Data-Efficient Tabular Classification with Transformer-Based Small Language Models

Mario Haddad-Neto
mario.haddad@fpf.br
Fundação Paulo Feitoza
Manaus, AM

Diógenes Silva
diogenes.silva@fpf.br
Fundação Paulo Feitoza
Manaus, AM

Ítalo Caliari
italo.caliari@fpf.br
Fundação Paulo Feitoza
Manaus, AM

Hendrio Bragança
hendrio.braganca@fpf.br
Fundação Paulo Feitoza
Manaus, AM

ABSTRACT

The application of deep learning to tabular data remains an ongoing challenge, with tree-based models such as XGBoost consistently outperforming neural network methods in most real-world scenarios. Recent advances in Large Language Models (LLMs) highlight their ability to generalize to new tasks via few-shot or one-shot prompting, yet questions remain about their utility in structured, tabular domains—particularly for resource-efficient Small Language Models (SLMs). In this work, we systematically evaluate the effectiveness of both large and small transformer-based language models for tabular classification tasks using few-shot strategies. Our approach investigates input serialization schemes and prompt engineering to maximize performance in low-data regimes. Experiments on benchmark datasets—including Diabetes, Heart Failure, and German Credit Risk—demonstrate that well-designed SLMs can approach, and occasionally match, the performance of much larger models, substantially reducing the need for extensive labeled data and retraining. However, further advances are required for language models to consistently rival the strongest tree-ensemble baselines. Our findings support the idea that SLMs, when properly prompted, offer a promising, flexible, and label-efficient alternative for automating and dynamizing machine learning pipelines on tabular data.

KEYWORDS

Large Language Models, Small Language Models, Few-shot Learning, Data Classification, Transformers, Tabular Data

1 INTRODUCTION

With the advent of transformers, originally introduced in natural language processing, many areas of machine learning have seen substantial progress [1–3]. Recent research has demonstrated that large language models (LLMs) exhibit a remarkable capacity for few-shot generalization across a wide range of tasks [4–7]. Despite being trained solely on language modeling objectives, these models can perform competitively on tasks they were not explicitly designed or fine-tuned for.

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2025). Rio de Janeiro, Brazil. Porto Alegre: Brazilian Computer Society, 2025.

ISSN 2966-2753

These models have demonstrated impressive capabilities in tasks like question answering [8–10] and performing summarization [11–13]. A key hypothesis indicates that their ability to generalize to new tasks arises from an implicit multitask learning process [14]. As these models learn to predict the next word, they inadvertently engage with a variety of implicit tasks embedded within their pretraining corpus, enhancing their versatility. For instance, when exposed to a wide range of articles, a model may unwittingly learn the nuances of summarization. This enables large language models to tackle new tasks with natural language prompts [15, 16], surpassing prior multitask generalization efforts.

However, their effectiveness is highly dependent on factors such as the model's size and the specific phrasing of the prompts [17, 18]. Larger models generally possess more parameters, allowing them to capture complex patterns and nuances within the data, enhancing their ability to generalize to new tasks. Additionally, the wording of prompts plays a crucial role; the clarity and structure of the prompt can significantly impact the model's performance.

This proficiency in natural language tasks encourages exploration into how language models might benefit domains like tabular data processing, where traditional deep learning methods have underperformed. Tabular data is ubiquitous in domains ranging from healthcare to climate and finance [19], it presents unique challenges such as mixed data types and limited features. In this paper, our goal is to enhance predictive performance in scenarios with limited training data by leveraging Language Models' ability to effectively integrate this prior knowledge.

As deep learning has revolutionized fields such as computer vision and natural language processing, its impact on tabular data remains limited. Although some deep learning approaches for tabular data have been developed [20–22], they typically underperform compared to ensembles of gradient boosted trees in fully supervised environments [23].

This gap arises due to the lack of locality in data, the presence of variables without meaningful information, mixed data types, and typically fewer variables compared to text or image data. Additionally, there is the increased cost and complexity of adapting deep learning methods to tabular formats, where traditional methods often excel [24, 25].

In summary, while the success of deep learning techniques has transformed many machine learning subfields, significant improvements are still needed in handling tabular data. Our goal is to

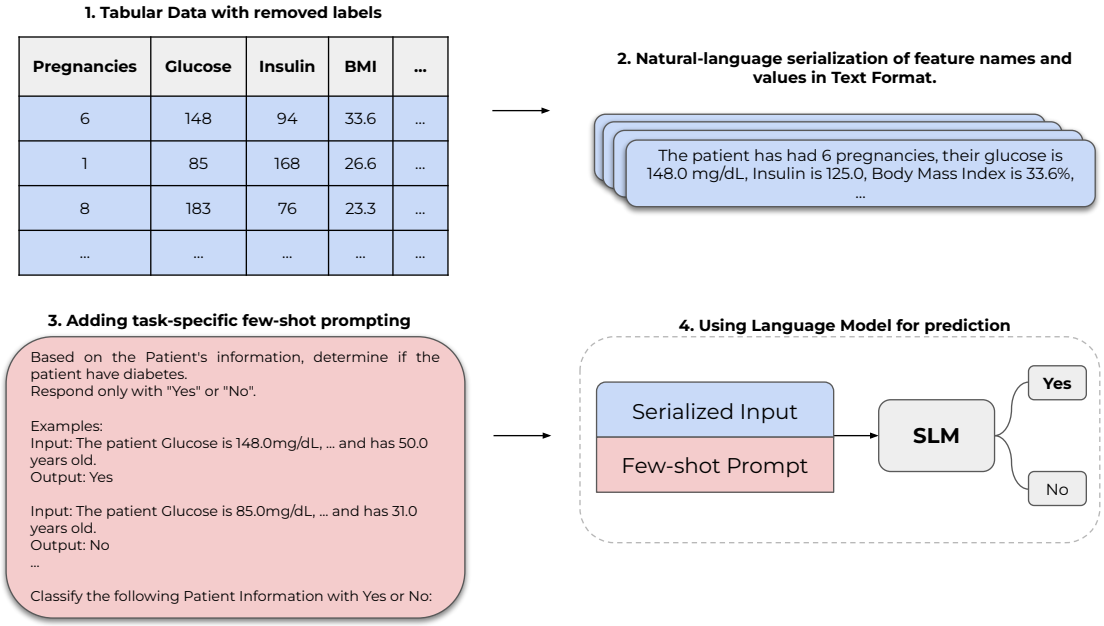

Figure 1: Visual representation of the main workflow of our evaluation. First, we take a tabular dataset and choose a serialization method (in this case, the Text Format). Next, we create a task-specific prompt comprising k serialized examples (shots) and input it into the Language Model, expecting 'Yes' or 'No' as the output.

address this gap by exploring the integration of language models into tabular data classification. This integration could enable more flexible machine learning pipelines, support efficient and iterative training with few examples, and even offer valuable insights for AutoML [26].

In addition, we focus our experiments on Small Language Models, which are transformer-based language models with parameter sizes ranging from 100 million to between 5 to 10 billion parameters [27]. This choice was made to evaluate their capabilities in resource-constrained environments.

The main contributions of this paper are:

- We present a systematic evaluation protocol for comparing various language models—including both large and small models—on benchmark tabular classification tasks using few-shot learning strategies.
- We explore how recent advances in language modeling can be leveraged to build task-specific classifiers for tabular data.
- We provide a thorough benchmarking study that situates language models alongside established conventional methods, offering new point of view into their relative strengths and limitations.
- Our findings support the idea that Small Language Models, when effectively prompted, can serve as strong alternatives to large-scale models for tabular data tasks, offering a promising path towards more accessible and flexible machine learning solutions.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 outlines the technical foundations of language model architectures and serialization methods. Section 4 describes the methodology and experimental setup used in this study. Section 5 reports the results along with their analysis. Finally, Section 6 presents the conclusions and outlines directions for future research.

2 RELATED WORK

In this section, we discuss important works and concepts that inspired our analysis.

2.1 Machine Learning on Tabular Data

The remarkable success of deep learning across various domains has fueled interest in developing methods that can surpass traditional techniques in tabular data tasks. Despite efforts to create differentiable trees that integrate tree ensembles with gradient-based optimization of deep neural networks [28, 29], studies consistently show that gradient-boosted tree ensembles, such as XGBoost [10] and LightGBM [30], continue to outperform these novel architectures. This highlights the need for innovative solutions that leverage the strengths of deep learning in this context.

As language models continue to evolve, performance improvements are being realized across a range of domains, including agentic AI, prompt engineering, and generative information retrieval. These advancements are often facilitated by multitask approaches

[1, 4–6], which enhance the models' versatility and effectiveness. However, improving the ability of language models to understand patterns in tabular data remains a challenge—particularly in few-shot learning scenarios, where these models often fall short of optimal performance.

2.2 Language Models

To address tabular tasks using language models, a primary strategy is leveraging large language models for feature enhancement. Researchers have attempted to enrich tabular data by adding textual information to each row, generating new features [31]. However, this approach relies on named entities, limiting its flexibility. Carballo et al. [32] created feature embeddings for healthcare datasets, still relying on traditional classification methods. Borisov et al. [25] focused on generating realistic data to improve outcomes. Dinh et al. [33] introduced the LIFT method, evaluating fine-tuned GPT-like models across tasks such as classification and regression on synthetic tabular and vision data. These efforts have informed our approach, which aims to advance beyond feature engineering to achieve effective classification using language models.

An essential step in applying language models for few-shot classification is input serialization, which involves converting a table's input row into a natural language representation. Some research includes column data types in this process [20]. Carballo et al. also explored various serialization techniques primarily for deriving feature embeddings [32]. Most works favor simple serialization formats, such as lists or straightforward sentences, as a starting point.

Our work builds on these approaches, focusing on achieving effective and efficient classification with language models in tabular data settings.

3 SMALL LANGUAGE MODELS AND SERIALIZATION

This section outlines the foundational architecture of the small language models central to our analysis. Additionally, we describe the serialization techniques employed during our evaluation to better understand the final comparison.

3.1 Small Language Models Architecture

A small language model is defined as one with a parameter size ranging from 100 million to between 5 to 10 billion [27]. Therefore, a basic SLM architecture is typically decoder-only, as illustrated in Figure 2, which helps reduce costs while maintaining a streamlined design. This architecture is built considering three major structures[3]: the Embedder, Decoder Blocks and the Output block.

Embedder: Given a sequence of words, the text is first tokenized into discrete tokens, which are then mapped to their corresponding integer identifiers (IDs). These IDs are arranged into a matrix $A \in \mathbb{R}^{b \times s}$, where b is the batch size and s is the sequence length. Each token ID is then used to retrieve a corresponding vector from a trainable embedding matrix $E_m \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $|\mathcal{V}|$ is the vocabulary size and d is the embedding dimension. This produces a tensor of token embeddings.

To inject information about the position of each token within the sequence, a positional embedding matrix $E_{pos} \in \mathbb{R}^{s \times d}$ is added to the token embeddings. These positional embeddings are improved during training, since transformers learn positional relations from scratch [34]. The final input tensor to the Transformer is $T \in \mathbb{R}^{b \times s \times d}$, combining both content and position information.

The embedding process can be described by:

$$T[i, j, :] = E_m[A[i, j]] + E_{pos}[j, :] \quad \text{for } i \in [0, b),\ j \in [0, s) \tag{1}$$

Where the indices $i \in [0, b)$ and $j \in [0, s)$ refer to the batch and token positions, respectively.

Decoder Block: Numerous studies have proposed variations of decoder blocks tailored to different architectures and applications. In this section, we focus on the core components of a Transformer decoder, as introduced in the original work by Vaswani et al. [3], which serves as the foundation for modern language models—including smaller-scale variants [27].

Given the input tensor $T \in \mathbb{R}^{b \times s \times d}$, the first operation within the decoder block is the Multi-Head Self-Attention (MHSA) mechanism.

This layer receives three inputs: the query Q, key K, and value V. In the case of self-attention, all three are derived from the same input and thus share the same shape: $Q = K = V \in \mathbb{R}^{s \times d}$. If h denotes the number of attention heads, then Q, K, and V are each linearly projected h times using different learned projection matrices. Each head independently computes scaled dot-product attention in parallel, defined as:

$$\text{Head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2}$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d_k}$ are learnable projection matrices for each head $1 \leq i \leq h$, and $d_k = d/h$. In decoder blocks, this attention mechanism is typically masked to preserve the autoregressive property required for generation tasks. Specifically, when predicting the token at position t, the model must not attend to tokens at positions $< t$. This is achieved using a causal mask, which is applied to the attention scores before the softmax. The mask sets all future-position scores to , effectively preventing the model from accessing future information during training:

$$\text{MaskedAttention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right)V \tag{3}$$

where $K^\top$ denotes the transpose of K, $M \in R^{s \times s}$ is a lower triangular matrix with zeros in allowed positions and in masked (future) positions. This ensures that each token can only attend to itself and preceding tokens, maintaining consistency with the left-to-right nature of generation. The outputs of all heads are then concatenated and projected back to the original embedding dimension. The final output of the MHSA layer is defined as:

$$\text{MHSA}(Q, K, V) = \text{Concat}(\text{Head}_1, \text{Head}_2, \ldots, \text{Head}_h)W^Z \tag{4}$$

where $W^Z \in \mathbb{R}^{d \times d}$ is a learnable output projection matrix. This output is followed by a residual connection and a Layer Normalization (LN) step. These operations are essential to stabilize training and accelerate convergence. The Layer Normalization operation is typically defined as:

$$\text{LN}(X) = \alpha(X - \mu) + \beta, \quad \alpha = \frac{\lambda}{\sqrt{\sigma^2 + \theta}} \tag{5}$$

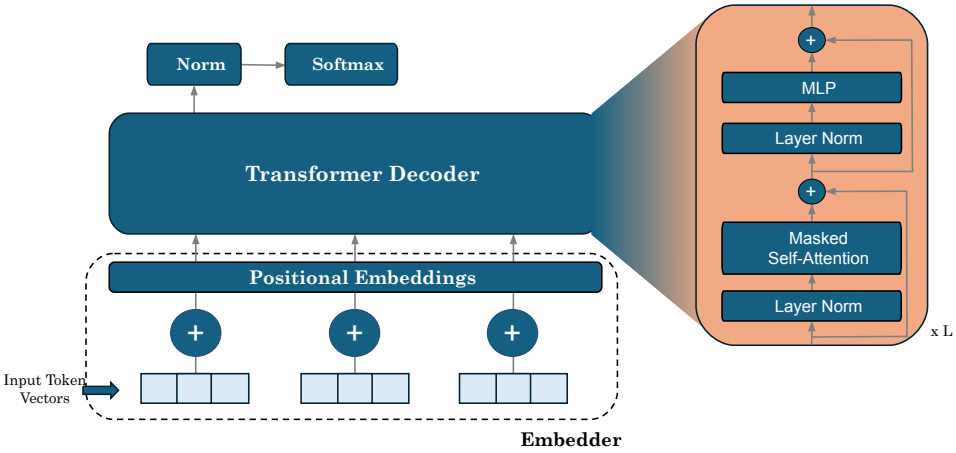

Figure 2: Transformer decoder architecture starts with the input token embeddings combined with positional encodings. The sequence passes through L masked self-attention layers followed by normalization and MLP blocks with residual connections. The output is normalized and projected via softmax to generate token probabilities.

being μ and σ^2 the average of the variance computed over the input X, $\theta = 10^{-5}$ a value added to denominator for numerical stability, λ and β are learnable affine transform parameters. At last, the decoder flows to a position-wise Multi-Layer Perceptron (MLP) structure which is applied to each token with the following formula:

$$\text{MLP}(X) = \text{GELU}(XW_1 + b_1)W_2 + b_2 \tag{6}$$

where $X \in \mathbb{R}^d$ is a token, $W_1 \in \mathbb{R}^{d \times 4d}$, $W_2 \in \mathbb{R}^{4d \times d}$, and $b_1 \in \mathbb{R}^{4d}$, $b_2 \in \mathbb{R}^d$ are bias terms. The matrix W_1 performs dimensionality expansion to $4d$, and W_2 projects the result back to the original dimension d. As in the original Transformer architecture, the Gaussian Error Linear Unit (GELU) is used as the activation function.

Now that we have described each component of the decoder block, we can visualize its structure as a sequence of operations illustrated in Figure 2. The decoder block is composed of two main types of sublayers. The first sublayer applies Multi-Head Self-Attention (MHSA), followed by a residual connection and Layer Normalization. The second sublayer consists of a position-wise Multi-Layer Perceptron (MLP), also followed by a residual connection and Layer Normalization.

A complete decoder is typically composed of several such stacked blocks. Let us consider a single decoder block that takes as input a sequence of token embeddings $X \in \mathbb{R}^{b \times s \times d}$ and transforms it as follows:

$$\begin{aligned} O' &= \text{LN}(\text{MHSA}(X) + X) \\ \text{Decoder}(X) &= \text{LN}(\text{MLP}(O') + O') \end{aligned} \tag{7}$$

where O denotes the Decoder output.

Output Block: After passing through the stacked decoder blocks, the resulting tensor $O \in \mathbb{R}^{b \times s \times d}$ contains contextualized representations for each token position in the input sequence. These representations are then passed to the output block, which transforms them into a probability distribution over the vocabulary at each position.

The output block typically consists of a linear transformation followed by a softmax function. The linear layer projects each d-dimensional token representation into a $|\mathcal{V}|$-dimensional space, where $|\mathcal{V}|$ is the vocabulary size. Let $W^O \in \mathbb{R}^{d \times |\mathcal{V}|}$ and $b \in \mathbb{R}^{|\mathcal{V}|}$ denote the weight matrix and bias vector of the output projection. The raw scores (logits) are computed from the decoder output $O \in \mathbb{R}^{b \times s \times d}$, and passed through a softmax function to yield a probability distribution over the vocabulary:

$$P(w_t \mid w_{<t}) = \text{softmax}(OW^O + b) \quad \in \mathbb{R}^{b \times s \times |\mathcal{V}|} \tag{8}$$

Here, $P(w_t \mid w_{<t})$ denotes the conditional probability of the token w_t given the preceding sequence $w_{<t}$, computed for each position in the sequence. The softmax function is applied along the vocabulary axis to ensure that the output values form valid probability distributions.

During inference, this probability distribution is used to determine the next token in the sequence. The model can either choose the token with the highest probability using `argmax`, or sample from the distribution using strategies such as top-k sampling, nucleus (top-p) sampling, or temperature scaling, depending on the desired level of randomness and diversity in the generated text.

Finally, considering an input X we can summarize the Transformer forward pass as follows:

$$\begin{aligned} \mathcal{T}_0 &= E_m[A] + E_{pos} \\ \mathcal{T}_l &= \text{Decoder}(\mathcal{T}_{l-1}), \quad \text{for } 1 \le l \le L \\ O &= \text{Softmax}(\text{LN}(\mathcal{T}_L)W^O) \end{aligned} \tag{9}$$

where $\mathcal{T}_0 \in \mathbb{R}^{b\times s\times d}$ is the output of the Embedder, E_m and E_{pos} denote the embedding matrix and the positional embeddings, respectively (see Eq. 1), l indicates the index of the current decoder block, and $L$is the total number of decoder blocks. Finally, $\mathcal{T}_L$ represents the output of the last Decoder block, which is normalized and linearly transformed before being passed to the softmax function.

3.2 Serialization

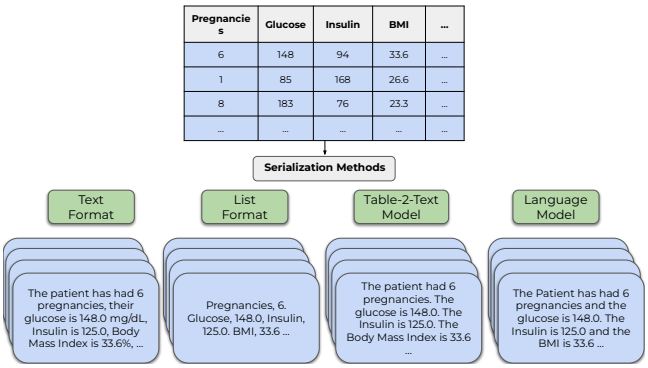

Figure 3: Overview of the main serialization techniques, illustrating how these methods converts table rows into natural language for input to language models.

Problem Formalization. To formalize the problem we are addressing, consider an input table $T \in \mathbb{R}^{l\times c}$, where l represents the number of samples and c denotes the dimensional feature vector. We consider a dataset $D = (c_i, y_i)_{i=1}^{n}$, where each y_i belongs to a set of classes C. The table T includes a feature vector c with column names represented by $F = f_1, f_2, \ldots, f_c$, where each f_i is a natural language string naming each column.

To convert table information into natural language suitable for model processing, we use serialization, transforming table row contents into a format compatible with language models. As Zhao et al. [35] point out, language models' performance is highly sensitive to the specifics of natural language input. In our approach, we use a straightforward description of the classification task for the prompt. We investigate a few serialization formats, each offering different levels of complexity. These methods require low human effort to adapt to new classification tasks.

- **List Format**: A straightforward enumeration of column names f_n and their corresponding feature values, maintaining a fixed column order.
- **Text Format**: A descriptive approach where each feature is expressed as "the *column name* is *value*". Usually built around a predefined script.
- **Table-2-text model**: Utilizes a fine-tuned language model specifically designed for converting tabular data into natural language descriptions.
- **Language Model**: This method is to use a prompt in a language model so it can generate the serialized row. This approach requires configuring the language model's environment to minimize randomness (e.g., setting low temperature in the softmax function) and crafting prompts that explicitly define how each table row should be serialized. This structured output can then be effectively used by downstream classifiers.

After evaluating these serialization techniques, we aim to identify the most suitable method for small language models (SLMs), taking into account both classification metrics and total processing time, which is influenced by model latency. Our objective is to determine whether SLMs can achieve acceptable performance on tabular tasks within resource-constrained environments.

4 METHODOLOGY

In this section, we describe the datasets, evaluation settings, and model settings.

4.1 Datasets and Preprocessing

In order to conduct this research, we selected three open-source tabular datasets for binary classification, each well-recognized within the machine learning community for benchmarking purposes. These datasets possess different feature spaces and varying levels of complexity, allowing us to evaluate the generalization ability and robustness of our models across different data distributions.

The first dataset used was Diabetes[36]. It consists of 768 samples, each with seven features related to medical attributes, including the number of pregnancies, glucose level, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age. During preprocessing, missing values were already filled with zeros, which may induce bias. To mitigate this issue, zero values were replaced with the median of their respective variables, preventing implausible values such as zero for features like Skin Thickness or Blood Pressure. Additionally, the dataset is considered imbalanced, with about 65% of samples labeled as non-diabetic and 35% as diabetic.

The second dataset utilized was the Heart Failure Prediction Dataset[37], which includes 299 samples with twelve features each, aimed at predicting heart failure in patients. A notable issue during preprocessing was the gender imbalance, with males comprising 64% of the samples. Despite this imbalance, the dataset is instrumental in verifying model robustness and performance across genders, adding an essential dimension to the model's evaluation.

The study also incorporated the German Credit Risk Prediction Dataset[38], which consists of 1000 samples with eight features related to customer credit history, occupation, residence type, and loan reason. The dataset labels were categorized into high risk or acceptable risk, maintaining a balanced setup with equal representation of both categories.

These datasets were selected to evaluate our language model-based few-shot classification approach. They represent a diverse range of challenges, such as class imbalance and feature redundancy, making them well-suited for validating the methodological framework proposed in this study. No additional preprocessing was applied, as our goal was to assess the inherent capabilities of the language models. Each row was serialized and used as input to the model, accompanied by an instruction prompt for classification.

Model	Diabetes			Heart Failure			Credit Risk		
	AUC	Accuracy	F1-Score	AUC	Accuracy	F1-Score	AUC	Accuracy	F1-Score
GPT-4o	0.735	0.751	0.657	0.620	0.525	0.544	0.689	0.689	0.663
GPT-4.1	0.727	0.747	0.645	0.619	0.528	0.543	0.717	0.717	0.648
XGBoost [39, 40]	0.80	0.76	-	0.853	0.859	-	0.85	0.77	-
Gemma2-9b	0.711	0.645	0.646	**0.649**	**0.598**	**0.558**	0.648	0.648	0.643
Llama3-8b	0.666	0.611	0.603	0.604	0.518	0.529	**0.672**	**0.673**	**0.663**
Llama3.1-8b	0.504	0.652	0.029	0.551	0.421	0.504	0.591	0.591	0.445
GPT-4o-mini	**0.721**	**0.696**	**0.648**	0.586	0.501	0.514	0.596	0.596	0.516
TabLLM (4-shots) [41]	0.61	-	-	0.76	-	-	**0.69**	-	-
TabPFN (4-shots) [42]	0.61	-	-	**0.84**	-	-	0.58	-	-

Table 1: Performance comparison of different models across three tabular classification datasets: Diabetes, Heart Failure, and Credit Risk. Metrics reported include Area Under the Curve (AUC), Accuracy, and F1-Score. Language Models) are evaluated in few-shot settings, while baselines are included for reference. Values in bold denote the most relevant SLMs results within each dataset.

4.2 Serialization

Our assessment adopted the Text Format as the serialization technique, converting table rows into natural language descriptions using dataset-specific Python scripts, as illustrated in Figure 3.

This approach proved to be the most effective during initial testing. Although the table-to-text models we assessed performed well in some tests, they occasionally ignored actual dataset features and introduced information from the fine-tuned data, suggesting that the model was overfitted and posing a significant issue for our automation process.

4.3 Models' Setting

In order to conduct our research, we selected language models with fewer than 10 billion parameters as the main subjects. The chosen models were Llama3-8B[43], Llama3.1-8B[43], Gemma-2-9B[44], and GPT-4o-mini[45]. We also conducted the same experiment using two large language models to help guide our evaluation: GPT-4o and GPT 4.1.

In this work, we aimed to perform classification using only a few examples. Learning from limited data can be highly effective in many scenarios, especially in resource-constrained environments. Therefore, our prompting strategy employed few-shot learning, comprising five shots—or five examples—embedded in the system prompt (described in Figure 1) to guide the classification process in the language model. All models received the same examples for each dataset, consisting of either three positive and two negative labels, or the reverse, in order to mitigate unwanted biases.

To prepare the models for evaluation, we configured all of them with a temperature setting of 0 to ensure deterministic outputs. Furthermore, we limited the number of output tokens to 10 in order to reduce both inference time and computational resource usage during the assessment, as the expected output was limited to a simple "Yes" or "No" answer.

For model instantiation, we employed two different approaches. Llama and Gemma models were accessed via the Groq API, while the GPT models were evaluated using the OpenAI API. All models were used through their respective hosted endpoints, with no local inference involved.

5 RESULTS

In this section, we present the results obtained using our method with different language models. Table 1 summarizes the evaluation metrics for all models discussed in this work. For comparison, we also include metrics for XGBoost, which is commonly the first choice for tabular tasks, as well as the reported performances of the TabLLM and TabPFN models, which serve the same purpose.

Overall, some Small Language Models achieved results comparable to those of larger models, demonstrating their strong potential in tabular tasks. However, the general performance of all the models analyzed, including the larger ones, demonstrates that further improvements are necessary—especially for handling few-shot classification and for matching the performance of the XGBoost baseline. These challenges seem to stem mainly from a lack of relevant prior knowledge and, in some cases, difficulties in fully understanding the relations of the data provided in the few-shot.

On the Diabetes dataset, Table 1 shows that our models achieved noteworthy accuracy. The top-ranking Small Language Model (SLM) was GPT-4o-mini, reaching an AUC of 72.1% and outperforming all other small language model in every metrics, including TabLLM and TabPFN. Gemma2-9b also achieved a relatively high AUC, although its overall performance metrics were a slightly lower than those of GPT-4o-mini.

On the Heart Failure dataset, however, all models assessed performed substantially worse. The highest AUC was obtained by Gemma2-9b, with 64.9%, which remains far below the TabPFN baseline of 84%. Notably, the Large modelS GPT-4o and GPT-4.1 underperformed compared to the SLM Gemma2-9b, indicating that these models may lack the necessary prior knowledge required for this task, and the limited available data may not have been sufficient to enable the models to capture the underlying feature relationships.

For the German Credit Risk dataset, the best-performing SLM was Llama3-8b, achieving an AUC of 67.2% and leading in other metrics as well. In this instance, the SLM's performance was only slightly behind TabPFN (69%) and GPT-4o-mini (68.9%), demonstrating strong potential for SLMs on this dataset.

This trend is further illustrated in Figure 4, which provides a visual comparison of F1-score, accuracy, and AUC for each model

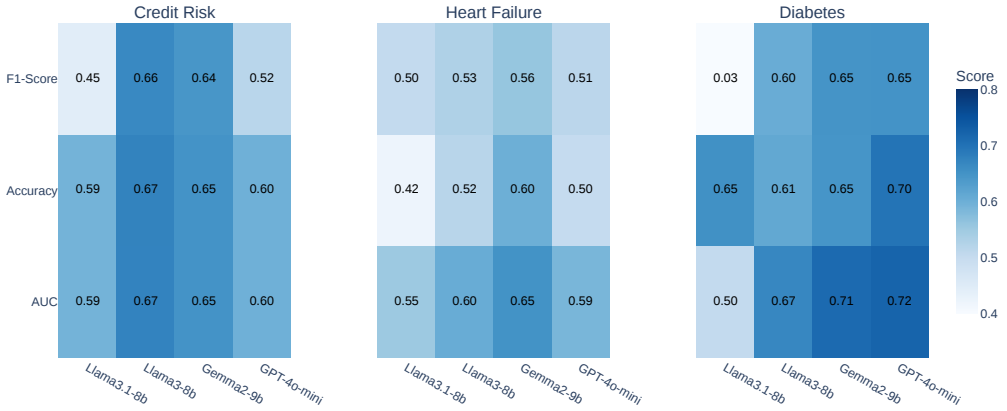

Figure 4: Performance of Small Language Models in each evaluated tasks, measured by F1-Score, Accuracy, and AUC.

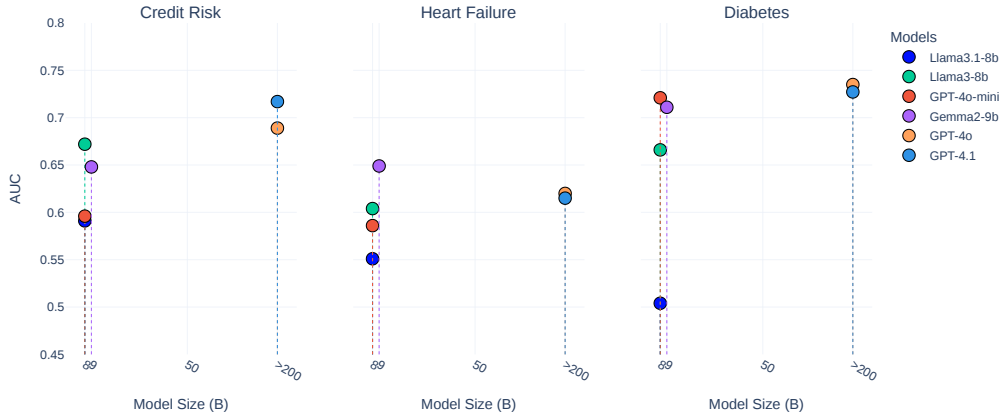

Figure 5: AUC scores versus model size for each dataset. Small Language Models often approach the performance of larger models across evaluated datasets.

across the three datasets. The heatmap confirms that GPT-4o-mini consistently outperforms other small language models on the Diabetes dataset, achieving the highest values across all metrics. In the Heart Failure dataset, Gemma2-9b stands out among SLMs, showing slightly better performance, though scores are lower overall, and all models underperform relative to baselines. For the German Credit Risk dataset, Llama3-8b leads among SLMs, with accuracy and AUC values closely approaching those of GPT-4o-mini and remaining competitive with larger model baselines. These patterns highlight that, for some tabular tasks, select Small Language Models can offer performance that matches or even exceeds larger models, depending on the dataset characteristics.

Overall, our findings reveals that Small Language Models show promise for tabular data tasks, particularly in scenarios where labeled data is scarce. By leveraging the extensive prior knowledge acquired during pre-training, these models can reduce the need for large training datasets. Our approach aims to maximize this prior knowledge through few-shot learning, enabling the efficient development of task-specific models. As a result, competitive performance can be achieved without resorting to traditional methods that require large-scale training on the entire dataset, thereby streamlining the adaptation process for new tabular problems.

Moreover, Figure 5 illustrates the relationship between model parameter size and Area Under the Curve (AUC) for each evaluated approach. Notably, the performance of Small Language Models is very close to that of Large Language Models. On the Credit Risk and Diabetes datasets, the difference in AUC between the best-performing SLM and the LLM was minimal, demonstrating that smaller models can deliver nearly equivalent performance to their larger counterparts. Remarkably, on the Heart Failure dataset, the SLM outperformed the LLM, which shows that in this case, the smaller model was better able to capture meaningful relationships between features and the target variable. These results demonstrate the potential of small language models for tabular learning tasks, especially when labeled data are limited.

We also included an evaluation of XGBoost, as it has long been considered the standard solution for tabular tasks in machine learning. Although XGBoost undeniably achieves outstanding results, its reliance on large amounts of labeled data makes it less practical for automated pipelines or for workflows that evolve over time. In such cases, the requirement for extensive retraining can lead to increased manual effort. In contrast, our approach aims to automate the learning process by leveraging pre-trained language models, enabling effective task adaptation with only a few examples.

6 CONCLUSION

Our study presented a systematic evaluation of various language models, with a special focus on Small Language Models (SLMs), to better understand their capabilities in tabular data classification tasks using few-shot learning. By leveraging prior knowledge learned during pre-training, we aimed to assess whether SLMs could deliver competitive results on classic tabular datasets without the need for extensive task-specific training.

The experimental results highlight that SLMs—especially models like GPT-4o-mini, Llama3-8b and Gemma2-9b—can often approach, and occasionally match, the performance of larger language models across diverse benchmarks such as Diabetes, Heart Failure, and German Credit Risk datasets. Notably, SLMs achieved robust performance while requiring significantly fewer computational resources and less labeled data. This demonstrates the potential of small-scale models for resource-constrained environments and iterative machine learning workflows.

Our study also found that the effectiveness of language models in tabular data is influenced both by model scale and by the intrinsic complexity of the dataset. While the largest models set the upper bound in performance, well-chosen SLMs were able to achieve near-parity in several cases—underscoring the value of leveraging model efficiency and prior knowledge. These findings supports the idea that prompt engineering and serialization strategies serve as a key factor in enabling strong generalization from just a few examples.

In summary, our work demonstrates that, with appropriate input serialization and few-shot learning, can streamline the deployment of machine learning solutions without the burdens of large-scale data collection or extensive retraining. While further research and development are needed for language models to consistently reach the strong benchmark results of traditional methods such as XGBoost, leveraging language models opens the door to much more dynamic and flexible machine learning pipelines—potentially reducing, and in some cases eliminating, the need for fully labeled datasets and exhaustive retraining. These insights pave the way for future research in automated machine learning and low-shot adaptation of language models to diverse structured data domains. As the field progresses, we anticipate that continued innovation in model efficiency and prompt strategies will further unlock the strengths of language models, even in traditional domains like tabular data.

REFERENCES

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, et al. Attention is all you need, 2023.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. Language models are few-shot learners, 2020.

[5] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, et al. Multitask prompted training enables zero-shot task generalization, 2022.

[6] Boseop Kim, HyoungSeok Kim, Sang-Woo Lee, Gichang Lee, Donghyun Kwak, Dong Hyeon Jeon, Sunghyun Park, Sungju Kim, et al. What changes can large-scale language models bring? intensive study on hyperclova: Billions-scale korean generative pretrained transformers, 2021.

[7] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2021.

[8] Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):8076–8084, Jul. 2019.

[9] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks, 2015.

[10] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. Finqa: A dataset of numerical reasoning over financial data, 2022.

[11] Griffin Adams, Alexander Fabbri, Faisal Ladhak, Eric Lehman, and Noémie Elhadad. From sparse to dense: Gpt-4 summarization with chain of density prompting, 2023.

[12] Toufique Ahmed and Premkumar Devanbu. Few-shot training llms for project-specific code-summarization, 2022.

[13] Avinesh P. V. S., Benjamin Hättasch, Orkan Özyurt, Carsten Binnig, and Christian M. Meyer. Sherlock: a system for interactive summarization of large text collections. *Proc. VLDB Endow.*, 11(12):1902–1905, August 2018.

[14] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[15] Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. Crossfit: A few-shot learning challenge for cross-task generalization in nlp, 2021.

[16] Vanessa Câmara, Rayol Mendonca-Neto, André Silva, and Luiz Cordovil-Jr. Db-vinci – towards the usage of gpt engine for processing sql queries. In *Proceedings of the 29th Brazilian Symposium on Multimedia and the Web*, WebMedia '23, page 91–95, New York, NY, USA, 2023. Association for Computing Machinery.

[17] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm, 2021.

[18] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models, 2021.

[19] Maria Sahakyan, Zeyar Aung, and Talal Rahwan. Explainable artificial intelligence for tabular data: A survey. *IEEE Access*, 9:135392–135422, 2021.

[20] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. TaBERT: Pretraining for joint understanding of textual and tabular data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online, July 2020. Association for Computational Linguistics.

[21] Sercan O. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning, 2020.

[22] Sungwon Han, Jinsung Yoon, Sercan O Arik, and Tomas Pfister. Large language models can automatically engineer features for few-shot tabular learning, 2024.

[23] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[24] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data?, 2022.

[25] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 35(6):7499–7519, 2024.

[26] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[27] Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. Small language models: Survey, measurements, and insights, 2025.

[28] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1467–1475, 2015.

[29] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data, 2019.

[30] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[31] Asaf Harari and Gilad Katz. Few-shot tabular data enrichment using fine-tuned transformer architectures. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1577–1591, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[32] Kimberly Villalobos Carballo, Liangyuan Na, Yu Ma, Léonard Boussioux, Cynthia Zeng, Luis R. Soenksen, and Dimitris Bertsimas. Tabtext: A flexible and contextual approach to tabular data representation, 2023.

[33] Tuan Dinh, Yuchen Zeng, Ruisu Zhang, Ziqian Lin, Michael Gira, Shashank Rajput, Jy yong Sohn, Dimitris Papailiopoulos, and Kangwook Lee. Lift: Language-interfaced fine-tuning for non-language machine learning tasks, 2022.

[34] Mario Haddad-Neto, André Silva, Rayol Mendonca-Neto, and Luiz Cordovil. Exploring the impact of zero-cost proxies for hybrid vision transformers. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2024.

[35] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models, 2021.

[36] Michael Kahn. Diabetes. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5T59G.

[37] Davide Chicco and Giuseppe Jurman. Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC Medical Informatics and Decision Making*, 20(1):16, 2020.

[38] Hans Hofmann. Statlog (German Credit Data). UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

[39] Jonindo Pasaribu, Novanto Yudistira, and Wayan Firdaus Mahmudy. Tabular data classification and regression: Xgboost or deep learning with retrieval-augmented generation. *IEEE Access*, 12:191719–191732, 2024.

[40] Srichand Doki, Siddhartha Devella, Sumanth Tallam, Sai Sujeeth Reddy Gangannagari, P. Sampathkrishna Reddy, and G. Pradeep Reddy. Heart disease prediction using xgboost. In *2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT)*, pages 1317–1320, 2022.

[41] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models, 2023.

[42] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second, 2023.

[43] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, et al. The llama 3 herd of models, 2024.

[44] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, et al. Gemma 2: Improving open language models at a practical size, 2024.

[45] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, et al. Gpt-4o system card, 2024.