MEASURING LLM SENSITIVITY IN TRANSFORMER-BASED TABULAR DATA SYNTHESIS

Maria F. Davila R.*
Computer Science Department
Carl von Ossietzky University
Oldenburg, Germany

Azizjon Turaev
Computer Science Department
Carl von Ossietzky University
Oldenburg, Germany

Wolfram Wingerath
Computer Science Department
Carl von Ossietzky University
Oldenburg, Germany

ABSTRACT

Synthetic tabular data is used for privacy-preserving data sharing and data-driven model development. Its effectiveness, however, depends heavily on the used Tabular Data Synthesis (TDS) tool. Recent studies have shown that Transformer-based models outperform other state-of-the-art models such as Generative Adversarial Networks (GANs) and Diffusion models in terms of data quality. However, Transformer-based models also come with high computational costs, making them sometimes unfeasible for end users with prosumer hardware. This study presents a sensitivity assessment on how the choice of hyperparameters, such as number of layers or hidden dimension affects the quality of the resultant synthetic data and the computational performance. It is performed across two tools, GReaT and REaLTabFormer, evaluating 10 model setups that vary in architecture type and depth. We assess the sensitivity on three dimensions: runtime, machine learning (ML) utility, and similarity to real data distributions. Experiments were conducted on four real-world datasets. Our findings reveal that runtime is proportional to the number of hyperparameters, with shallower configurations completing faster. GReaT consistently achieves lower runtimes than REaLTabFormer, and only on the largest dataset they have comparable runtime. For small datasets, both tools achieve synthetic data with high utility and optimal similarity, but on larger datasets only REaLTabFormer sustains strong utility and similarity. As a result, REaLTabFormer with lightweight LLMs provides the best balance, since it preserves data quality while reducing computational requirements. Nonetheless, its runtime remains higher than that of GReaT and other TDS tools, suggesting that efficiency gains are possible but only up to a certain level.

1 Introduction

Synthetic data is used when access to real data is limited due to privacy concerns or scarcity [1, 2, 3]. It is popular across domains such as images, text, and tabular data. In this study, we focus on Tabular Data Synthesis (TDS) and define tabular data as consisting of one or more tables, where rows represent records and columns represent features [4, 5].

Synthetic data is only useful if TDS tools are capable of generating high-quality data. Unlike synthetic images, where quality can be assessed visually, the quality of tabular data is harder to measure. It depends heavily on the type of the data and its intended use. As discussed by Goodfellow et al. [6], different applications have different requirements for fidelity, diversity, and privacy. In simple terms, high-quality synthetic data should preserve the key patterns and distributions of the original dataset without duplicating existing records.

In earlier work [7], we introduced a benchmark to evaluate the synthetic data quality generated by TDS tools and their underlying models across six critical dimensions: handling of class imbalance, dataset augmentation, imputation of missing values, privacy preservation, ML utility, and computational performance. These dimensions were chosen based on the main use cases for synthetic data and the ***privacy versus utility trade-off*** [8].

Data utility refers to how well synthetic data can serve the same purpose as the real dataset. For example, if the original data is used to train a classification model, the utility of the synthetic dataset is its ability to train a model with similar

*Corresponding Author: maria.fernanda.davila.restrepo@uni-oldenburg.de

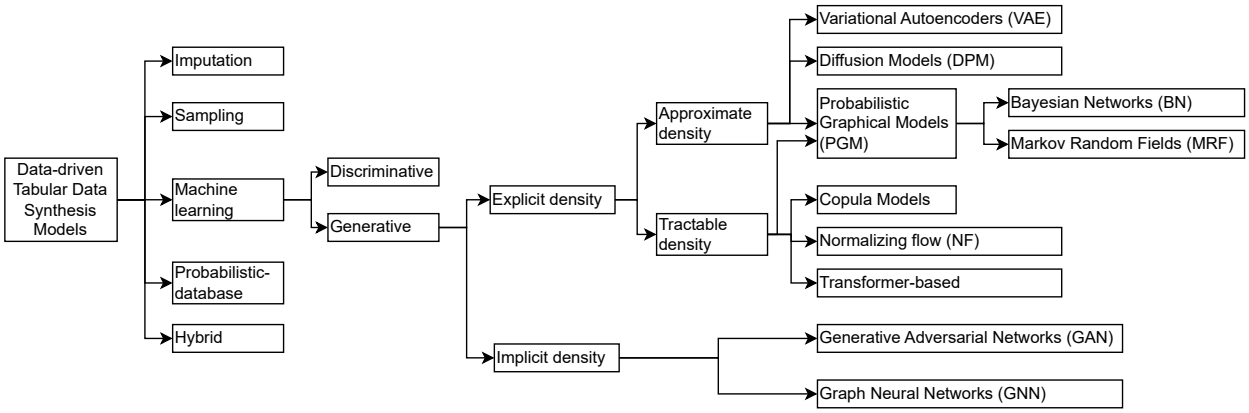

Figure 1: Taxonomy of data-driven TDS models (illustration taken from [7, Fig. 1]).

predictive performance. However, generating synthetic data that protects privacy often comes at the cost of reduced utility [9]. Simple anonymization methods, such as noise injection, can break the patterns in the data [10].

The evaluated tools were selected based on the taxonomy shown in Figure 1. Our results showed that Transformer-based models outperformed other state-of-the-art models in five of the six evaluation dimensions. The only dimension where they underperformed was computational performance, which included CPU usage, GPU usage, memory consumption, and total runtime during synthetic data generation.

These strong results could position Transformer-based tools as the default choice for practitioners. However, their high computational demands limit their accessibility for users with prosumer hardware. Therefore, our motivation is to assess whether it is possible to improve the computational performance of Transformer-based TDS tools, without compromising synthetic data quality. This paper explores whether the quality of synthetic data is heavily influenced by the choice of the language model, and if similar quality can be achieved using models with lower resource requirements.

Our contribution is a sensitivity assessment of how the choice of large language model (LLM) affects the quality and performance of synthetic tabular data generation. We evaluate sensitivity along three key dimensions: runtime, ML utility, and data similarity. The LLM parameters include the model type (such as GPT2 and GPT-NeoX), number of layers, and the TDS tool (REaLTabFormer and GReaT).

The remainder of this paper is organized as follows. Section 2 reviews related work and justifies our choice of tools and evaluation metrics. Section 3 describes the experimental setup. Section 4 presents our findings across the three sensitivity dimensions. Finally, Section 5 concludes the paper and outlines future work.

2 Related Work

This section first gives the theoretical basis to understand Transformer-based TDS models, and further describes the tools used in this assessment, GReaT and REaLTabFormer. In this paper, models refer to the underlying architectures used to generate synthetic tabular data. Models define the theoretical and mathematical principles governing synthetic data generation. Each model is based on a core algorithm, which provides the computational framework for learning and generating data. On the other hand, tools refer to specific implementations of these models. Tools transform theoretical models into practical applications by integrating them into software packages. Finally, this section also presents an introduction to synthetic tabular data evaluation, in order to present the metrics used in the assessment.

2.1 Transformer-based TDS Models

Based on the TDS model taxonomy in Figure 1, Transformer-based TDS models are part of the *Machine Learning*, *generative*, *explicit*, and *tractable density* categories. In this context, **ML** refers to TDS models that learn patterns directly from data rather than relying on predefined rules. **Generative** means that these models are designed not only to discriminate between classes but to generate new synthetic data that follows the same distribution as the original. **Explicit** means that the models specify a concrete probability distribution for the data, in contrast to implicit approaches such as GANs. Finally, **tractable density** means that the probability of any given data point under the model can be computed exactly [11, 12].

Unlike recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) that process data step by step, Transformers use self-attention to consider entire sequences at once, which greatly improves modeling of long-range dependencies. The ***self-attention mechanism*** [13] can determine what words in a sentence, or columns in a dataset, are the most important to understand it. It operates by first transforming each input element into three vectors:

- **Query** ($\mathbf{Q}$): The Query represents what the current element is searching for in other elements.
- **Key** ($\mathbf{K}$): The Key represents the content each element provides.
- **Value** ($\mathbf{V}$): The Value holds the actual representation to be retrieved.

The attention scores are obtained by multiplying Queries with Keys, producing an **attention matrix** $\mathbf{QK}^T$ that measures how relevant the elements are to each other. These scores are multiplied by the Value vectors, and their sum gives the updated representation of each input, composed by the original value and its importance.

LLMs build directly on the Transformer architecture described above, extending its self-attention mechanism to very large scales. An LLM is a neural network of stacked Transformer blocks, where the parameters are the numerical weights that the model learns during training, which are after multiplied by the input token to calculate the Q, K and V for each token. The total number of parameters defines the model's ability to capture patterns in data [14]. This process is known as autoregressive data generation, hence many references categorize this group of models as autoregressive, but we refer to them as Transformer-based because, in TDS, all the relevant tools leverage the Transformer architecture.

Currently, there are a few transformer-based TDS tools, with GReaT [15], and REaLTabFormer [16], being the most popular examples. They both use pre-trained LLMs to learn the patterns in the dataset. GReaT transforms tabular data into natural language sentences by doing *textual encoding* of the dataset, which means it turns every row into a sentence using a subject-predicate-object structure. Then, each row is permuted, which means the order of the columns is shuffled to remove artificial order. After, the model is fine-tuned using the input dataset. This means that the pretrained LLM is further trained on the textualized tabular data, adjusting its parameters to better capture the statistical properties and dependencies within the dataset. After fine-tuning is complete, GReaT is able to generate new samples by autoregressively predicting column values. Autoregressively predicting means generating each token sequentially by conditioning on the previously generated tokens. Given a set of column-value pairs, the model samples the next token from the learned conditional probability distribution. This is shown in Figure 2.

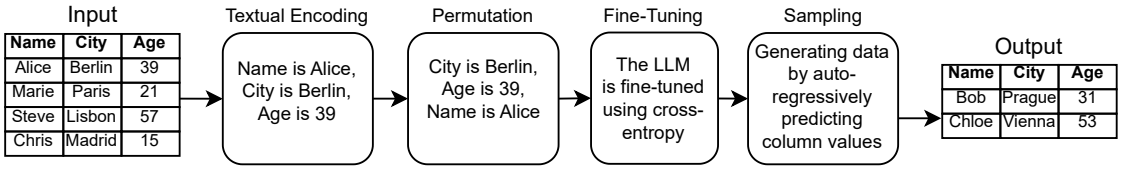

Figure 2: Overview of the synthetic data generation process of the Transformer-based tool, GReaT.

On the other hand, REaLTabFormer [16] uses a two-step approach that first generates a parent table using an autoregressive LLM model, similar to GReaT, and then introduces a second model to generate child tables that maintain relational dependencies. This process is shown in Figure 3, where the parent table generation is similar to that of GReaT, but now there is also a step for child-table generation. Here, the child-table generation follows a sequence-to-sequence (Seq2Seq) approach, where the pretrained model from the parent table acts as an encoder for the relational dataset. First, the generated parent table is used to condition the child table generation. This is then fed into a transformer decoder, which autoregressively predicts the child table values while ensuring relational constraints are maintained.

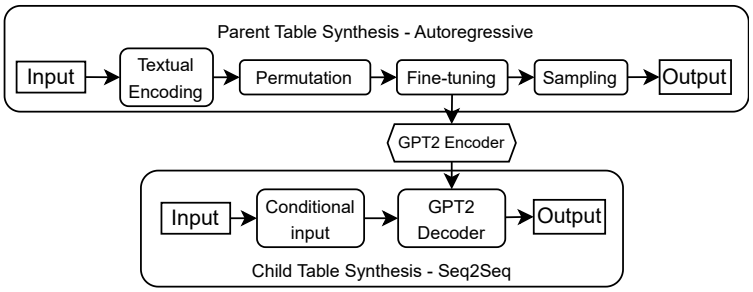

Figure 3: Overview of the REalTabFormer tool, using two models to generate data for relational tables.

2.2 Synthetic Tabular Data Evaluation

The evaluation of synthetic tabular data remains challenging, as no universally accepted metrics exist, which measure the quality of the synthetic data generated with TDS tools [17]. We argue that the reason for this is that the choice of evaluation method largely depends on the intended use of the data [7], which makes a single metric insufficient to capture all relevant aspects.

Many metrics have been proposed [18, 19, 20], each targeting specific properties such as correlation structure, distributional similarity, or machine learning utility. One early effort to standardize evaluation is TabSynDex [17], which aggregates several measures into a single score.

The Synthetic Data Vault (SDV) [21] also provides the open-source library `sdmetrics` [22], implementing a broad set of metrics, including Correlation Similarity, Key Uniqueness, Missing Value Similarity, and Sequence Length Similarity, among many others. In this work, we use a selection of metrics from `sdmetrics` and extend them to support our sensitivity analysis, as described in Section 3.

3 Methodology for Evaluating TDS Tool Sensitivity to Hyperparameters

This section first describes how we selected the LLMs included in the assessment. Then, it presents the method we developed to calculate the size of the LLM for each experiment, which is our approach to make the experiments comparable between LLM families. Finally it presents the experimental setup in terms of the datasets used for evaluation, their pre-processing, and the hardware on which the experiments were conducted.

3.1 LLM Selection

The LLMs currently available can be divided into three main types: (i) decoder-only models, such as the GPT [23] family, (ii) encoder-only models, such as BERT [24], and (iii) encoder-decoder models, such as T5 [25]. Each architecture is different and works for different types of tasks. Encoder-only models, such as BERT, process the entire input sequence at once, in this case all columns and rows. After, it uses bidirectional attention, which means it tries to learn the relationships of the entire table at once [24].

Decoder-only models, handle tabular data like a sequence. They generate one column value at a time, and use causal attention, which means each prediction only sees what came before. This is ideal for tabular data, where the model learns the step-by-step pattern of how real rows are structured [23]. Encoder-decoder models, like T5, include both parts. First, the encoder reads and understands the input table. Then, the decoder takes that understanding and generates new values or a transformed version of the table [25]. However, this type of LLM has not been implemented for TDS.

For this study, we focused on decoder-only LLMs, which are commonly used in TDS tools. All selected models were sourced from the Hugging Face Model Hub [26], which provides a a platform for accessing and deploying pretrained Transformer models. The LLMs selected for this sensitivity analysis were chosen to cover a wide range of sizes and configurations. The final selection is shown in Table 1.

Table 1: Selected LLMs for the sensitivity assessment and their standard configurations.

LLM	Standard Layers	Hidden Dimension	Attention Heads	Standard Parameters
GPT-2 [23]	12	768	12	124M
LLaMA [27]	32	4096	32	7B
GPT-Neo [28]	32	2048	16	2.7B
GPT-BigCode [29]	40	5120	40	15B

3.2 LLM Size Estimation

To ensure a fair comparison across different LLMs, we estimated the model size in each experiment, defined as the number of trainable parameters. With a single model family (e.g., GPT-2), this would be straightforward, for example, reducing the number of layers from 12 to 6 approximately cuts the parameter count in half. However, across different LLMs, the number of parameters also depends on the hidden dimension, number of attention heads, and feed-forward network size. Therefore, the first step in our method was to develop a procedure for consistently estimating LLM size across experiments.

In a transformer, the hidden dimension H is the size of the internal token representation. During self-attention, the input of size H is projected into queries, keys, and values. Attention scores (query $\times$ key) weight the values, and each head outputs a vector of size H/heads. The head outputs are concatenated and projected back to size H, so the hidden dimension is preserved before and after attention [13].

A layer is one complete block of computation: multi-head self-attention followed by a feed-forward network. Each layer maps vectors of size H back to H, and stacking many layers refines the token representations, enabling deeper reasoning and more complex pattern extraction.

This is the reasoning we use to estimate the size of an LLM: the number of parameters grows roughly with the square of the hidden dimension (H^2) and linearly with the number of layers (L). The quadratic dependence on H comes from the fact that most of the parameters are in weight matrices that map vectors of size H to other vectors of size H. Each such matrix has $H \times H$ entries, so doubling the hidden dimension multiplies the parameter count by four. The linear dependence on L comes from stacking more layers, since each layer contributes approximately the same number of parameters. Therefore, both the width of the model (hidden dimension) and its depth (layers) directly determine its total parameter count, and thus its overall size.

$$\text{Size}_{LLM} \approx c \cdot L \cdot H^2 \tag{1}$$

where the constant c changes for each LLM family. In order to find constant c for each LLM family, we used the fact that we know the number of parameters of the standard LLMs, their number of layers and hidden size. Table 2 shows constant c for each LLM family and the estimated size of the LLM for each experiment.

Table 2: Experiment configurations, showing the LLM family, the number of layers, hidden dimension, best constant c, and estimated LLM size.

LLM	**Experiment Layers** (L)	**Hidden Dimension** (H)	**constant** (c)	**LLM Size** ($Size_{LLM}$)
GPT-2	6	768	18	57M
GPT-2	12	768	18	113M
GPT-Neox	1	2048	12	528M
GPT-Neo	2	2048	20	151M
GPT-Neo	4	2048	20	302M
GPT-Neo	6	2048	20	453M
GPT-Neo	8	2048	20	604M
GPT-J	1	4096	13	218M
GPT-BigCode	12	5120	14	3.46B
GPT-BigCode	6	5120	14	1.73B
LLaMA	2	4096	13	403M
LLaMA	1	4096	13	201M

3.3 Datasets

For the evaluation, we selected four widely used tabular datasets: *adult* [30], *customer* [31], *house* [32], and *stroke_healthcare* [33]. The *adult* dataset predicts income level by classifying salaries under and over $50.000, the *customer* dataset models whether customers will leave the bank, the *house* dataset predicts median house values, and the *stroke_healthcare* dataset estimates stroke risk from patient records. These datasets provide a mix of demographic, financial, and healthcare use cases, covering both classification and regression tasks. The dataset characteristics are shown in Table 3.

Table 3: Characteristics of the datasets used in this study.

Dataset	**Rows**	**Columns**	**Categorical**	**Continuous**	**Task**
Stroke_healthcare [33]	5110	11	7	4	Classification
Customer [31]	6400	22	16	6	Classification
House [32]	13690	9	1	8	Regression
Adult [30]	30260	14	9	5	Classification

We preprocessed the datasets to ensure consistent evaluation: (i) Categorical columns were converted into numerical representations using encoding, (ii) Continuous columns were normalized to reduce the impact of scale differences and improve comparability, (iii) Missing values were handled by removing any incomplete records, and (iv) All datasets were shuffled and split to avoid any temporal or positional bias.

All experiments were conducted on a Linux laptop with 32GB RAM, an Intel Core i9-12900H, and an external NVIDIA RTX 4090 GPU (24GB VRAM) in a Razer Core X eGPU enclosure with a 1000W PSU.

4 Results

This section presents the results of the sensitivity analysis along the three evaluation dimensions: runtime, ML utility, and similarity. Each dimension captures a different aspect of the TDS tool's behavior and together they provide a comprehensive view of the trade-offs between efficiency, predictive performance, and data fidelity. The following subsections present the detailed results for each dimension and highlight the main patterns observed across datasets, tools, and LLM configurations.

4.1 Runtime

The first aspect examined in this sensitivity analysis is runtime, defined as the total time required for training and synthetic data generation. Runtime was measured in seconds using a monitoring shell script on Linux. To increase the reliability of the results, each experiment was repeated five times, and the average runtime was reported.

The results are shown in Figure 4, which presents a separate plot for each dataset and LLM. Results are reported for both tools, with ReaLTabFormer shown in orange and GReaT in teal. Each plot combines two components: the runtime, represented as bars, and the size of the LLM, represented as a line.

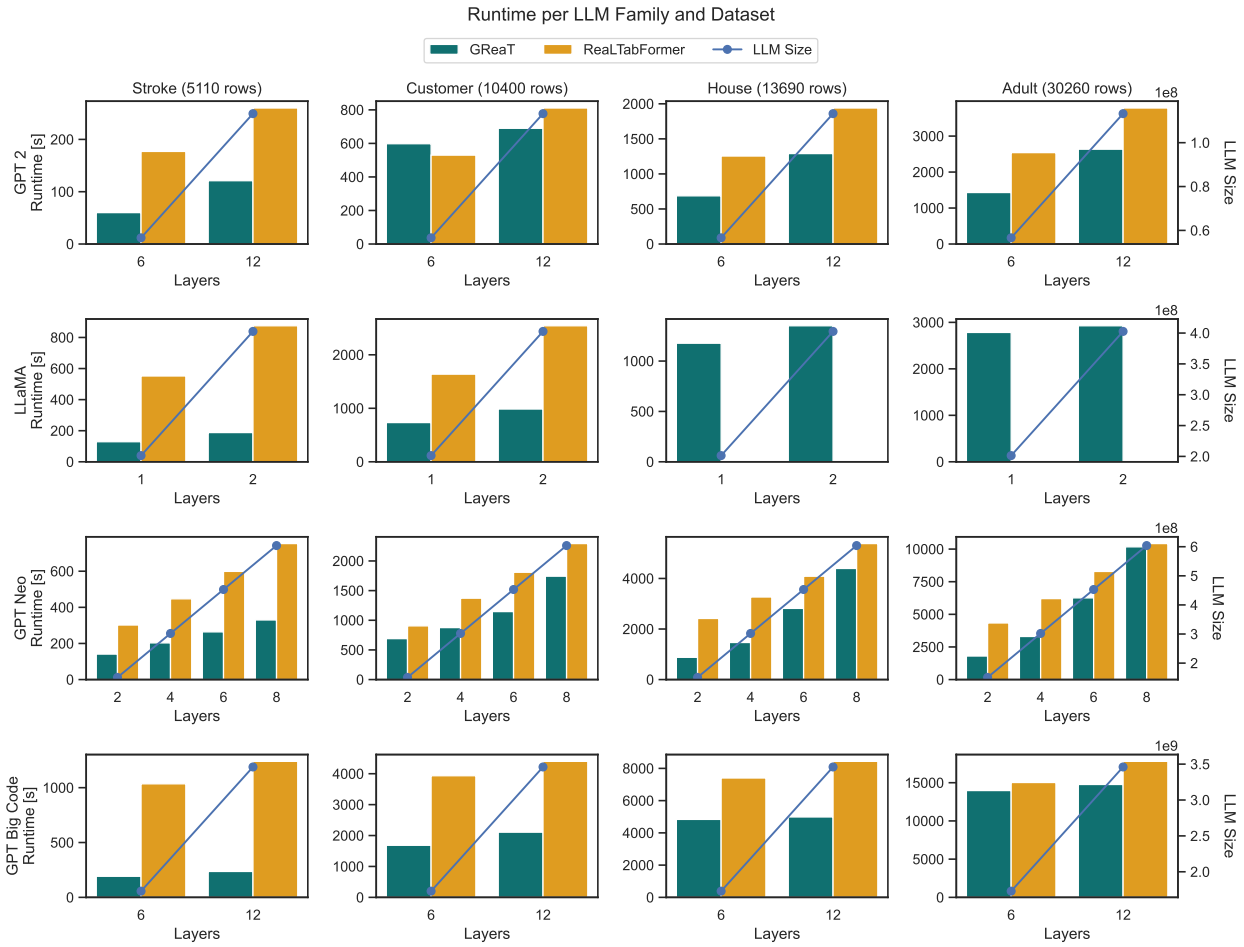

Figure 4: Runtime comparison across LLM families and datasets. Bars show runtime (in seconds) per number of layers and tool (GReaT vs. ReaLTabFormer). The overlaid line indicates the corresponding LLM size (parameters), plotted on the secondary y-axis.

The runtime results in Figure 4 show consistent trends across all datasets that highlight the impact of both the choice of TDS tool and the LLM configuration. An initial observation is that ReaLTabFormer did not converge on the larger datasets when using the LLaMA model.

Overall, GReaT consistently outperforms ReaLTabFormer in terms of runtime, often requiring substantially less time for the same number of layers. The advantage is most pronounced on smaller datasets, such as *stroke* and *customer*, while the gap narrows as dataset size increases. For larger datasets like *house* and particularly *adult*, runtimes for ReaLTabFormer and GReaT begin to converge. This suggests that the relative efficiency of GReaT holds primarily for small to medium-sized datasets.

Another important factor is the number of layers in the LLM configuration, which has a direct and systematic effect on runtime across all datasets and tools. This finding confirms the initial claim of this work that runtime improves with fewer layers. The pattern is particularly clear for ReaLTabFormer, where runtime consistently increases with the number of layers regardless of the dataset size.

In contrast, for GReaT this effect depends on dataset size. On the smaller datasets, runtime grows only marginally with LLM size, suggesting that the tool is less sensitive to model depth. Still, for both tools the trend remains that shallow configurations, such as the two-layer versions of GPT Neo or LLaMA, complete substantially faster than deeper configurations, while models with six or eight layers require significantly more runtime. This scaling effect is visible in every dataset: runtime grows almost linearly with the number of layers for smaller models, and accelerates further in larger configurations such as GPT Big Code. The question is whether these few-layer configurations are still able to capture the patterns in the dataset, or whether this comes at the cost of reduced utility and similarity.

While the absolute runtimes differ depending on the dataset size, the relative ordering of models and tools is preserved: shallow configurations of GPT 2, and LLaMA are always the fastest, whereas GPT Big Code configurations, particularly with 12 layers, are consistently the slowest.

In short, the results illustrate two central drivers of runtime: (i) the efficiency of the underlying tool, with GReaT consistently outperforming ReaLTabFormer, especially on smaller datasets, while this advantage decreases as dataset size grows and runtimes begin to converge on larger datasets, and (ii) the depth of the LLM, with runtime improving for shallower models but increasing substantially with the number of layers. This scaling effect is visible across all datasets: shallow configurations of GPT 2, and LLaMA, are always the fastest, whereas deeper models, particularly GPT Big Code with 12 layers, are consistently the slowest.

4.2 ML Utility

The second evaluation in the LLM selection sensitivity analysis focused on **ML utility**. Here, ML models were trained on both the original datasets and the synthetic datasets generated by GReaT and ReaLTabFormer. Their performance was assessed using standard metrics, which were then compared between real and synthetic training. The goal was to determine whether models trained on synthetic data could achieve performance comparable to, or better than, those trained on real data. ML models and evaluation metrics used are shown in Table 4.

Table 4: ML models and evaluation metrics used for classification and regression tasks.

Task	**ML Models**	**Evaluation Metrics**
Classification	Logistic Regression, Random Forest	Accuracy, Macro-F1
Regression	Linear Regression, Random Forest	R^2

Each experiment with each ML model was run five times to increase reliability. We report the mean score per dataset. The results are shown in Figures 5, and 6. For classification, we plot F1 and Accuracy, for regression, we plot R^2. The bars represent the metrics based on the synthetic data from GReaT and ReaLTabFormer with the different LLMs. The x-axis lists the LLM model and layer count used to generate the synthetic data, and a secondary y-axis shows the capacity on a logarithmic scale.

The first observation regarding ML utility is that, for the smallest dataset (*Stroke*), synthetic data from both TDS tools reach a utility level comparable to the original data. In fact, REaLTabFormer even outperforms the baseline on this task. This is a good and expected result, since one of the goals of synthetic data is to keep or improve model performance when the real dataset is very small. For the other three datasets, however, synthetic data generated with GReaT consistently underperforms compared to the original data, showing a clear loss in utility.

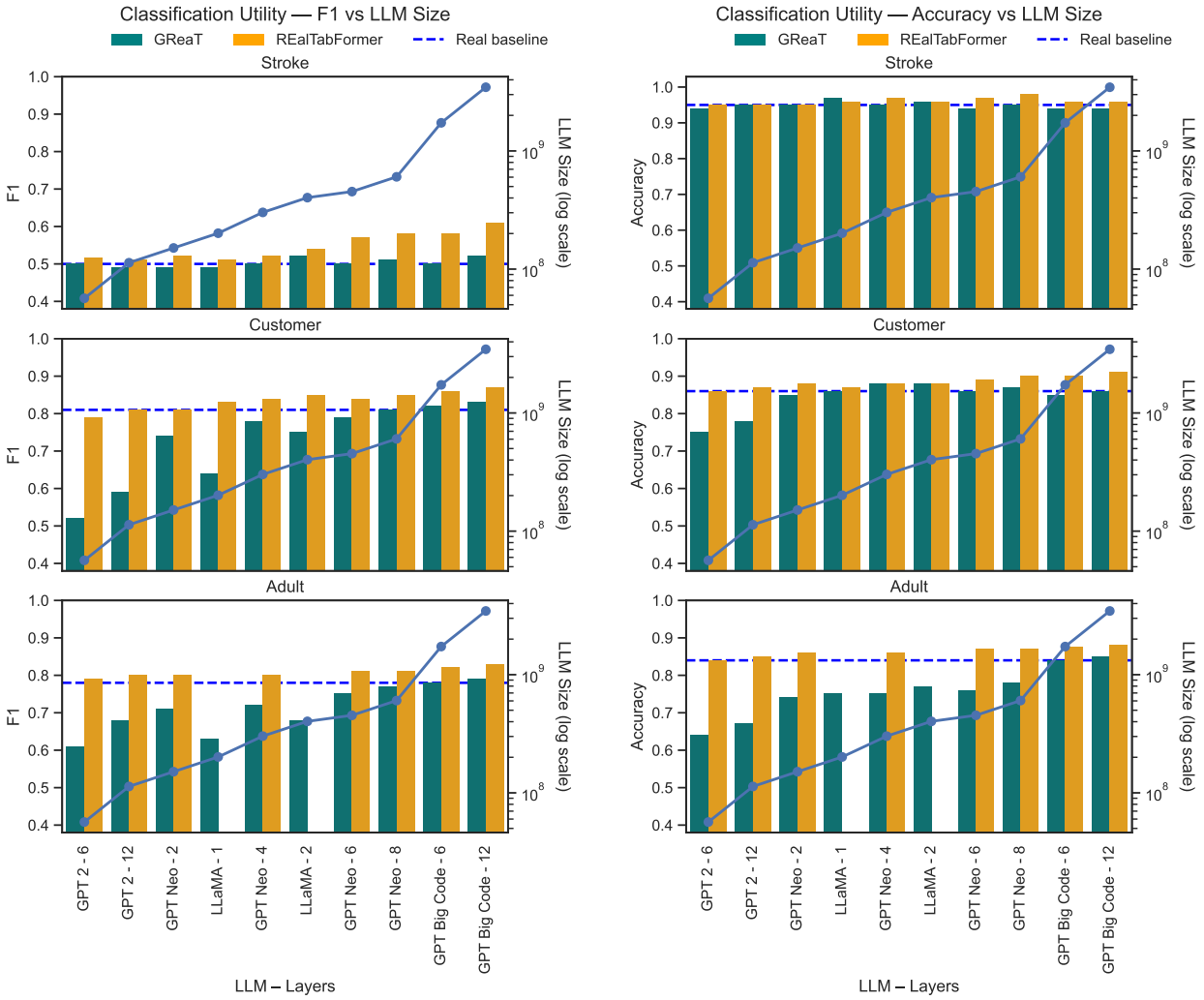

Figure 5: Accuracy and F1 of the classification task (Stroke, Customer, Adult datasets). The real-data baseline is included for reference. A secondary axis shows the LLM size on a log scale starting at 10^7.

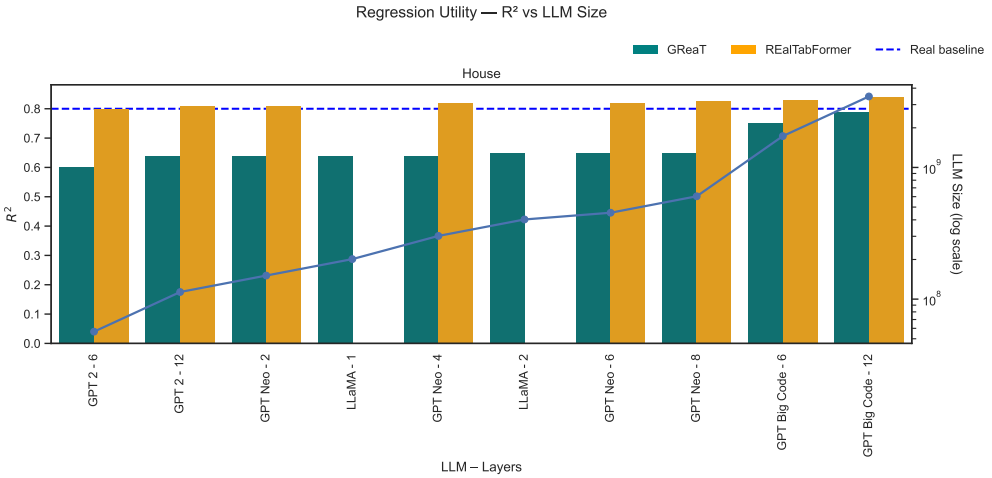

Figure 6: R^2 of the regression task (House dataset). The real-data baseline is included. A secondary axis shows the LLM size on a log scale.

When comparing configurations within the same LLM family, REaLTabFormer shows stable ML utility across layer reductions, suggesting that fewer layers do not decrease downstream performance. In contrast, the utility of the datasets generated with GReaT declines as the number of layers decreases.

4.3 Similarity

The final evaluation aspect in the LLM sensitivity analysis is similarity. To evaluate how similar the synthetic data is to the real data, a classifier-based approach was used. Specifically, a Random Forest model was trained as a Discriminator to distinguish between real and synthetic samples. The model was given a binary classification task where the input data combined both real and synthetic records, each labeled accordingly.

The accuracy of the Discriminator is a measure of similarity between the two distributions. If the classifier achieves high accuracy, it means the real and synthetic data are easily distinguishable, indicating low similarity between real and synthetic distributions. In contrast, if the accuracy is close to 50%, the classifier performs no better than random guessing, suggesting that the synthetic data is statistically similar to the real data. This metric does not assess privacy or utility directly but provides an indication of how realistically the synthetic data mimics the original distribution. The accuracy results for the similarity evaluation is shown in Figure 7.

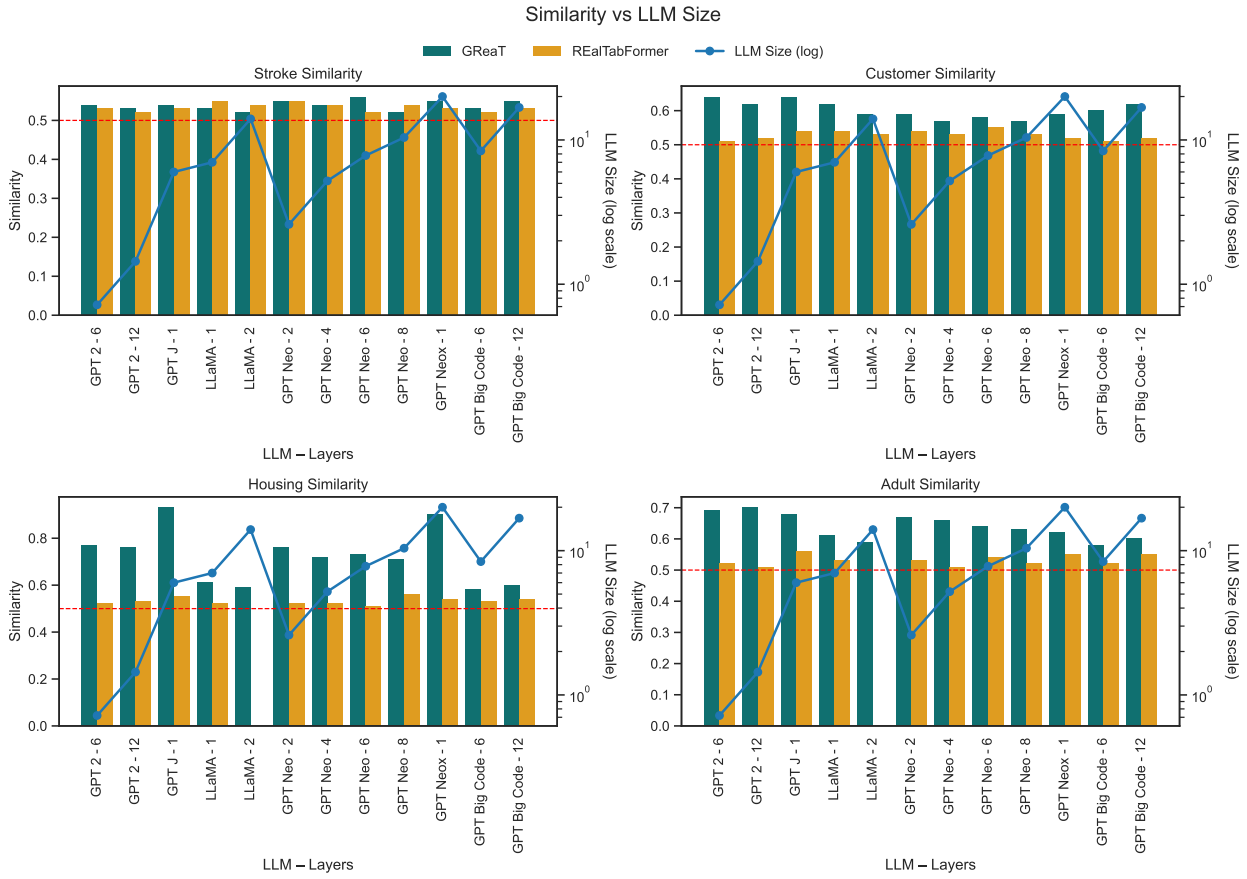

Figure 7: Similarity between real and synthetic datasets for each LLM configuration and TDS tool across the four evaluation datasets. Bars represent similarity scores, with the red dashed line marking the 0.5 threshold. The black line indicates model capacity on a logarithmic scale.

Just like with ML utility, for the smallest dataset *Stroke* both TDS tools achieve the ideal result, where the synthetic datasets have similarity close to 0.5, which means the discriminator is having a hard time identifying the synthetic from the real one. This indicates that for this dataset, both GReaT and REaLTabFormer generate data that closely mimics the original distribution. For REaLTabFormer, this pattern continues across all datasets, the discriminator's accuracy remains close to the 0.5 threshold, suggesting that the generated datasets are consistently hard to distinguish from the real ones, regardless of the LLM configuration or dataset size.

In contrast, the results for GReaT diverge from this trend. For the larger datasets, the discriminator achieves substantially higher accuracy, meaning it can more easily separate synthetic from real records. This indicates that GReaT's synthetic data departs more strongly from the real distribution as dataset complexity increases. The gap between the two tools becomes particularly visible in datasets such as *Adult* and *Customer*, where REaLTabFormer remains close to the ideal similarity value, while GReaT outputs synthetic datasets that are easily identified as artificial.

5 Conclusion

The goal of this work was to assess whether it is possible to reduce the computational requirements of Transformer-based TDS tools while maintaining data utility and similarity to real distributions. We carried out this sensitivity analysis by systematically varying the choice of LLMs across two widely used tools, GReaT and REaLTabFormer, and evaluating them along three dimensions: runtime, ML utility, and similarity.

Our findings include three main insights. First, runtime scaled as expected, with shallower LLMs consistently performing better. GReaT achieved lower runtimes than REaLTabFormer, though the difference became smaller on the larger dataset. However, this efficiency came at the expense of utility and similarity, limiting the benefits of using shallow LLMs with GReaT.

With respect to data quality, we observed that for small datasets both TDS tools behaved as expected, where the ML utility of the synthetic data remained similar or better compared to the real dataset, and the synthetic records could not be easily distinguished from real ones, indicating optimal similarity. For larger datasets, however, only REaLTabFormer maintained high utility and similarity values close to 0.5, whereas GReaT diverged significantly.

As a general conclusion, only REaLTabFormer performs appropriately with simple LLM configurations, showing that the quality of synthetic datasets is not compromised even when using lightweight models with substantially lower runtime than larger ones. Nevertheless, this runtime remains higher compared to GReaT and other TDS tools, suggesting that computational performance can be improved, but only up to a certain level.

While our study provides valuable insights into the sensitivity of Transformer-based TDS tools, it also has several limitations that represent our future work. First, we restricted our evaluation to two Transformer-based tools (GReaT and REaLTabFormer) and a selected set of LLM families. It remains unclear whether the observed trade-offs generalize to other Transformers. Second, our experiments used four datasets. Although these cover both regression and classification tasks, they do not reflect very large-scale, high-dimensional, or domain-specific datasets. Extending the evaluation to larger and more heterogeneous datasets would strengthen the generalizability of our findings. Finally, our analysis focused on runtime, ML utility, and similarity to real data distributions, but did not explicitly assess privacy preservation, which is a central motivation for synthetic data. Future work should incorporate privacy risk evaluation to ensure that efficiency gains do not come at the expense of sensitive data leakage.

References

[1] Alvaro Figueira and Bruno Vaz. Survey on synthetic data generation, evaluation methods and GANs. *Mathematics*, 10(15):1–41, 2022.

[2] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022.

[3] Ju Fan, Tongyu Liu, Guoliang Li, Junyou Chen, Yuwei Shen, and Xiaoyong Du. Relational data synthesis using generative adversarial networks: A design space exploration. *Proceedings of the VLDB Endowment (PVLDB)*, 13(11):1962–1975, 2020.

[4] Edgar F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[5] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press.

[7] Maria Davila R., Sven Groen, Fabian Panse, and Wolfram Wingerath. Navigating tabular data synthesis research: Understanding user needs and tool capabilities. *SIGMOD Record*, 53:18–35.

[8] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment (PVLDB)*, 11(10):1071–1083.

[9] Chang Ge, Shubhankar Mohapatra, Xi He, and Ihab F. Ilyas. Kamino: constraint-aware differentially private data synthesis. *Proceedings of the VLDB Endowment (PVLDB)*, 14(10):1886–1899.

[10] Mengmeng Yang, Chi-Hung Chi, Kwok-Yan Lam, Jie Feng, Taolin Guo, and Wei Ni. Tabular data synthesis with differential privacy: A survey.

[11] David Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. "O'Reilly Media, Inc.". Google-Books-ID: RqegDwAAQBAJ.

[12] Jakub M. Tomczak. *Deep Generative Modeling*. Springer Nature. Google-Books-ID: uidgEAAAQBAJ.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010. Curran Associates Inc.

[14] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025.

[15] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pages 1–18, 2023.

[16] Aivin V. Solatorio and Olivier Dupriez. Realtabformer: Generating realistic relational and tabular data using transformers. *CoRR*, abs/2302.02041:1–17, 2023.

[17] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, Mukund Lahoti, and Pratik Narang. A universal metric for robust evaluation of synthetic tabular data. 5(1):300–309, 2022.

[18] Scott Cheng-Hsin Yang, Baxter Eaves, Michael Thomas Schmidt, Kenneth Brian Swanson, and Patrick Shafto. Structured evaluation of synthetic tabular data.

[19] Patricia A. Apellániz, Ana Jiménez, Borja Arroyo Galende, Juan Parras, and Santiago Zazo. Synthetic tabular data validation: A divergence-based approach. *IEEE Access*, 12:103895–103907, 2024. Conference Name: IEEE Access.

[20] Anton D. Lautrup, Tobias Hyrup, Arthur Zimek, and Peter Schneider-Kamp. Syntheval: a framework for detailed utility and privacy evaluation of tabular synthetic data. *Data Mining and Knowledge Discovery*, 39(1):6, 2024.

[21] SDV. The synthetic data vault. put synthetic data to work!

[22] SDV. SDMetrics | SDMetrics.

[23] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 1(8), 2018. `https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf`.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics, 2019.

[25] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[26] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[28] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, March 2021. If you use this software, please cite it using these metadata.

[29] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023.

[30] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[31] Muhammad Shahid Azeem. Customer churn dataset, 2023.

[32] Luis Torgo. House dataset, 2014.

[33] Federico Soriano. Stroke prediction dataset, 2020.