicable.

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation.
Not Applicable.

A Detailed results of experiments

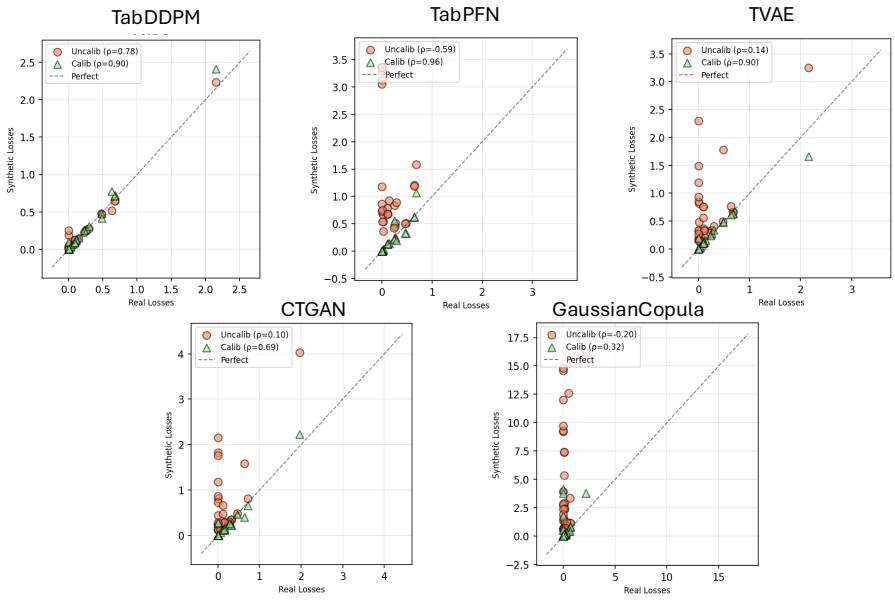

Figure 2: Comparison of model losses ($\mathcal{H}_{\text{eval}}$) on real test data and synthetic data for the *mushroom* dataset

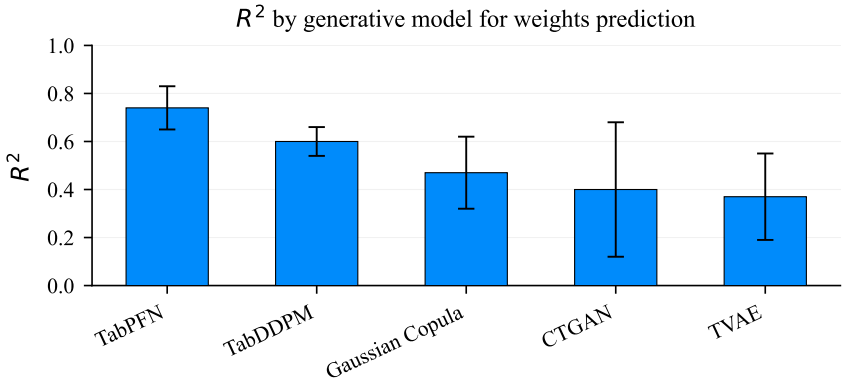

Figure 3: The coefficient of determination for a surrogate model that predicts calibration weights based on data points.

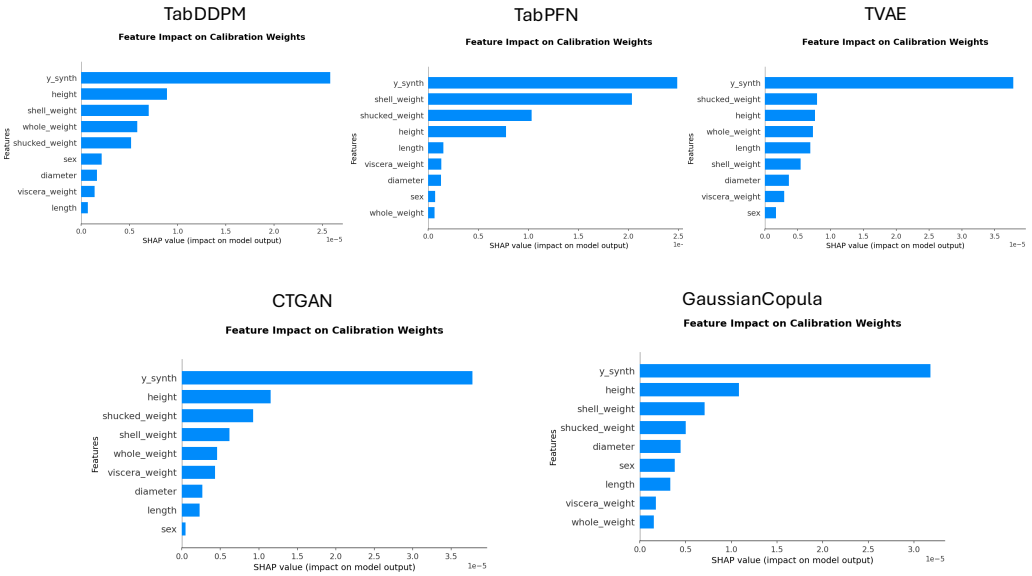

Figure 4: Estimation of the influence of features in the *abalone* dataset on the obtained calibration weight values based on Shapley value analysis.

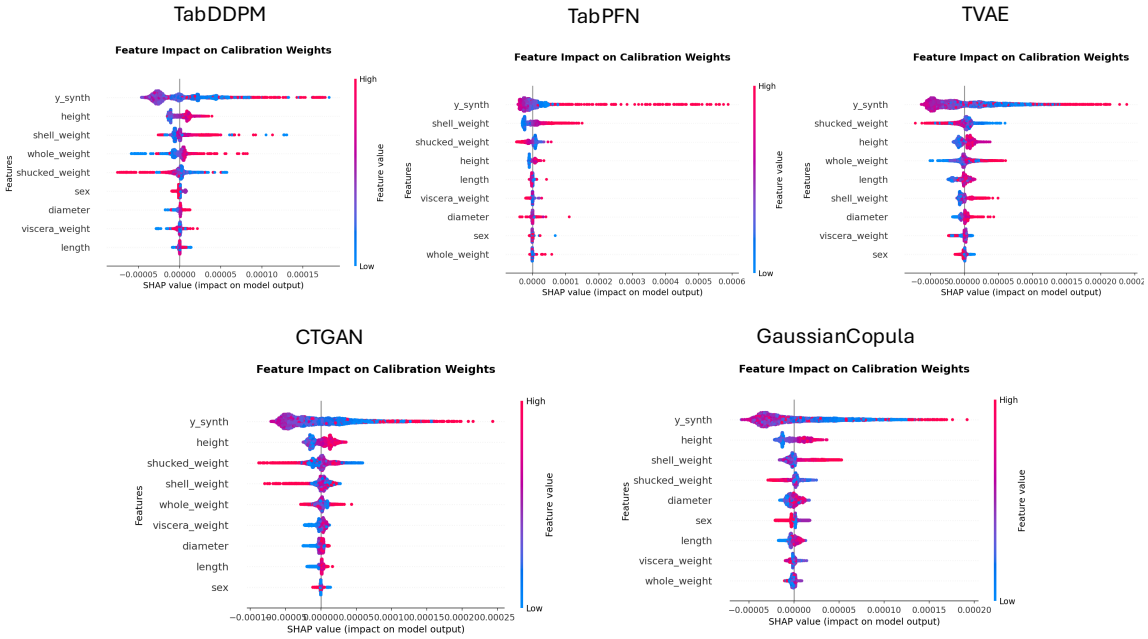

Figure 5: Visualization of the results of analyzing the influence of feature values in the *abalone* dataset on calibration weight values using tornado plots with Shapley values.

Table 1: Spearman correlation (mean±std) before and after calibration for each dataset and generative model.

Task	Dataset	Generator	Uncalibrated	Calibrated
Classification	Mushroom	CTGAN	0.129±0.031	**0.772±0.129**
Classification	Mushroom	Gaussian Copula	-0.219±0.096	**0.173±0.179**
Classification	Mushroom	TVAE	0.148±0.066	**0.777±0.083**
Classification	Mushroom	TabDDPM	0.746±0.023	**0.867±0.080**
Classification	Mushroom	TabPFN	-0.460±0.200	**0.961±0.021**
Classification	diabetes	CTGAN	0.666±0.239	**0.749±0.085**
Classification	diabetes	Gaussian Copula	0.559±0.146	**0.682±0.175**
Classification	diabetes	TVAE	0.748±0.113	**0.748±0.098**
Classification	diabetes	TabDDPM	**0.710±0.186**	0.656±0.191
Classification	diabetes	TabPFN	**0.603±0.199**	0.566±0.184
Classification	german_credit	CTGAN	0.607±0.045	**0.788±0.156**
Classification	german_credit	Gaussian Copula	0.607±0.070	**0.729±0.180**
Classification	german_credit	TVAE	0.790±0.071	**0.804±0.074**
Classification	german_credit	TabDDPM	0.555±0.123	**0.804±0.186**
Classification	german_credit	TabPFN	0.328±0.107	**0.566±0.172**
Classification	heart_disease	CTGAN	0.314±0.401	**0.690±0.070**
Classification	heart_disease	Gaussian Copula	-0.256±0.323	**0.725±0.171**
Classification	heart_disease	TVAE	0.456±0.352	**0.696±0.169**
Classification	heart_disease	TabDDPM	0.014±0.387	**0.630±0.196**
Classification	heart_disease	TabPFN	0.467±0.236	**0.480±0.230**
Classification	obesity	CTGAN	0.178±0.090	**0.947±0.027**
Classification	obesity	Gaussian Copula	0.012±0.044	**0.920±0.031**
Classification	obesity	TVAE	0.288±0.166	**0.960±0.011**
Classification	obesity	TabDDPM	0.250±0.164	**0.965±0.019**
Classification	obesity	TabPFN	**0.846±0.076**	0.798±0.043
Regression	abalone	CTGAN	0.215±0.194	**0.744±0.182**
Regression	abalone	Gaussian Copula	0.708±0.090	**0.838±0.083**
Regression	abalone	TVAE	0.612±0.146	**0.852±0.074**
Regression	abalone	TabDDPM	**0.873±0.066**	0.813±0.118
Regression	abalone	TabPFN	0.795±0.084	**0.864±0.052**
Regression	california_housing	CTGAN	0.169±0.224	**0.441±0.336**
Regression	california_housing	Gaussian Copula	0.664±0.084	**0.794±0.059**
Regression	california_housing	TVAE	0.643±0.088	**0.680±0.180**
Regression	california_housing	TabDDPM	**0.995±0.004**	0.992±0.004
Regression	california_housing	TabPFN	**0.969±0.019**	0.968±0.013
Regression	concrete_strength	CTGAN	0.257±0.186	**0.932±0.021**
Regression	concrete_strength	Gaussian Copula	0.516±0.268	**0.936±0.018**
Regression	concrete_strength	TVAE	0.314±0.338	**0.951±0.01**6
Regression	concrete_strength	TabDDPM	0.833±0.034	**0.951±0.016**
Regression	concrete_strength	TabPFN	0.895±0.029	**0.929±0.032**
Regression	diabetes_regression	CTGAN	0.091±0.389	**0.797±0.146**
Regression	diabetes_regression	Gaussian Copula	-0.141±0.132	**0.785±0.162**
Regression	diabetes_regression	TVAE	**0.786±0.083**	0.769±0.089
Regression	diabetes_regression	TabDDPM	0.490±0.168	**0.502±0.185**
Regression	diabetes_regression	TabPFN	**0.665±0.147**	0.635±0.151
Regression	wine_quality	CTGAN	0.262±0.142	**0.663±0.099**
Regression	wine_quality	Gaussian Copula	-0.222±0.148	**0.720±0.127**
Regression	wine_quality	TVAE	0.562±0.174	**0.729±0.155**
Regression	wine_quality	TabDDPM	0.518±0.345	**0.705±0.110**
Regression	wine_quality	TabPFN	0.912±0.047	**0.937±0.024**

B Tuning hyperparameters of generative models

To decouple generator selection from evaluation, we first tune each generative model on a fixed *reference split* (80/20 with a fixed seed), using Optuna to minimize the Wasserstein distance between real validation data and a synthetic sample of matching size.

- **CTGAN [Xu et al., 2019]:**

  Introduction: A GAN-based model for tabular data that employs conditional sampling and mode-specific normalization to handle mixed data types.

  Implementation: Implemented via the `sdv` library [Patki et al., 2016] as `CTGANSynthesizer` with integrated Optuna tuning.

  Hyperparameter Space:
  - **epochs**: integer, range $[300, 1500]$, step 100
  - **batch_size**: categorical, choices $\{32, 64, 128, 256, 512, 1024\}$
  - **embedding_dim**: categorical, choices $\{64, 128, 256\}$
  - **gen_width**: categorical, choices $\{128, 256, 512\}$ (mapped to a 2-layer generator MLP)
  - **disc_width**: categorical, choices $\{128, 256, 512\}$ (mapped to a 2-layer discriminator MLP)
  - **pac**: categorical, choices $\{1, 2, 4, 8, 16, 32, 64\}$
  - **generator_lr**: log-uniform, range $[10^{-4}, 5 \times 10^{-4}]$
  - **discriminator_lr**: log-uniform, range $[10^{-4}, 5 \times 10^{-4}]$

- **TVAE [Xu et al., 2019]:**

  Introduction: A variational autoencoder adapted to mixed-type tabular data, using mode-specific normalization and symmetric encoder-decoder networks.

  Implementation: Integrated via the sdv library as TVAESynthesizer, with Optuna-based tuning of architectural and training parameters.

  Hyperparameter Space:
  - **epochs**: integer, range $[300, 1500]$, step 100
  - **batch_size**: categorical, choices $\{32, 64, 128, 256, 512, 1024\}$
  - **embedding_dim**: categorical, choices $\{64, 128, 256\}$
  - **compress_width**: categorical, choices $\{128, 256, 512\}$ (used as encoder hidden size, repeated for two layers)
  - **decompress_width**: categorical, choices $\{128, 256, 512\}$ (used as decoder hidden size, repeated for two layers)

- **Gaussian Copula [Patki et al., 2016]:**

  Introduction: A copula-based model that represents dependencies between variables via a Gaussian copula while modeling marginal individually.

  Implementation: Implemented via the `sdv` library as `GaussianCopulaSynthesizer`.

  Hyperparameter Space:
  No Optuna tuning is applied; all internal parameters are left at library defaults.

- **TabPFN: [Ma et al., 2024]**

  Introduction: An energy-based generator that uses TabPFN as a pretrained predictor and samples synthetic points via Stochastic Gradient Langevin Dynamics (SGLD) in the input space.

  Implementation: Integrated via the `tabpfgen` library. TabPFGen operates directly on the training data and does not expose an Optuna objective.

  Hyperparameter Space:
  - **n_sgld_steps**: integer, default 500
  - **sgld_step_size**: float, default 0.01
  - **sgld_noise_scale**: float, default 0.01

- **TabDDPM [Kotelnikov et al., 2023]:**

  Introduction: A denoising diffusion probabilistic model for tabular data that iteratively denoises Gaussian noise to recover realistic samples.

  Implementation: Implemented via the `synthcity` [Qian et al., 2023] ddpm plugin and integrated with Optuna tuning on hyperparameter space.

  Hyperparameter Space:
  - **training_iterations** (n_iter): integer, range $[500, 1000]$, step 100
  - **learning_rate**: log-uniform, range $[10^{-4}, 10^{-2}]$
  - **batch_size**: categorical, choices $\{64, 128, 256, 512, 1024\}$
  - **diffusion_timesteps**: categorical, choices $\{50, 100, 200, 500, 1000\}$
  - **dim_hidden**: categorical, choices $\{64, 128, 256, 512\}$
  - **depth**: integer, range $[1, 4]$, step 1
  - **dropout**: float, range $[0.0, 0.3]$