Robust Detection of Synthetic Tabular Data under Schema Variability

G. Charbel N. Kindji[1,2],
Elisa Fromont[2],
Lina M. Rojas-Barahona[1], **Tanguy Urvoy**[1]
[1]Orange Labs Lannion
first.last@orange.com
[2]Université de Rennes, CNRS, Inria, IRISA UMR 6074
first.last@irisa.fr

Abstract

The rise of powerful generative models has sparked concerns over data authenticity. While detection methods have been extensively developed for images and text, the case of tabular data, despite its ubiquity, has been largely overlooked. Yet, detecting synthetic tabular data is especially challenging due to its heterogeneous structure and unseen formats at test time. We address the underexplored task of detecting synthetic tabular data in the wild, where tables have variable and previously unseen schemas. We introduce a novel datum-wise transformer architecture that significantly outperforms the only previously published baseline, improving both AUC and accuracy by 7 points. By incorporating a table-adaptation component, our model gains an additional 7 accuracy points, demonstrating enhanced robustness. This work provides the first strong evidence that detecting synthetic tabular data in real-world conditions is not only feasible, but can be done with high reliability.

Introduction

Over the past five years, deep learning-based generative models have surged in popularity (Bengesi et al. 2024), raising significant concerns about their potential misuse (Marchal et al. 2025), including opinion manipulation, fraud, and harassment. In response, numerous detection methods have been developed for uniformly structured media such as images and text (Ciftci, Demir, and Yin 2020; Yu et al. 2025). However, detecting synthetic tabular data remains largely underexplored, despite the modality's importance in high-stakes domains where data integrity is crucial. It also presents unique challenges, such as heterogeneous structures, diverse feature types and distribution, and variable table sizes. Additionally, an effective synthetic tabular data detector must be *table-agnostic*, meaning it should function independently of a fixed table structure. This requirement disqualifies most state-of-the-art tabular predictors, including (Breiman 2001; Chen and Guestrin 2016; Prokhorenkova et al. 2018), as well as recent transformer-based models tailored to specific tabular structures (Huang et al. 2020; Arik and Pfister 2021; Somepalli et al. 2022).

(Kindji et al. 2025) has categorized the detection of synthetic tabular data into three levels of "wildness": (i) *Same-table* detection: Identifying synthetic data within a single table structure, as for the *Classifier Two-Sample Test* (C2ST) (Lopez-Paz and Oquab 2016). In this setup, the detector does not need to be table-agnostic. (ii) *Cross-table* detection: Identifying synthetic data across multiple tables (e.g. training and testing both on *Adult* and *Insurance* tables). This setup requires a table-agnostic detector that generalize across different tables within a predefined corpus. (iii) *Cross-table shift* detection: Handling deployment scenarios where the tables encountered at inference differ from those seen during training (e.g., training across both *Adult* and *Insurance*, and evaluating across both *Higgs* and *Abalone*). Each level presents an increasing challenge, and our goal in this paper is to address the most challenging one: the *cross-table shift*. In this setting, detecting synthetic versus real rows is framed as a binary classification problem. Our contributions are as follows.

- We introduce a novel transformer-based architecture that is both table-agnostic and invariant to column permutations, addressing the critical need for robustness in real-world deployment.
- We explore a new type of distribution shift: the cross-table shift with a domain component (e.g. science vs. finance domain tables). This involves handling tables that differ between training and inference, both in terms of structure and domains, adding complexity to the detection task. We refer to this as *cross-domain table shift*.
- We incorporate a *table adaptation* strategy inspired by (Ben-David et al. 2010), resulting in notable performance improvements over existing baselines.

Our architecture, being table-agnostic and invariant to column permutations, has the potential to be applied to tasks beyond synthetic data detection, such as regression and classification on tabular data. We present the related work in Section , followed by the detailed description of our model in Section . We then present our experimental setup, results and limitations in Sections and . Finally, we conclude and outline future research directions in Section .

Related Work

Recent research on tabular data has increasingly shifted toward the development of foundation models (Kim, Grinsztajn, and Varoquaux 2024; Iida et al. 2021; Herzig et al.

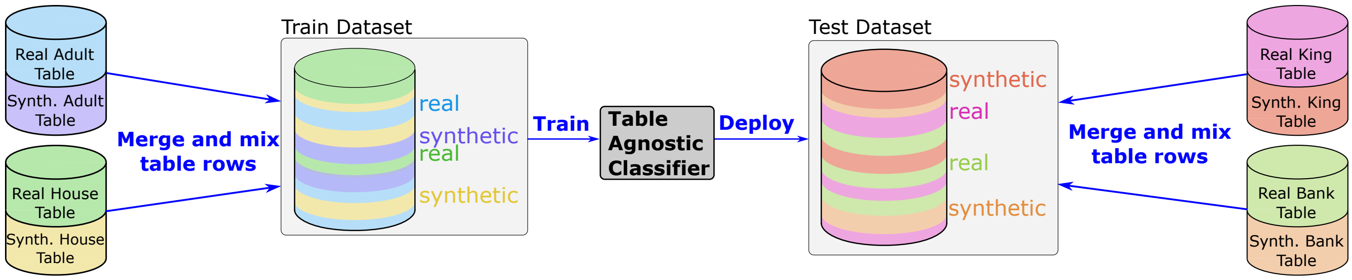

Figure 1: *Cross-table shift* protocol: the real-vs-synthetic detector is trained on a mixture of table rows and tested/deployed on a mixture from holdout tables.

2020; Liu et al. 2022), inspired by the impressive progress in text and vision. These models aim to produce table-agnostic representations through pretraining on diverse tables, with the goal of generalizing across tasks and schemas. However, most of them break this table-agnosticism at inference: they rely on fixed schemas for finetuning and deployment, making them unsuitable for settings where training and test rows come from different tables. In this work, we consider the synthetic tabular data detection problem as a row-by-row classification task and we study a cross-table setup (Figure 1) where no schema alignment or retraining is possible, and generalization across heterogeneous table structures is required. In such settings, the main challenge is not the classifier itself, standard models could suffice, but rather the design of robust and transferable representations. As such, the following related work will primarily focus on approaches to tabular representation learning under schema variability.

Many models achieve table-agnostic pretraining via text-based encodings. TaBERT (Yin et al. 2020) and TABBIE (Iida et al. 2021) rely on the seminal BERT LLM to encode tables. TAPAS (Herzig et al. 2020) and TAPEX (Liu et al. 2022) adapt this approach for question answering. TabuLa-8B (Gardner, Perdomo, and Schmidt 2024) converts table rows to text for LLM finetuning, while STab (Hajiramezanali et al. 2022) and STUNT (Nam et al. 2023) emphasize generalization through data augmentation and meta-learning. Others like Xtab (Zhu et al. 2023), UniTabE (Yang et al. 2024), TransTab (Wang and Sun 2022), PORTAL (Spinaci et al. 2024), and CARTE (Kim, Grinsztajn, and Varoquaux 2024) instead rely on *type-specific* encoders and require a fixed schema across training and testing. To tackle this problem, we introduce the *Datum-wise Transformer*, a lightweight transformer trained on individual rows. It textualizes each feature and uses separate embeddings per column, achieving column permutation invariance and flexibility across table structures. The model is designed for row-level training and inference on any table, with no assumption of schema consistency.

Our approach contrasts with traditional BERT-like tabular encoders such as *Flat Text* (Kindji et al. 2025) and TaBERT, which apply positional encodings across entire rows and thus remain sensitive to column order. While models like PORTAL, TransTab, and UniTabE also implement a form of independent feature encoding, they typically rely on partially handcrafted strategies, often conditioned on feature types or metadata. In contrast, our method learns feature representations directly from raw input without requiring data-type-specific mechanisms. Additionally, many BERT-based tabular models use 768-dimensional embeddings inherited from the original architecture. In contrast, our 192-dimensional representation aligns with our lightweight Transformer design, reducing computational and memory costs. This demonstrates that robust feature learning and permutation handling can compensate for reduced capacity, challenging the need for high-dimensional embeddings.

Table-agnostic Datum-wise Transformer

In the following, we describe our method for detecting synthetic tabular data. First, we describe our *datum-wise* architecture, which is designed to generate effective representations for the synthetic data detection task. We then describe the procedure used for table adaptation to further improve performance.

Datum-Wise Transformer Architecture The proposed detector uses two transformers as its backbone: a *datum transformer* and a *row transformer*. The *datum transformer* processes batches of text *datums*, and the *row transformer* works on a pooled *datum* representation. The whole pipeline and architecture are described in Figure 2.

Each table row is converted into text (i.e. `datums`), which is the concatenation of `<column>:<value>` strings. The `datums` are then tokenized at a character level. Technically, in the first step (Step 1 in Figure 2), our model applies two levels of padding. *Intra-datum* padding extends the length of each *datum* to match the longest `<column>:<value>` string. Then, *extra-datum* padding adds dummy *datums* to handle varying numbers of *datums* in each table of the training set. Each *datum* is appended with a *CLS* token, serving as a representation of the feature. In the following sections, we refer to these tokens as *CLS-Datums*.

A key architectural feature of our model is the restriction of positional encoding to individual *datums* (Step 2 in Fig. 2). Processing the *datums* independently as a first stage enables an independent "featurization", where the feature encodings are inferred directly from raw data. By processing data at the *datum* level rather than entire rows, the *datum transformer* also benefits from shorter input sequences,

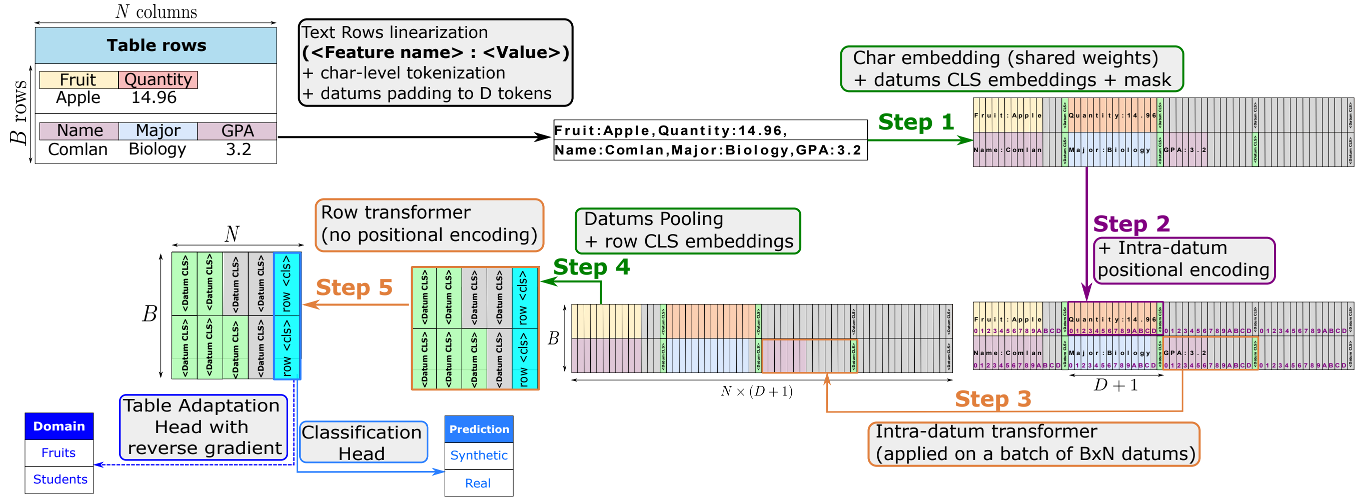

Figure 2: Datum-wise transformer pipeline with table adaptation head.

reducing computational costs. We avoid the reliance on column order induced by global positional encoding, which can cause problems when the detector is applied to tables with different column arrangements. We evaluate the effect of this particular feature in Section . The positional encoding within the *datum transformer* enables it to focus on column-related information without being conditioned on any specific column order. Without it, the transformer would not distinguish the positions of the characters and the entire row would be viewed as a bag of characters (where for instance $elbow : 201.1$ and $below : 1.012$ would be considered identical). While positional encoding matters within a datum, it is important for tabular data, especially in fake data detection, to produce predictions that are invariant to column permutations.

At the end of *Step 2*, each *datum* is encoded as a single 192-dimensional embedding which aggregates both column name and value. This representation serves as input to the *datum transformer* in *Step 3*, and we only retrieve the *CLS-Datums* embeddings from its output (*Step 4*). This datum pooling operation drastically reduces the input size for the subsequent steps. We expect the *CLS-Datums* to provide sufficient information to effectively represent the features. The *CLS-Datums* are appended with an additional row-level *CLS* token (192-dimensional, as for the *CLS-Datums*) that will serve for our classification task. In the following paragraphs, we will refer to this token as *CLS-Target*. The result of this operation is given as input to the *row transformer* (*Step 5*). This transformer does not incorporate any positional encoding as all the position-related information are already processed by the *datum transformer*. We extract the *CLS-Target* from the output and pass it to the classification head. Our detector is trained using a binary cross-entropy loss.

Table Adaptation To enhance our detector's performance "in the wild", we integrate a well-established adaptation strategy that improves a detector's ability to generalize from a specific data distribution to unlabeled target data with a different distributions. In our context, we call it *table adaptation*. Specifically, we employ the gradient reversal techniques from (Ganin and Lempitsky 2015; Saito et al. 2018) to minimize the classifier's reliance on table structures in its embeddings while emphasizing the values within the cells of the tables. For example, some tables may exhibit characteristics that make them easily identifiable (e.g., all numeric versus mixed), encouraging the model to exploit these spurious signals rather than learning to distinguish real from synthetic rows more generally. In practice, we add a table classification head in our architecture from which the gradient reversal is applied down to the representation learning layers. This table classification head predicts the name of the table and also utilizes the *CLS-Target* produced by the *row transformer* for its predictions. This is shown in the bottom left part of the Figure 2. We still use a cross-entropy loss for optimization.

Experimental Setup

Training Data

We took tables from the OpenML repository and the Kaggle platform (see Appendix 1.1). To make the detection task reasonably challenging, we employed state-of-the-art generators for creating synthetic data. Poorly trained or low-quality generators might produce data that is easily distinguishable from real data, hence, we prioritized high-quality generators with carefully optimized hyperparameters for each table. The generators are TabDDPM (Kotelnikov et al. 2023), TabSyn (Zhang et al. 2024), TVAE, and CTGAN (Xu et al. 2019). Hyperparameters were selected following the protocol presented in (Kindji et al. 2024). We employed the same tables as in the mentioned articles to leverage the available pretraining.

To train the detectors to distinguish between real and synthetic data, we constructed each table as a balanced mix of both types of data as illustrated in Figure 1. Specifically, for

[1]https://www.openml.org
[2]https://www.kaggle.com/datasets

Table Domain	Name	Size	#Num	#Cat
Societal & Demographic	Adult[1]	48842	6	9
	HELOC[2]	5229	23	1
	House 16H[1]	22784	17	0
	King[2]	21613	19	1
Consumer & Financial Behavior	Bank Marketing[1]	45211	7	10
	Churn Modelling[2]	4999	8	4
	Diamonds[1]	26970	7	3
	Black Friday[1]	166821	6	4
	Insurance[2]	1338	4	3
Science & Environment	Abalone[1]	4177	7	2
	Cardio[2]	70000	11	1
	Higgs[1]	98050	28	1
	Bike Sharing[1]	17379	9	4
	MiniBooNE[1]	130064	50	1

Table 1: Description of the tables considered in the experiments. We categorized the tables into three main domains (*Social*, *Finance*, and *Science*) for the *cross-domain table shift* evaluation. "Size" is the number of total instances in the table, "#Num" and "#Cat" refer respectively to the number of numerical and categorical attributes.

a given table with n real rows, we added n synthetic rows composed of $n/4$ rows from each of the four generators, resulting in a final table with $2n$ rows. This design maintains a balance between real and synthetic data by ensuring equal contribution from each generator within each table, thereby preventing any imbalance that could bias the detectors. Besides, it avoids introducing additional variables, such as varying real-to-synthetic ratios, that could interfere with the evaluation at this stage. However, it is important to note that the global balance across all tables is not enforced (e.g., one table may contain $2,000$ rows, while another may contain $20,000$). This variability reflects natural differences between tables while preserving local balance within each table.

Baselines

As discussed in Section , very few table encoders can be readily adapted to create a synthetic tabular data detector for real-world use. We view the *Flat Text Transformer* (detailed below) as our closest baseline, given its alignment with our objectives. Nonetheless, we adapted other approaches to our goals to facilitate a more comprehensive comparison. Implementation and training details of our model and the considered baselines are provided in Appendix 1.3.

Flat Text Transformer (Kindji et al. 2025) This model is explicitly designed for the cross-table setting and processes fully textual rows. It uses character-level tokenization with a global positional encoding across the entire row and employs a lightweight BERT-like transformer trained from scratch.

TaBERT embedding (Yin et al. 2020) TaBERT can be reasonably adapted to our setup, due to its use of text-based row linearizations like `<column>|<type>|<value>`. Since it is designed to encode entire tables, we only considered its pretrained version setting the number of rows in each table to 1 (*K=1* [3]). This allows us to use it in our row-by-row classification task. $TaBERT_{base}$ is initialized from $BERT_{base}$, which has 12 heads and 12 layers of attention. Each row in our pool of tables is considered as a single table and encoded by $TaBERT_{base}$. The row context in natural language was generated by prompting *GPT-4O-mini* to describe each table in our pool (see Table 1). $\text{TaBERT}_{\text{base}}$ then processes a row formatted as a table, along with an associated context, as recommended by the authors. We retrieved each row's *CLS* that we used as input for a classification head. As the authors noted, their model can be viewed as an encoder for systems that require table embeddings as input.

BART embedding and fine-tuning (Lewis et al. 2020) BART is a general-purpose pretrained encoder, capable of directly processing our textualized row format for sequence classification. We evaluated both pretrained embeddings and a fine-tuned version of this model, using the same *bart-base* checkpoint with 12 attention heads and 6 layers. In both cases, the BART tokenizer was used to ensure consistency with the model's input format. For the pretrained version, we deployed the model using the same procedure as for TaBERT. We generated embeddings for each row and extracted the first token acting as a *CLS* embedding that we used as input for the classification head. For the embedding baseline, only the weights of the classification head were updated. For the fine-tuned baseline the BART model was also modified.

Among the other pretrained models mentioned in Section , we evaluated PORTAL using its original table-specific protocol, wherein a separate model was trained and deployed for each test table. While effective in this context, we did not include PORTAL as a *cross-table shift* baseline due to the extensive refactoring and tuning required to generalize it across unseen tables.

Detection Setups

Cross-table Shift In this setup, the model is deployed on unseen tables, as illustrated in Figure 1. Note that our implementation involves a *cross-table shift* between the train and validation sets, as well as between the train and test sets.

For the *datum-wise* detector, we evaluated two versions: one with table adaptation and one without. The experiments were conducted under a strict 3-fold cross-validation procedure. Performance of all detectors is reported using the ROC-AUC and accuracy metrics. ROC-AUC offers a threshold-independent measure of a detector's ability to distinguish real from synthetic data, essential in imbalanced or uncertain scenarios. Accuracy complements this by showing performance at a fixed decision threshold at 0.50.

Cross-domain Table Shift We considered an additional setup with the same characteristics as the *cross-table shift* but which involves tables from different domains between training and deployment. The domain distinctions used are

[3]https://github.com/facebookresearch/TaBERT

Model	Metrics	
	AUC	Accuracy
BART-embd	0.50 ± 0.00	0.50 ± 0.00
BART-tuned	0.52 ± 0.03	0.52 ± 0.02
TaBERT-embd	0.51 ± 0.00	0.50 ± 0.00
Flat Text	0.60 ± 0.07	0.52 ± 0.01
Datum-wise	0.67 ± 0.05	0.59 ± 0.08
Datum-wise + TA	$\mathbf{0.69 \pm 0.04}$	$\mathbf{0.66 \pm 0.05}$

Table 2: Cross-table shift AUC and Accuracy performance reported for transformer detectors. Our proposed *Datum-wise* method is evaluated with and without table adaptation (TA). "BART-embd" and "TaBERT-embd" refer respectively to the embeddings produced by BART and TaBERT. "BART-tuned" refers to the fined-tuned version of BART.

presented in Table 1. For simplicity, we refer to the table domains as *Social*, *Finance*, and *Science*. Due to limited computational access, we did not perform cross-validation on this setup. Instead, we report bootstrapped metrics (Efron and Tibshirani 1994) to provide confidence intervals.

Results

We provide the experimental results for the *cross-table shift* setup in Section , for the *cross-domain table shift* one in Section , and for a detailed analysis in Section . We discuss the limitations of our method in Section . The main results are reported in Table 2 but additional results are reported in the text and detailed in the appendices.

Cross-table Shift

The results from our experiments on the *cross-table shift* setup are reported in Table 2. We provide the performance for the considered baselines and the *datum-wise* detector evaluated with and without table adaptation using table names as domains. Our method consistently outperforms all baselines across all metrics, achieving an average AUC of 0.67 and accuracy of 0.59, establishing state-of-the-art results for the *cross-table shift* detection setup. In comparison, the best baseline, *Flat Text*, tailored for this task, reaches an AUC of 0.60 and accuracy of 0.52. We observed consistent improvements in all three folds, though statistical significance testing is not reliable at this scale.

As for BART and TaBERT's embeddings (respectively *BART-embd* and *TaBERT-embd*), we observe that they achieve performance close to random, with an AUC of 0.50 for BART and 0.51 for TaBERT. Both models obtain an accuracy of 0.50. Analyzing the training logs reveals a notable improvement in performance on the training set, with an average AUC of 0.63 for TaBERT's embedding and 0.59 for BART. However, both models struggle to generalize to the validation and test sets.

These observations highlight a broader limitation when repurposing pretrained models designed for language understanding or reasoning over structured text for tasks involving synthetic data detection in tabular domains. Both TaBERT and BART were adapted with care to fit our row-by-row classification setting, leveraging configurations (e.g., TaBERT with *K=1*) and processing pipelines that remain as faithful as possible to their architectural expectations. Nonetheless, the fundamental shift in task (from textual reasoning to real versus synthetic row classification across heterogeneous table structures) presents challenges that these models were not originally optimized to address. In particular, the lack of full-table context and the absence of inter-row dependencies reduce their effectiveness in this setting.

The fine-tuned BART model achieves an average AUC of 0.52, slightly outperforming the *BART-embd* baseline, which scores 0.50. To investigate these limited results, we analyzed embeddings extracted from the decoder output before the classification head on the first fold. A T-SNE (van der Maaten and Hinton 2008) visualization (Appendix 2) showed that the model mainly relies on table-specific characteristics, with clear separations between tables. This was confirmed quantitatively by training an XG-Boost classifier on the 768-dimensional embeddings to predict table names, achieving 0.99 accuracy.

Table Adaptation Strategy In this configuration, a parameter *lambda* regulates the intensity of gradients propagated from the table classification head. This parameter is gradually increased from 0 to 1 over the course of training. Initially, the model undergoes a few iterations to learn the primary target classification task, i.e., distinguishing between real and synthetic data, before being exposed to negative gradients. This gradual introduction avoids early suppression of table-specific features, ensuring the model has time to learn informative patterns for the primary task.

Our initial experiments with the original scheduling approach proposed by (Ganin and Lempitsky 2015) showed it to be overly aggressive for our setup, often leading to early stopping after just a few epochs. To mitigate this issue, we implemented a smoother, cosine-based lambda schedule, which ultimately yielded the best performance, as demonstrated in Table2. With this adjusted table adaptation strategy, the accuracy improved from 0.59 to 0.66, and the AUC increased from 0.67 to 0.69. The simultaneous improvement in both metrics suggests that the model initially relied on table-related features, among other factors. This influence was effectively mitigated through the adapted training strategy. We further analyze the impact of the table adaptation strategy in a detailed analysis presented in Section .

Cross-domain Table Shift

According to Table 2 the *datum-wise* model combined with table adaptation is able to generalize well across different table structures. However, in order to evaluate the ability of the model to generalize across both domains and structures, we partitioned the tables into three domains: *Social*, *Finance*, and *Science* (See Table 1). We then trained the datum-wise classifier on the tables from one domain and evaluated its ability to detect synthetic content on tables from other domains. This *cross-domain table shift* setting is extremely difficult as it combines both a *cross-domain shift* and a *cross-table shift*. As expected, it does not work (AUC around $1/2$

Train \Test	Social	Finance	Science
Social	-	0.49 ± 0.00	0.48 ± 0.00
Finance	0.48 ± 0.00	-	0.50 ± 0.00
Science	0.48 ± 0.00	0.51 ± 0.00	-
SocialAFN	-	0.49 ± 0.00	$\mathbf{0.52 \pm 0.00}$
FinanceAFN	0.50 ± 0.00	-	$\mathbf{0.55 \pm 0.00}$
ScienceAFN	0.50 ± 0.00	0.51 ± 0.00	-

Table 3: Cross-domain AUC for the *datum-wise* classifier trained on one domain and tested on others. This setting combines both a domain shift and a cross-table shift. We report the average performance over 500 bootstrapped tests along with the 95% confidence intervals. "AFN" refers to anonymized features, and noise (perturbed rows).

is equivalent to random decision). The results of this experiment are reported at the top of Table 3. They suggest that cross-table generalization is only possible among semantically similar tables. A limitation of this experiment, however, is that the domain-specific models were only trained on subsets of the dataset we used for the main *cross-table shift* experiment of Table 2. This reduced coverage likely contributes to the poor out-of-domain generalization.

To investigate this problem, we analyzed the embeddings by extracting the *CLS-Target* and visualizing them using T-SNE plots (see Appendix 2). This revealed a strong reliance on table characteristics when predicting the outcome (real or synthetic). In response, we explored various strategies aimed at mitigating this behavior and improving out-of-domain generalization. We anonymized all table features by replacing original column names with generic feature indices (e.g., `feature_<i>:<value>`) to prevent the model from relying on specific column names. Additionally, we simulated realistic noise in synthetic data and replaced 20% of the rows in each synthetic table with noisy versions. Categorical values in synthetic tables were replaced with either a generic placeholder, a random scrambled string drawn from the column's character set, or a shuffled permutation of a randomly chosen existing category (e.g., replacing "apple" with a shuffled "orange", like "aegnor"). Additional details are in Appendix 1.2. Perturbations replace synthetic rows only in the source domain, also maintaining a balance between real and synthetic data.

Each strategy was evaluated individually and in combination. The combination of anonymized features (AF) and added noise (N) in the synthetic data yielded the best performance for the detection task, both in-domain and out-of-domain. The results (with suffix AFN) are reported at the bottom of Table 3. We observe a slight out-of-domain performance improvement when adapting to the *Science* domain; for instance, adapting from *Finance* to *Science* yields an AUC increase from 0.50 to 0.55. Though modest, this gain indicates that the strategy may enhance cross-domain adaptation.

Detailed Analysis

We conduct a detailed analysis to evaluate the impact of individual components within the *datum-wise* detector.

Impact of Table Adaptation To evaluate the impact of table adaptation, we extracted and visualized the *CLS-Target* embeddings from the *row transformer* just before the classification and adaptation heads (Appendix 2).

We computed the average pairwise cosine distance between L2-normalized table centroids in our 192-dimensional embedding space. After adaptation, embeddings became more clustered, reducing inter-table distances from 3.30×10^{-5} to 6.17×10^{-6} (a 81.3% relative decrease) indicating a more unified representation. Although absolute distances are small due to normalization, this relative reduction indicates a positive effect of the table adaptation strategy. Additionally, an XGBoost classifier distinguishing synthetic and real row embeddings dropped in accuracy from 0.99 before adaptation to 0.89 after, further confirming reduced inter-table separability.

Permutation Invariance A widely used method, especially in image processing, to enforce invariance of a predictive model to a group of transformations such as symmetry or rotation, is to augment the training data by applying these transformations randomly to the training instances. In image processing this *data augmentation* procedure is considered essential to improve the generalization ability of the models, but it requires a longer training phase and it does not provide strong guarantees, especially if the transformation space is large. A better option is to use neural architectures that are explicitly designed to be invariant or equivariant to these transformations (Cohen and Welling 2016; Xu et al. 2023). These architectures provide a better generalization, especially when facing distributional shift (Elesedy and Zaidi 2021). Regarding tabular data, there is no spacial ordering relationship between features, it is hence natural to seek is the invariance by permutation of the columns. In (Borisov et al. 2023) the authors propose to train their models on random permutations of the columns, but this strategy is not guaranteed to cover all possible permutations.

Our goal here is to assess the gain of our *datum-wise* model that was explicitly designed for column permutation invariance (referred to as *Datum-wise*) against an equivalent text transformer with global positional encoding (referred to as *Flat Text*). For the *Flat Text* model, we consider two training strategies: one without permutation of the columns (referred to as "Orig.") and one with permutation of the columns (referred to as "Dynamic Perm."). We consider two tasks: synthetic tabular data detection under *cross-table shift* as in Table 2 and a *single table* setting (here, the *Black Friday* table). We evaluate the obtained models on both permuted and non-permuted sets. These results, reported in Table 4, confirm that our *datum-wise* model generalizes better than a classical text transformer with global positional encoding.

In the sub-column "Train (Perm.)" we provide the results obtained when *evaluating* the models on a permuted version of its training data. These results, compared to the sub-column "Train (Orig.)", confirm that permutation alone

Setup	Model	Training Data	Evaluation Data			
			Train	Train (Perm.)	Test	Test (Perm.)
Cross-table Shift	Flat Text	Orig.	$\mathbf{0.74 \pm 0.02}$	0.67 ± 0.03	0.60 ± 0.07	0.60 ± 0.08
		Dynamic Perm.	0.67 ± 0.03	0.66 ± 0.04	0.60 ± 0.08	0.61 ± 0.06
	Datum-wise	Orig.	0.72 ± 0.02	$\mathbf{0.72 \pm 0.02}$	$\mathbf{0.69 \pm 0.04}$	$\mathbf{0.69 \pm 0.04}$
Single Table (Black Friday)	Flat Text	Orig.	0.65 ± 0.00	0.51 ± 0.00	0.64 ± 0.01	0.52 ± 0.01
		Dynamic Perm.	0.51 ± 0.00	0.55 ± 0.00	0.51 ± 0.01	0.54 ± 0.01
	Datum-wise	Orig.	$\mathbf{0.66 \pm 0.00}$	$\mathbf{0.66 \pm 0.00}$	$\mathbf{0.66 \pm 0.01}$	$\mathbf{0.66 \pm 0.01}$

Table 4: AUC and standard deviation results for a text transformer with global positional encoding (*Flat Text*) and our column-permutation invariant model (*Datum-wise*). At the top we use the *cross-table shift* setting as in Table 2, at the bottom we perform the same experiment on a *single-table*: Black Friday. We compare the evaluation on unchanged datasets and column-permuted datasets ("Perm."). "Dynamic Perm." applies a random column shuffling during training. We use 500 bootstrapped tests and 95% confidence intervals.

(without changing the cell values in the tables) strongly degrades the *Flat Text* model performance but does not impact the *Datum-wise* model. The dynamic column permutation strategy ("Dynamic Perm.") seems not to improve significantly the test performance of the *Flat Text* model which remains behind the *Datum-wise* model. A larger scale experiment could reveal a slight improvement of the *Flat Text* model with dynamic column permutation, but the differences we obtain between "Test" and "Test (Perm.)" are not significant for the *cross-table shift* setting. This could be explained by the fact that the test columns names in our test tables do not appear in the training sets or are too different from the ones encountered in the training tables.

We also performed the same experiment on a *single table* (the *Black Friday* table) to see if the dynamic column permutation strategy would improve the generalization ability of the *Flat Text* model when facing the same features names, but it seems on contrary to overfit strongly the position of the columns and their permutations. As expected, the *Datum-wise* model remains insensitive to these perturbations.

Limitations

While our tables are diverse, they may not capture the full spectrum of real-world data, especially from specialized or proprietary domains. Evaluating on more application-specific tables and synthetic generators would better test the generality and robustness of our findings. Moreover, the cross-domain generalization remains a key challenge and points to the need for more data and more advanced adaptation techniques. Additionally, our model is designed to be invariant to column permutations, but we do not systematically study other perturbation types, such as missing values, adversarial modifications, all of which may impact detector reliability in practice. Moreover, we only considered row-by-row detection, but we acknowledge that some statistical properties, especially for time-series or sequential data, may become apparent only when considering multiple rows together. Finally, although the *datum-wise* Transformer is more lightweight than standard BERT-based encoders, we do not provide a detailed evaluation of computational efficiency or scalability for large-scale or real-time deployments. Intermediate pooling strategies, such as extracting additional tokens beyond *CLS-Datums*, may also enhance representation capacity, but remain unexplored in this work.

Conclusion

In this study, we address the underexplored challenge of detecting synthetic tabular data in real-world scenarios, where models must generalize to unseen table structures. We introduced a novel *datum-wise* Transformer architecture that operates on character-level embeddings and employs local positional encoding at the column level. Our method significantly outperforms the sole existing baseline for this task and other purposefully designed competitors. Through a thorough evaluation under both cross-table and cross-domain protocols, we demonstrated that generalizing across table structures presents a greater challenge than domain shift alone. Further, we showed that incorporating a lightweight table adaptation strategy can significantly enhance performance. Our results provide the first compelling evidence that robust detection of synthetic tabular data in real-world conditions is not only possible but can be effectively achieved with tailored architectures and targeted adaptations. This architecture opens up several promising avenues for future research, such as supporting pretraining-finetuning pipelines for tabular prediction tasks, using objectives like Masked Language Modeling (MLM) from TaBERT, or employing few-shot learning strategies like STUNT.

References

Arik, S. Ö.; and Pfister, T. 2021. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 6679–6687.

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Mach. Learn.*, 79(1–2): 151–175.

Bengesi, S.; El-Sayed, H.; Sarker, M. K.; Houkpati, Y.; Irungu, J.; and Oladunni, T. 2024. Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers. *IEEE Access*, 12: 69812–69837.

Borisov, V.; Sessler, K.; Leemann, T.; Pawelczyk, M.; and Kasneci, G. 2023. Language Models are Realistic Tabular Data Generators. In *The Eleventh International Conference on Learning Representations*.

Breiman, L. 2001. Random forests. *Machine learning*, 45: 5–32.

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

Ciftci, U. A.; Demir, I.; and Yin, L. 2020. FakeCatcher: Detection of Synthetic Portrait Videos using Biological Signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

Cohen, T. S.; and Welling, M. 2016. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, 2990–2999.

Efron, B.; and Tibshirani, R. J. 1994. *An introduction to the bootstrap*. Chapman and Hall/CRC.

Elesedy, B.; and Zaidi, S. 2021. Provably Strict Generalisation Benefit for Equivariant Models. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 2959–2969. PMLR.

Ganin, Y.; and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, 1180–1189. PMLR.

Gardner, J. P.; Perdomo, J. C.; and Schmidt, L. 2024. Large Scale Transfer Learning for Tabular Data via Language Modeling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Hajiramezanali, E.; Diamant, N. L.; Scalia, G.; and Shen, M. W. 2022. STab: Self-supervised Learning for Tabular Data. In *NeurIPS 1st Table Representation Workshop*.

Herzig, J.; Nowak, P. K.; Müller, T.; Piccinno, F.; and Eisenschlos, J. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Annual Meeting of the Association for Computational Linguistics*, 4320–4333.

Huang, X.; Khetan, A.; Cvitkovic, M.; and Karnin, Z. 2020. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

Iida, H.; Thai, D.; Manjunatha, V.; and Iyyer, M. 2021. TABBIE: Pretrained Representations of Tabular Data. In *North American Chapter of the Association for Computational Linguistics*, 3446–3456.

Kim, M. J.; Grinsztajn, L.; and Varoquaux, G. 2024. CARTE: Pretraining and Transfer for Tabular Learning. In *International Conference on Machine Learning (ICML)*.

Kindji, G. C. N.; Fromont, E.; Rojas-Barahona, L. M.; and Urvoy, T. 2025. Synthetic Tabular Data Detection In the Wild. In *International Symposium on Intelligent Data Analysis*. Konstanz, Germany.

Kindji, G. C. N.; Rojas-Barahona, L. M.; Fromont, E.; and Urvoy, T. 2024. Under the Hood of Tabular Data Generation Models: the Strong Impact of Hyperparameter Tuning. arXiv:2406.12945.

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

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Jurafsky, D.; Chai, J.; Schluter, N.; and Tetreault, J., eds., *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

Liu, Q.; Chen, B.; Guo, J.; Ziyadi, M.; Lin, Z.; Chen, W.; and Lou, J.-G. 2022. TAPEX: Table Pre-training via Learning a Neural SQL Executor. In *International Conference on Learning Representations*.

Lopez-Paz, D.; and Oquab, M. 2016. Revisiting Classifier Two-Sample Tests. In *International Conference on Learning Representations*.

Marchal, N.; Xu, R.; Elasmar, R.; Gabriel, I.; Goldberg, B.; and Isaac, W. 2025. Generative AI Misuse: A Taxonomy of Tactics and Insights from Real-World Data. In *ICML Workshop: Humans, Algorithmic Decision-Making and Society: Modeling Interactions and Impact*.

Nam, J.; Tack, J.; Lee, K.; Lee, H.; and Shin, J. 2023. STUNT: Few-shot Tabular Learning with Self-generated Tasks from Unlabeled Tables. In *The Eleventh International Conference on Learning Representations*.

Prokhorenkova, L.; Gusev, G.; Vorobev, A.; Dorogush, A. V.; and Gulin, A. 2018. CatBoost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31.

Saito, K.; Yamamoto, S.; Ushiku, Y.; and Harada, T. 2018. Open Set Domain Adaptation by Backpropagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Somepalli, G.; Schwarzschild, A.; Goldblum, M.; Bruss, C. B.; and Goldstein, T. 2022. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. In *NeurIPS 1st Representation Workshop*.

Spinaci, M.; Polewczyk, M.; Klein, T.; and Thelin, S. 2024. PORTAL: Scalable Tabular Foundation Models via Content-Specific Tokenization. In *NeurIPS 3rd Table Representation Learning Workshop*.

van der Maaten, L.; and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605.

Wang, Z.; and Sun, J. 2022. Transtab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35: 2902–2915.

Xu, L.; Skoularidou, M.; Cuesta-Infante, A.; and Veeramachaneni, K. 2019. Modeling Tabular data using Conditional GAN. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Xu, R.; Yang, K.; Liu, K.; and He, F. 2023. $E(2)$-Equivariant Vision Transformer. In *Uncertainty in Artificial Intelligence*, 2356–2366. PMLR.

Yang, Y.; Wang, Y.; Liu, G.; Wu, L.; and Liu, Q. 2024. UniTabE: A Universal Pretraining Protocol for Tabular Foundation Model in Data Science. In *International Conference on Learning Representations (ICLR)*.

Yin, P.; Neubig, G.; Yih, W.-t.; and Riedel, S. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 8413–8426.

Yu, P.; Chen, J.; Feng, X.; and Xia, Z. 2025. CHEAT: A Large-scale Dataset for Detecting CHatGPT-writtEn AbsTracts. *IEEE Transactions on Big Data*, 1–9.

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

Zhu, B.; Shi, X.; Erickson, N.; Li, M.; Karypis, G.; and Shoaran, M. 2023. XTab: cross-table pretraining for tabular transformers. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.