PRIORBRIDGE: INTEGRATING LANGUAGE PRIORS INTO PRIOR-FITTED NETWORKS VIA RULE-GUIDED PREDICTION CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Prior-fitted networks (PFNs) have become strong predictors for tabular data. They use labeled examples as context to infer target values from feature dependencies. However, their standard inference interface cannot directly accept user-specified knowledge expressed in natural language. This may limit their use in real-world scenarios, especially when the context data does not fully reflect the target distribution or users prioritize compliance with predefined rules over predictive quality. To bridge this gap, we propose **PriorBridge**, a framework that integrates language priors into PFN predictions through tree-based rules refined by a large language model (LLM). First, we make the PFN's prediction behavior readable as explicit rules. Given a labeled training set $(X_{\text{train}}, Y_{\text{train}})$ as context, we query the PFN on target inputs X_{test} and fit a small tree-based proxy to the resulting predictions $\hat{Y}_{\text{test}}$. The proxy's decision paths can then be expressed as natural-language rules. Second, we convert the language prior into input-dependent output ranges. The LLM combines the proxy rules with the natural-language constraint and assigns an output range to each leaf. Finally, we apply the refined ranges to the original PFN predictions. Predictions outside the corresponding ranges are projected to their nearest boundaries. Experiments show that PriorBridge effectively integrates language priors into PFN predictions. It achieves high prior compliance across different PFN backbones. When a prior captures patterns missing from the context data, our method can also use it to improve predictive quality.

1 INTRODUCTION

Tabular prediction is important in many real-world applications. Given a row of structured features, the goal is to estimate its corresponding target value. Because they can capture relationships between features with data-driven rules, tree-based ensembles are widely used for this task. In recent years, prior-fitted networks (PFNs) have provided a different approach to tabular prediction. PFNs are pre-trained on synthetic data and use labeled training examples as context at inference time (Hollmann et al., 2022; 2025). Recent PFN-based models achieve strong performance on classification and regression benchmarks (Grinsztajn et al., 2026; Zhang et al., 2026), and can even surpass traditional tree-based models on some tasks.

At the same time, different tables involve different forms of domain knowledge. Practitioners may know relationships between the observed variables and the expected outcome in advance. For example, it is common knowledge that stronger winds increase air flow and help disperse particulate matter. Therefore, wind speed is negatively related to particulate levels when other factors are controlled. Such knowledge can complement the patterns learned by the model and improve its predictions, especially when the relevant priors are not fully represented in the context data. Moreover, in some real-world settings, satisfying prior knowledge may be the main goal. Users may require predictions to satisfy prior relationships, even when doing so does not directly maximize a standard predictive metric. Therefore, incorporating such priors into predictions for specific datasets can benefit tabular prediction.

However, current PFNs do not provide a direct way to use these language priors. Given a labeled training set $(X_{\text{train}}, Y_{\text{train}})$, a PFN uses it as context to predict targets for test inputs X_{test}. The stan-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

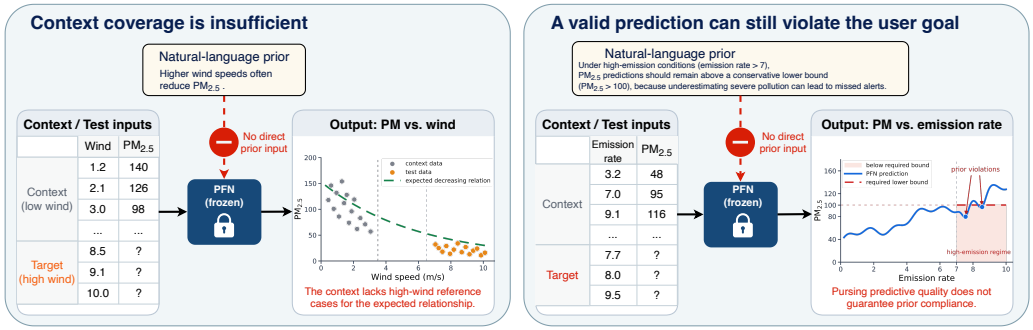

Figure 1: Limitation of standard PFNs in using language priors. PFN cannot receive user-specified constraints, which can lead to failures when (a) context lacks reference cases for test data or (b) compliance with human-specified constraints is prioritized over reducing overall prediction error.

dard PFN interface does not accept a natural-language constraint as part of this prediction process. Its end-to-end computation also lacks an explicit rule module that can be edited at inference time. This limitation is especially important when the context does not fully represent the relationship expected by the user. For example, if $(X_{\text{train}}, Y_{\text{train}})$ contains mostly low-wind observations, the PFN may produce inaccurate prediction $\hat{Y}_{\text{test}}$ for high-wind test samples due to lack of reference cases in the context. In addition, users may prioritize satisfying a constraint over retaining the PFN's raw predictions. For example, as shown in the right panel of Figure 1, a practitioner may require predictions at high emission rates to remain above a predefined bound. Avoiding missed alerts in extreme cases may be more important than minimizing overall prediction error. In summary, these limitations highlight the need for an effective way to connect language constraints to PFN predictions.

To address this challenge, we propose PriorBridge, a framework for integrating language priors into PFN predictions through tree-based rules refined by a large language model (LLM). Our method consists of three stages. First, for a given dataset, we train a proxy tree-based model to imitate the predicting pattern of frozen PFN model. We query the PFN with $(X_{\text{train}}, Y_{\text{train}})$ as context to obtain its predictions $\hat{Y}_{\text{test}}$ on X_{test}. Then we train a small tree-based proxy model to fit the pairs $(X_{\text{test}}, \hat{Y}_{\text{test}})$. The proxy's decision paths summarize the local prediction behavior of the PFN as explicit rules. Second, we use an LLM to interpret the predefined language priors and refine proxy's rules with value constraints. Since proxy model's rules from the first stage can be expressed in natural language, an LLM can read them. The LLM receives these rules together with the user's prior constraint and refines them into an allowed output range for each relevant input region. Third, we apply the corresponding ranges to the original PFN predictions during final inference. A prediction outside its allowed range is projected to the nearest boundary, while an in-range prediction is left unchanged. By deploying the above pipeline, our method combines the PFN's predictions with the user-provided priors. To verify PriorBridge's effectiveness, we conduct experiments on various tasks. Our method achieves an average prior compliance above 96% across all evaluated settings. For some specific tasks, it improves predictions using priors that are not indicated by the context data, which further confirms the significance of utilizing available priors for tabular prediction tasks. All these results show that PriorBridge can effectively integrate prior-based control into PFN predictions and address the challenge of modality misalignment.

Our contributions are summarized as follows:

- We formulate the task of integrating priors for PFNs as enforcing user-specified natural-language constraints on their predictions.
- We propose PriorBridge, which connects these constraints to PFN outputs through a tree-based proxy and LLM-refined rules. Our method provides an effective and flexible way to address the challenge that a raw PFN model cannot utilize priors.
- We evaluate the method under both in-distribution and out-of-distribution settings. The results show high prior compliance and confirm the effectiveness of our method.

2 Related Work

Predictive models for tabular data. Tabular data are widely used in real-world applications. A table is typically organized with columns representing features and rows representing samples. Prediction tasks on tabular data therefore follow the standard supervised learning setup. Since tabular data combine structured features with substantial heterogeneity, tree-based models have been highly effective for these tasks. Random forests reduce variance through bagging, while boosted-tree systems such as XGBoost and CatBoost build accurate task-specific predictors (Breiman, 2001; Chen & Guestrin, 2016; Prokhorenkova et al., 2018). With the rapid development of deep learning, researchers have also explored neural networks that learn continuous representations for tabular data (Zhang et al., 2024; Tong et al., 2025). However, broad evaluations continue to show that the advantage over tuned tree ensembles depends on the dataset and training budget (Gorishniy et al., 2021; Grinsztajn et al., 2022). Tree ensembles still demonstrate strong performances and are widely seen as the best choice for deployment.

Prior-fitted networks and tabular foundation models. In recent years, motivated by in-context learning (Dong et al., 2024), prior-fitted networks (PFNs) have provided a new paradigm for tabular prediction (Hollmann et al., 2022; 2025). Their strong predictive performance, which can surpass that of tree-based models, has made PFNs an important architecture for tabular foundation models. PFNs are first pretrained on synthetic datasets sampled from causal graphs, allowing them to encode statistical priors during pretraining. During inference, a PFN uses labeled data as context and predicts the targets of new samples. Several models in this family, including TabPFN (Hollmann et al., 2022; Grinsztajn et al., 2025; 2026), LimiX (Zhang et al., 2025; Wang et al., 2026; Zhang et al., 2026), TabICL (Qu et al., 2025; 2026), and Xiaomi-TabLDM (Team et al., 2026), have demonstrated strong predictive performance.

Language for tabular prediction. Benefiting from large-scale pretraining and post-training (Jaech et al., 2024; Guo et al., 2025; Shao et al., 2024), large language models (LLMs) have rapidly advanced in their ability to respond to user queries. Because language provides a natural medium for expressing information, researchers have also explored using LLMs as direct predictors for tabular data (Hegselmann et al., 2023; Gardner et al., 2024). For example, TABULA (Gardner et al., 2024) finetunes a Llama-based model (Meta, 2024) to answer questions about tabular data. This approach differs from conventional tabular prediction because numerical values are treated as tokens, alongside textual descriptions. However, it requires suitable tabular training data and often depends on large-scale training. Other recent works use LLMs as tools rather than training them directly for prediction (Ye et al., 2026; Yang et al., 2026). These methods treat the LLM as a carrier of knowledge and focus on extracting information that can support downstream tasks.

3 Preliminaries

3.1 Background: Prior-Fitted Networks

Prior-fitted networks (PFNs) adopt in-context learning as a general prediction procedure. During pretraining, they are trained on synthetic tabular data generated from a large collection of causal graphs. Part of each table is provided as labeled context, while the remaining samples serve as prediction targets. Recent PFNs typically use a transformer architecture to model complex dependencies among features. They acquire the ability to generalize to real-world data by learning diverse feature interactions from these simulated tasks.

During inference on a new tabular task, a labeled training set

$$\mathcal{D}_{\mathrm{ctx}} = (X_{\mathrm{train}}, Y_{\mathrm{train}}) = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n} \tag{1}$$

is used as context, where $\boldsymbol{x}_i \in \mathbb{R}^d$. $X_{\mathrm{test}} = \{\boldsymbol{x}_j\}_{j=1}^{m}$ denotes the test inputs whose targets are to be predicted. A pretrained PFN with fixed parameters θ performs in-context learning through

$$p_\theta\left(y_j \mid \boldsymbol{x}_j, \mathcal{D}_{\mathrm{ctx}}\right). \tag{2}$$

The model adapts to the new table through $\mathcal{D}_{\mathrm{ctx}}$ and produces predictions for X_{test} without parameter updates. We use $\hat{y}_j$ for the direct PFN output. For regression, these outputs are scalar estimates of a continuous target. For classification, they denote the predicted probability of the class specified by the prior, from which the final class prediction is obtained.

3.2 Motivation: Integrating Domain Priors into Tabular Prediction

In tabular data, each column represents a feature with a domain-specific meaning. These features are closely related to the realistic scenarios in which the table is used. Therefore, prior knowledge about them is often available through expert experience. Domain experts may know that certain observations should have bounded responses or favor a particular class. For example, it is a common sense that stronger winds increase airflow and help disperse particulate matter. Wind speed is thus often negatively related to particulate levels when other factors are controlled. These natural-language priors are useful in two common situations. First, some test samples $(X_{\text{test}}, Y_{\text{test}})$ may exhibit prediction patterns that are not represented in the training context $\mathcal{D}_{\text{ctx}}$. Therefore, incorporating prior knowledge during inference may improve predictions under an out-of-distribution shift. Second, practitioners may require predictions to satisfy predefined relationships rather than simply improve predictive quality. In this case, fulfilling the prior constraints is the main objective. It may conflict with the original training objective of the model and leads to unexpected estimation.

However, as introduced in Section 3.1, PFN's predictions are mainly determined by the relationship learned from $\mathcal{D}_{\text{ctx}}$ and applied to X_{test}. Current PFNs lack an explicit interface for incorporating natural-language priors into this process. This limitation constrains their deployment for customized application scenarios. It motivate us to design a pipeline that enables PFNs to apply such priors while preserving their original prediction ability.

3.3 Task Definition

In this section, we formulate the task mathematically. We follow the notation introduced in Section 3.1. The user provides a collection of natural-language constraints $\mathcal{P} = \{\pi_k\}_{k=1}^{K}$, where each constraint stands for a prior introduced above. Conceptually, each π_k specifies an input condition $a_k(\boldsymbol{x})$ and a corresponding constraint $\mathcal{C}_k(\boldsymbol{x}, y)$ on the relevant output. The constraint may be given explicitly as a numerical range for the output. It may also be given implicitly as a relationship between the output and an input feature, *i.e.*, requiring the output to increase or decrease with that feature. Such a relationship does not directly specify a numerical range, but instead describes how the output should respond to the input. A prediction satisfies constraint π_k when

$$a_k(\boldsymbol{x}) = 0 \quad \text{or} \quad \mathcal{C}_k\left(\boldsymbol{x}, \hat{y}(\boldsymbol{x})\right) = 1. \tag{3}$$

The user specifies each π_k in natural language, so a_k and $\mathcal{C}_k$ are not directly available as executable functions. In addition, $\mathcal{C}_k$ is not allowed to use or reveal the real test labels.

When several language priors apply to the same input, their joint constraint is the conjunction

$$\mathcal{C}_{\mathcal{P}}(\boldsymbol{x}, u) = \bigwedge_{k: a_k(\boldsymbol{x})=1} \mathcal{C}_k(\boldsymbol{x}, u). \tag{4}$$

For a non-empty set of outputs satisfying $\mathcal{C}_{\mathcal{P}}$, we denote the corrected output by $\hat{y}^{\text{proj}}(\boldsymbol{x})$. It preserves the direct PFN output whenever it is valid and otherwise makes the smallest change:

$$\hat{y}^{\text{proj}}(\boldsymbol{x}) = \underset{u:\, \mathcal{C}_{\mathcal{P}}(\boldsymbol{x}, u)=1}{\arg\min} \left|u - \hat{y}(\boldsymbol{x})\right|. \tag{5}$$

The test labels $Y_{\text{test}} = \{y_j\}_{j=1}^{m}$ are unavailable during rule construction and prediction. They are used only for evaluation after the predictions have been produced. The task is therefore to turn the language priors in $\mathcal{P}$ into executable input-dependent constraints and apply Eq. 5 while keeping model parameters θ unchanged.

4 Methodology

In this section, we introduce **PriorBridge**, a framework that integrates language priors into PFN predictions through an explicit rule representation. Our method has three stages, which is illustrated in Figure 2. First, given $\mathcal{D}_{\text{ctx}}$, we query the PFN on the unlabeled inputs X_{test} and fit a small tree-based proxy to its direct predictions. The proxy's decision paths provide a readable summary of the PFN's prediction behavior. Second, a large language model (LLM) receives these proxy rules together with the language priors and refines them into input-dependent constraints. Finally, we use the refined constraints to correct the original PFN output when it conflicts with the specified priors. Model parameters remain fixed throughout this process. Section 4.1, 4.2, and 4.3 separately describe these stages in detail.

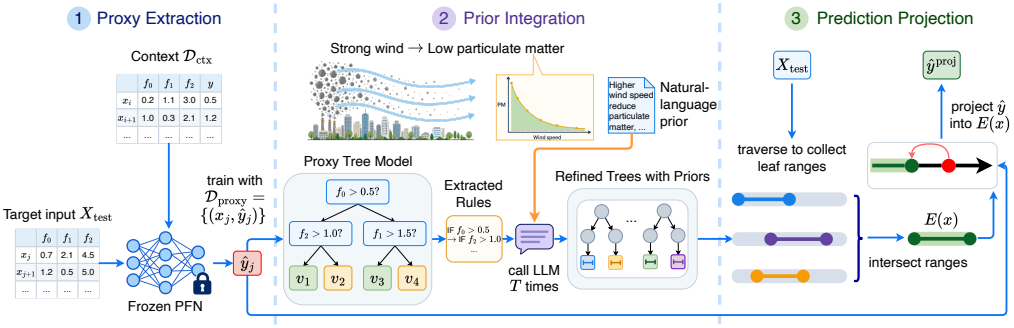

Figure 2: PriorBridge's running pipeline. Our method consists of three stages: proxy extraction (Section 4.1), prior integration (Section 4.2), and prediction projection (Section 4.3).

4.1 Training a Proxy to Extract Rules

The first challenge in applying prior rules to PFNs is the mismatch between their input modalities. PFN-based models process numerical tabular values, and they do not provide an interface for receiving language input. We therefore seek a proxy that can carry the essential information in the PFN's predictions into a language-readable form. Previous works Breiman (2001); Chen & Guestrin (2016); Prokhorenkova et al. (2018); Ye et al. (2026) have shown that tree-based models are a natural option for this purpose. Tree ensembles make predictions through multi-step decision paths, which can be expressed as hierarchical judgments in natural language. If a tree-based model can reproduce the PFN's prediction behavior, its extracted rules can serve as a proxy for the PFN. The problem is therefore reduced to training an accurate tree-based proxy for the PFN.

Given $\mathcal{D}_{\text{ctx}}$, we query the PFN on each input $\boldsymbol{x}_j \in X_{\text{test}}$ and obtain its direct output $\hat{y}_j$. We then construct the proxy dataset

$$\mathcal{D}_{\text{proxy}} = \{(\boldsymbol{x}_j, \hat{y}_j)\}_{j=1}^{m}. \tag{6}$$

The proxy h is a small tree ensemble trained to fit these pairs. For continuous PFN outputs, we train h as a tree-based regressor with a squared-error objective. For classification outputs, we train h as a tree-based classifier using the PFN's hard predictions as targets and impurity-reducing splits. Because its targets are generated by the PFN, h approximates the PFN's prediction behavior rather than groundtruth labels.

Each tree divides the target feature space by repeatedly applying threshold conditions to individual features. A root-to-leaf path can be written as

$$R = \{\boldsymbol{x} : q_1(\boldsymbol{x}) \wedge \cdots \wedge q_L(\boldsymbol{x})\}, \tag{7}$$

where each q_l is a split condition on one feature, such as $x_r \leq \tau$. We export these paths as text together with the corresponding proxy outputs. They form a compact, dataset-specific summary of PFN behavior expressed as language-readable rules.

This stage serves two purposes. First, it changes a dense neural prediction function into conditions over named tabular features. Second, it preserves a behavioral reference for the PFN when the LLM incorporates the prior. These proxy rules provide the representation used to incorporate prior constraints in the next section.

4.2 Integrating Prior Constraints with an LLM

With the language-readable proxy rules extracted in Section 4.1, we next use an LLM to incorporate the user's prior constraints. Benefiting from large-scale pretraining and post-training (Jaech et al., 2024; Guo et al., 2025), current LLMs can interpret and organize complex natural-language descriptions. The LLM receives three types of information: feature and target descriptions, the exported proxy paths, and the user's prior constraints $\mathcal{P}$. The prompt asks it to preserve useful proxy splits, revise paths when a constraint requires a different region, and attach an output interval to each leaf.

We query the LLM T times to reduce dependence on a single generated rule set. The t-th query returns a rule tree g_t. Its internal nodes specify a feature, a comparison operator, and a threshold. Its

leaves specify an interval:

$$I_t(\boldsymbol{x}) = [\ell_t(\boldsymbol{x}), u_t(\boldsymbol{x})], \tag{8}$$

where $I_t(\boldsymbol{x})$ is selected by traversing g_t with $\boldsymbol{x}$. For regression, the interval is expressed in the original target units. For classification, it bounds the probability of a designated class $c^\star$ and lies in $[0, 1]$. An unconstrained side of an interval is represented by $-\infty$ or $+\infty$. Thus, each $I_t(\boldsymbol{x})$ provides a concrete, executable form of the output constraint for the region reached by $\boldsymbol{x}$.

In this stage, we translate the natural-language priors and the proxy's behavioral reference into editable, region-specific output restrictions. The LLM does not replace the PFN or produce a new prediction. Instead, it associates each input region with an explicit constraint that can be checked against the PFN output. Keeping both the region and the interval explicit makes the resulting correction guidance traceable to a decision path and a leaf interval.

4.3 Correcting PFN Predictions with Refined Rules

With the refined rules, we finally apply them to each direct PFN prediction to fulfill prior constraints. For an input $\boldsymbol{x}$, we traverse the T rule trees and collect the interval $I_t(\boldsymbol{x})$ reached in each tree. Their intersection defines the joint admissible range:

$$\mathcal{I}(\boldsymbol{x}) = \bigcap_{t=1}^{T} I_t(\boldsymbol{x}) = [L(\boldsymbol{x}), U(\boldsymbol{x})], \qquad L(\boldsymbol{x}) = \max_t \ell_t(\boldsymbol{x}), \quad U(\boldsymbol{x}) = \min_t u_t(\boldsymbol{x}). \tag{9}$$

This range combines the output constraints provided by the individual rule trees. When $\mathcal{I}(\boldsymbol{x})$ is non-empty, the corrected PFN output is the projection of the direct output onto this range:

$$\hat{y}^{\mathrm{proj}}(\boldsymbol{x}) = \Pi_{\mathcal{I}(\boldsymbol{x})}(\hat{y}(\boldsymbol{x})) = \operatorname*{arg\,min}_{u \in \mathcal{I}(\boldsymbol{x})} |u - \hat{y}(\boldsymbol{x})|. \tag{10}$$

This operation applies to the task-relevant scalar output, including a regression value or a designated class probability. If the direct PFN output already satisfies the joint constraints, it remains unchanged. Otherwise, it moves to the nearest admissible boundary.

Our correction process is applied to the original PFN output rather than to the proxy output. When the proxy accurately summarizes PFN behavior, the joint range is aligned with the regions in which the PFN operates, so valid PFN predictions are preserved. Predictions that conflict with the translated priors are changed only as much as needed to satisfy the joint constraint. Therefore, our method could combine the predictive behavior learned by the PFN with explicit prior-based control.

5 Experiments

5.1 Experimental Setting

Datasets and splits. We evaluate 21 tabular datasets with specific semantics from OpenML (Bischl et al., 2017) and UCI Machine Learning Repository (Asuncion et al., 2007), covering both classification and regression tasks. Eleven datasets use standard in-distribution (IID) splits, while ten use explicit out-of-distribution (OOD) splits. The OOD shifts are defined by task-relevant features, time, or categories. For example, the bike-sharing task uses unseen time periods to create a shift between the context and test data. The evaluation includes 55 natural-language constraints: 29 in the IID group and 26 in the OOD group. All features in the evaluated datasets have a clear task-specific meaning. Each prior is defined over selected features and the target variable. We construct the priors according to the task definition in Section 3.3. Some specify explicit value ranges, whereas others describe only monotonic relationships between features and the target variable.

PFN backbones. We use two PFN-based models as backbones: TabPFN-v2 (Grinsztajn et al., 2025) and LimiX-2M (Zhang et al., 2025). Within each backbone, the base and corrected predictions use identical data and preprocessing. To isolate the effect of our method, we use the standard implementations without additional techniques like retrieval.

Table 1: Results on standard IID datasets. *#P* denotes the number of applied priors on each dataset. *Comp.* denotes dataset-level prior compliance, which is our main evaluation protocol.

		LimiX-2M			TabPFN-v2		
Dataset	#P	Base	Ours	Comp. (%)	Base	Ours	Comp. (%)
Regression (MAE ↓)							
Abalone	3	**1.40**	1.42	53.9→100	**1.38**	1.41	59.9→100
Beijing PM2.5	3	**48.6**	49.1	92.1→99.6	**49.4**	80.8	91.7→99.9
California Housing	3	**26523**	26619	86.6→100	**23649**	46195	88.5→100
Wine Quality Red	3	**0.370**	0.410	45.6→88.9	0.474	**0.471**	28.0→88.9
Classification (accuracy ↑)							
Australian Credit	2	**0.885**	**0.885**	65.5→100	**0.885**	0.817	71.4→100
Diabetes	2	0.750	**0.767**	92.6→100	0.733	**0.750**	92.6→100
Electricity	3	0.672	**0.790**	45.5→100	0.702	**0.790**	52.1→100
Gas Drift	3	0.897	**0.924**	3.2→99.8	0.925	**0.947**	11.2→99.8
KC2	2	**0.873**	0.861	63.3→100	**0.861**	**0.861**	78.9→100
NSL-KDD	3	0.782	**0.929**	26.7→100	0.764	**0.885**	23.1→100
QSAR Biodegradation	2	0.862	**0.881**	60.9→77.4	0.855	**0.874**	58.5→75.0
Average compliance				57.8→**96.9**			59.6→**96.7**

Table 2: Results on datasets with distribution shifts. The protocol and notation follow Table 1.

		LimiX-2M			TabPFN-v2		
Dataset	#P	Base	Ours	Comp. (%)	Base	Ours	Comp. (%)
Regression (MAE ↓)							
Airfoil	3	5.87	**4.70**	60.5→100	5.79	**4.74**	60.9→100
CCPP	3	11.2	**7.56**	37.2→100	4.49	**3.95**	95.4→100
Concrete	3	14.7	**14.5**	96.0→100	8.21	**8.19**	98.5→100
Bike Sharing	3	128	**121**	92.0→95.9	**89.6**	114	90.0→94.0
Gas Turbine NOx	3	**6.39**	7.32	100→100	14.5	**12.5**	45.2→100
Seoul Bike	3	400	**367**	99.7→99.8	**339**	396	99.5→99.8
Superconductivity	3	34.1	**33.8**	87.5→100	23.9	**23.8**	98.2→100
Classification (accuracy ↑)							
Dry Bean	1	0.610	**0.840**	0.0→100	0.704	**0.705**	99.3→100
CICIDS2017	3	0.874	**0.905**	15.7→100	0.873	**0.906**	14.8→100
Gesture Phase	1	0.433	**0.451**	2.2→100	0.425	**0.434**	8.9→100
Average compliance				59.1→**99.6**			71.1→**99.4**

Proxy training and rule generation. We run our method separately for each PFN. For each backbone, we query the PFN on the unlabeled target inputs and fit a backbone-specific proxy forest to its predictions. Following (Ye et al., 2026), we use proxy forests with five trees, a maximum depth of 10, and at most 50 leaves per tree. For larger datasets, we increase the maximum number of leaves to 200 to improve proxy fidelity. We then provide the extracted paths, the task description, and the natural-language prior constraints to the LLM. An example is provided in the Appendix. We adopt Gemini-2.5-Flash (Comanici et al., 2025) as the LLM to refine proxy rules. T is set to 5, so five refined rule trees are generated for each dataset. Therefore, each backbone is evaluated with its own distilled proxy and refined rule set. We also ensure that target labels and test-set performance statistics are excluded from the prompts. They are only used for measuring final performances.

Evaluation metrics. We evaluate two aspects of our method: the degree to which the refined predictions satisfy the prior constraints and their predictive quality. Each constraint specifies a subset of target inputs and a required condition on their predictions. Its compliance is the fraction of predictions in that subset that satisfy the condition. Dataset-level compliance is the unweighted

Table 3: Prior-source ablation with model-specific proxies. We fix the tree structures and sample routing, and vary only the leaf intervals.

Dataset	PFN	Metric	Base	Random	Train-Quantile	Ours
Airfoil	LimiX	MAE ↓	5.87	5.85	4.82	**4.70**
Airfoil	TabPFN	MAE ↓	5.79	5.67	4.83	**4.74**
NSL-KDD	LimiX	Acc. ↑	0.782	0.609	0.756	**0.929**
NSL-KDD	TabPFN	Acc. ↑	0.764	0.546	0.666	**0.885**

mean over its constraints, and group-level results are averaged across datasets with equal weight. Compliance is the primary metric because the method is designed to enforce specified language priors. Predictive quality is reported separately because a constraint can improve or harm accuracy depending on its agreement with the unobserved target labels. Specifically, we use mean absolute error (MAE) for regression and accuracy for classification.

5.2 Main Results

IID results. Table 1 reports the results on the IID datasets. Our method consistently increases prior compliance for both PFN backbones. Most datasets reach approximately complete compliance. However, predictive quality does not improve uniformly. Our method is designed to satisfy the given prior. It does not directly optimize predictive accuracy. When the prior does not align with the dataset statistics, the correction in Section 4.3 may move predictions away from the target values. Predictive quality may therefore decrease even as compliance improves.

OOD results. We further evaluate the method under explicit distribution shifts. In these settings, the prior constraints describe expected test-time patterns. They may provide information that is not captured by the context data. Table 2 shows that average compliance increases from 59.1% to 99.6% for LimiX. For TabPFN, it increases from 71.1% to 99.4%. Predictive quality also improves in most cases because the provided priors are now generally consistent with the OOD patterns in these experiments. Experiment on Dry Bean illustrates how this benefit depends on the original PFN predictions. TabPFN already achieves 99.3% compliance before correction. It therefore leaves little room for PriorBridge to improve either compliance or accuracy. In contrast, LimiX initially does not fulfill with the prior at all. After correction, its prior compliance reaches 100%. The accuracy also increases from 61.0% to 84.0%. This contrast indicates that the OOD prior is useful for prediction, but its effect depends on whether the backbone already follows it.

Overall, the IID and OOD results show that our method provides an effective interface for integrating language priors into PFN predictions. It consistently improves prior compliance across the two backbones. In addition, when a prior is consistent with the prediction objective, our correction can even improve predictive quality.

5.3 Prior-Source Ablation

Our main results show that range projection enforces the generated rules and thereby enables prior-based control. However, they do not quantify whether language priors provide useful information for determining the ranges. To isolate this effect, we keep the model-specific proxy structure and sample routing fixed for each PFN while changing only the leaf intervals. We then compare our method with two variants that use different sources to determine these intervals. *Random* shuffles the association between input regions and intervals over multiple seeds. *Train-Quantile* derives leaf ranges from labeled context examples without an LLM. Table 3 reports the results for two representative tasks with both PFNs. The language-derived intervals perform best in all four settings. Random intervals can be even worse than the uncorrected model. Train-quantile intervals provide a stronger alternative, but remain less effective than our method because they introduce no new prior information. These results indicate that the benefit does not come from clipping alone. It depends on assigning suitable ranges to the relevant input regions according to the given prior constraints.

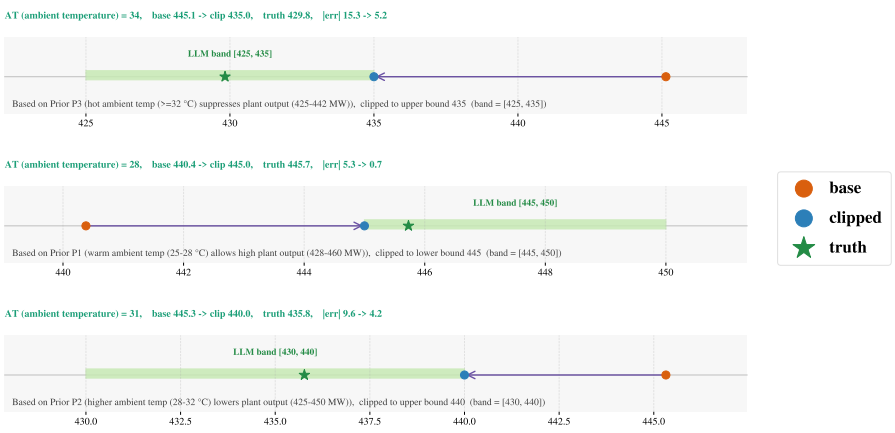

Figure 3: Sample-level analysis. Each panel shows the base prediction, the corrected prediction, the groundtruth, and the input-dependent band produced from the broad prior and proxy rules.

5.4 Sample-Level Analysis of Prediction Correction

Aggregate metrics show how the method affects overall predictive quality, but not how it corrects individual predictions. We therefore provide a sample-level analysis using TabPFN on the CCPP dataset. In Figure 3, the feature AT denotes ambient temperature, which is related to the power output of the combined-cycle plant. The broad prior describes this relationship at a semantic level: warmer ambient conditions should limit the attainable output. By combining this prior with the proxy rules, the LLM derives more specific output ranges for different temperature regions.

Figure 3 shows how three predictions are corrected under different input-dependent rules. For example, at $\mathrm{AT} = 34$, the base prediction is 445.1 while the refined band is $[425, 435]$. The correction moves the prediction to 435.0, reducing its absolute error from 15.3 to 5.2. Three observations follow from these examples. First, each refined band covers the corresponding test label, and projection reduces the absolute error in all three cases. Because the broad prior does not directly specify sample-level bounds, it must be interpreted and aligned with the proxy rules. The LLM provides this capability by translating the language description into refined ranges. Second, each correction can be traced through the decision path of the proxy tree and the rule selected for that sample. The tree-based proxy therefore makes the basis of the correction inspectable. Third, the three samples activate different rules according to their input regions. As a result, the correction can move a prediction in either direction instead of applying a fixed global shift.

These cases also clarify the role of the LLM stage described in Section 4.2. The input prior, shown in gray, provides only a broad description of the relationship between temperature and output. Rather than copying its coarse ranges directly, the LLM combines the prior with the proxy rules. This process produces narrower, region-specific bands, shown in green. The resulting bands express the language prior as constraints that can be applied to individual predictions. Overall, these refined bands preserve the intended relationship while providing more precise ranges. This refinement enables stricter prior-based control over individual predictions.

6 Conclusion

In this paper, we explore how to integrate human-specified language priors into PFN-based models. This task presents the challenge of aligning these two modalities. To address it, we propose **PriorBridge**, a pipeline that uses readable rules as an intermediate representation. These rules are extracted from trained tree-based proxy models and summarize the prediction behavior of PFNs. An LLM then combines the proxy rules with the prior constraints and refines them into decision paths with specified output ranges. A final projection applies these ranges to PFN predictions during inference, providing prior-based control. Our experiments confirm that our method consistently improves prior compliance across different PFN backbones. They also show that the incorporated priors can complement patterns not fully represented in the context data. Overall, PriorBridge provides an effective way to integrate language priors into PFN-based tabular prediction.

AI USE STATEMENT

In this work, we used generative AI tools for *generating synthetic data sets, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting dataset* . We have not used generative AI tools for *supporting qualitative and thematic data analysis, interpreting results* , and *helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and proposing or refining hypotheses* are not applicable to this work. Additionally, we used generative AI tools for *sourcing/searching for information, editing our paper to improve readability* . We have reviewed all AI-assisted work. Two main authors check the code and examine all the texts in this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not raise any ethical issues. We did not collect sensitive personal data or release datasets that could potentially cause harm. All experiments were conducted in compliance with relevant institutional guidelines and legal regulations.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we include the implementation code in the supplementary materials. Upon acceptance of this paper, subject to internal review approval and feasible conditions, we plan to further open-source the model weights and datasets to support future research.

REFERENCES

Arthur Asuncion, David Newman, et al. Uci machine learning repository, 2007.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *arXiv preprint arXiv:1708.03731*, 2017.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. Large scale transfer learning for tabular data via language modeling. *Advances in Neural Information Processing Systems*, 37:45155–45205, 2024.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943, 2021.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35: 507–520, 2022.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Benjamin Jäger, Dominik Safaric, Simone Alessi, Adrian Hayler, et al. Tabpfn-2.5: Advancing the state of the art in tabular foundation models. *arXiv preprint arXiv:2511.08667*, 2025.

Léo Grinsztajn, Klemens Flöge, Oscar Key, Felix Birkel, Philipp Jund, Brendan Roof, Mihir Manium, Shi Bin Hoo, Magnus Bühler, Anurag Garg, et al. Tabpfn-3: Technical report. *arXiv preprint arXiv:2605.13986*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. Tabllm: Few-shot classification of tabular data with large language models. In *International conference on artificial intelligence and statistics*, pp. 5549–5581. PMLR, 2023.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

AI Meta. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*, 2(5): 6, 2024.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*, 2026.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

TabLDM Team, Penghui Wang, Wei Liu, Hong Wang, Chengyue Huang, Yuxi Sun, Zirui Wang, Hongming Huang, Quan Wang, Chunxiao Liu, et al. Xiaomi-tabldm: A tabular foundation model technical report. *arXiv preprint arXiv:2609.03880*, 2026.

Yunze Tong, Fengda Zhang, Zihao Tang, Kaifeng Gao, Kai Huang, Pengfei Lyu, Jun Xiao, and Kun Kuang. Latent score-based reweighting for robust classification on imbalanced tabular data. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Yuanrui Wang, Xingxuan Zhang, Han Yu, Mingchao Hao, Gang Ren, Hao Yuan, Li Mao, Yunjia Zhang, Chun Yuan, and Peng Cui. Limix-2m: Mitigating low-rank collapse and attention bottlenecks in tabular foundation models. *arXiv preprint arXiv:2606.04485*, 2026.

Zhihan Yang, Jiaqi Wei, Xiang Zhang, Haoyu Dong, Yiwen Wang, Xiaoke Guo, Pengkun Zhang, Yiwei Xu, and Chenyu You. Forestllm: Large language models make random forest great on few-shot tabular learning. *arXiv preprint arXiv:2601.11311*, 2026.

Hangting Ye, Jinmeng Li, He Zhao, Dandan Guo, and Yi Chang. Llm meeting decision trees on tabular data. *Advances in Neural Information Processing Systems*, 38:118095–118131, 2026.

Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The twelfth International Conference on Learning Representations*, 2024.

Xingxuan Zhang, Gang Ren, Han Yu, Hao Yuan, Hui Wang, Jiansheng Li, Jiayun Wu, Lang Mo, Li Mao, Mingchao Hao, et al. Limix: Unleashing structured-data modeling capability for generalist intelligence. *arXiv preprint arXiv:2509.03505*, 2025.

Xingxuan Zhang, Gang Ren, Hao Yuan, Hao Zou, Hongze Tan, Hui Wang, Jianhao Song, Jiansheng Li, Jiayao Zhang, Jinghan Zhang, et al. Limix-2: A contextual mechanism network towards general structured-data intelligence. *arXiv preprint arXiv:2609.17488*, 2026.

A Additional Method Details

A.1 Classification Probability Correction

For binary classification, the correction step operates on the probability p_1 of the positive class. A prior referring to either class is translated into an interval constraint on p_1. We then project p_1 to the applicable leaf interval and use the same decision threshold as the base PFN to conduct correction.

For multiclass classification, the constraint specification identifies the class probability to which the prior applies. Let p_c denote this probability for a target class c. The applicable leaf interval is used to project p_c. The other class probabilities are then rescaled proportionally so that the corrected output remains a valid probability vector. The final prediction is obtained from the corrected vector using the standard maximum-probability decision rule.

A.2 Filtering Invalid Rule Trees

Before applying the correction, we check whether each LLM-generated rule tree can be used as a complete rule structure. If a tree does not satisfy the required structure, we discard the entire tree rather than repairing individual branches. This conservative treatment avoids combining partial rules from an internally inconsistent tree. The remaining valid trees determine the constraints used for prediction correction. Since we obtain multiple candidate trees as described in Section 4.2, occasional failed generations can often be tolerated without discarding the entire correction step. The remaining candidates can then provide usable rule trees for defining the correction constraints.

B Additional Analysis

B.1 How Language Priors Reshape PFN Predictions

Based on airfoil dataset, Figure 4 illustrates how the correction changes overall prediction trends. Before correction, LimiX produces a constant partial-dependence curve over the analyzed feature. Its rank correlation with feature frequency is therefore undefined. After correction, the curve follows the decreasing ranges stated by the prior, with a Spearman correlation of -0.83. All evaluated individual conditional-expectation points then lie inside the allowed ranges, compared with 60.5% before correction. This result illustrates the mechanism on a monotonic regression prior: the tree rules turn a language-specified relationship into an input-dependent change in the PFN prediction function. Due to the natural existence of leaf nodes for tree models, our method achieves splitting regions and in turn corrects original predictions.

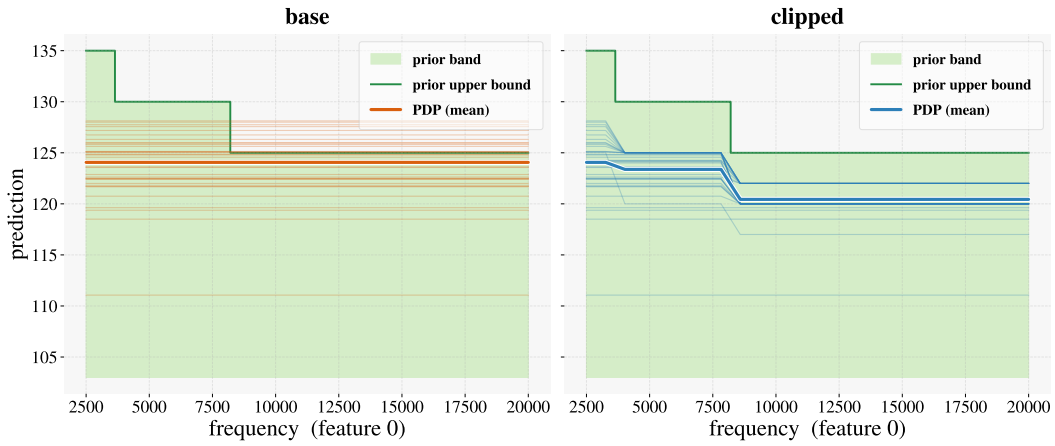

Figure 4: Effect of our method on the LimiX prediction function for airfoil dataset. The base partial-dependence curve is constant over frequency, whereas correction introduces the decreasing, stepwise response specified by the language prior.

Table 4: Cross-model proxy results on IID datasets. The base predictor is TabPFN-v2, while the proxy rules are distilled from LimiX predictions. All other setting aligns with Section 5.1.

Dataset	#P	Base	Ours	Comp. (%)
Regression (MAE ↓)				
Abalone	3	**1.38**	1.41	59.9→100
Beijing PM2.5	3	**49.4**	50.0	91.7→99.9
California Housing	3	**23649**	23837.6	88.5→100
Wine Quality Red	3	0.474	**0.459**	28.0→88.9
Binary classification (accuracy ↑)				
Australian Credit	2	0.885	**0.894**	71.4→100
Diabetes	2	0.733	**0.750**	92.6→100
Electricity	3	0.702	**0.790**	52.1→100
KC2	2	**0.861**	**0.861**	78.9→100
NSL-KDD	3	0.764	**0.930**	23.1→100
QSAR Biodegradation	2	0.855	**0.874**	58.5→75.0
Multiclass classification (accuracy ↑)				
Gas Drift	3	0.925	**0.947**	11.2→99.8
Average compliance		59.6→**96.7**		

Table 5: Cross-model proxy results on datasets with distribution shifts. The setting follows Table 4.

Dataset	#P	Base	Ours	Comp. (%)
Regression (MAE ↓)				
Airfoil	3	5.79	**4.69**	60.9→100
CCPP	3	4.49	**4.44**	95.4→100
Concrete	3	**8.21**	8.24	98.5→100
Bike Sharing	3	**89.6**	115	90.0→95.4
Gas Turbine NOx	3	14.5	**13.0**	45.2→100
Seoul Bike	3	**339**	374	99.5→99.8
Superconductivity	3	23.9	**23.8**	98.2→100
Binary classification (accuracy ↑)				
CICIDS2017	3	0.873	**0.906**	14.8→100
Multiclass classification (accuracy ↑)				
Dry Bean	1	0.704	**0.705**	99.3→100
Gesture Phase	1	0.425	**0.446**	8.9→100
Average compliance		71.1→**99.5**		

B.2 Cross-Model Proxy Rules

In our main experiments, each proxy is distilled from the PFN that is subsequently corrected. In this section, we instead evaluate a cross-model configuration in which TabPFN provides the base predictions, while the proxy rules are distilled from LimiX predictions. The LLM then refines these LimiX-derived rules using the same language prior. The resulting constraints are applied to the TabPFN predictions.

This setting tests whether the correction procedure requires the proxy source to match the PFN being corrected. Tables 4 and 5 report the predictive metrics and prior compliance before and after correction. Our method still improves prior compliance in this setting. This confirms that the generated ranges remain aligned with the language constraints even when the proxy and the corrected PFN are different models. The ranges are therefore not tied to the particular PFN that supplied the proxy rules. For some datasets, applying the cross-model rules also improves predictive quality, including cases where the corresponding main setting does not show the same improvement. We leave a more systematic study of how to combine different PFN models for future work.

Overall, these results show that rules extracted from one PFN can guide predictions from another PFN. They provide additional evidence that the framework is not tied to a particular PFN backbone and that its proxy interface can transfer across PFN models.

C Examples of Language Priors

The language priors used in our experiments describe domain relationships between input features and target predictions. They may specify output ranges for particular input regions, conditional class relationships, or monotonic trends. To illustrate their form, we show selected parts of the prompt used for the CCPP dataset below. The prompt combines an out-of-distribution context with region-specific output ranges and a monotonicity constraint. Our instructions on proxy rules refer to the template of DeLTa (Ye et al., 2026).

Representative Instruction Prompt for CCPP

```
You are an expert in tabular machine learning domain. I will provide the meta
information, the CART tree rules about the prediction task. Please help me design a
better rule for inference.

...

## OOD Context
Training data contains only cool/moderate ambient temperatures (AT < 25 degrees C).
Test data contains only hot conditions (AT >= 25 degrees C). The combined cycle gas
turbine (CCGT) operates at reduced efficiency in hot conditions due to thermodynamic
constraints (Carnot cycle). The model has never seen high-temperature operating
conditions and may systematically mis-predict power output in this regime.

## Prior Knowledge (Hard Constraints)
The following constraints are MANDATORY and must be strictly encoded in your rule tree.
 They derive from thermodynamic first principles (Carnot efficiency, gas turbine theory
):

1. Range Constraint: When 25 <= AT < 28 (feature 0), PE should be in [428, 460].
   Near the training boundary. Carnot extrapolation: +3 degrees C above the boundary
reduces efficiency by approximately 1%.

2. Range Constraint: When 28 <= AT < 32 (feature 0), PE should be in [425, 450].
   Moderate hot conditions. Greater efficiency loss narrows the output range.

3. Range Constraint: When AT >= 32 (feature 0), PE should be in [425, 442].
   Extreme hot conditions. Maximum efficiency loss gives the tightest range.

4. Monotonicity: AT should have a non-increasing relationship with PE.
   Higher ambient temperature should lead to lower power output.

## Prediction Envelope: range for each leaf

CRITICAL NEW REQUIREMENT: Each leaf node must include a "range" field specifying the
allowed prediction interval [lo, hi] for that region.

Rules for setting ranges:
- The constrained regions MUST use the ranges specified above.
- For free regions within the same AT bracket, you may use slightly wider ranges.
- Ranges across leaves ordered by AT should reflect monotonicity.
  Higher AT should correspond to a lower or equal range upper bound.
- Use null for unbounded sides if appropriate.
- V (exhaust vacuum) also affects PE: higher V leads to lower PE. Consider this for sub
-regions.

## Operator and branching semantics

CRITICAL: The "left" branch is ALWAYS the branch where the condition is TRUE.
- "<" means: if X[feature] < threshold, go to "left"; otherwise go to "right".
- ">=" means: if X[feature] >= threshold, go to "left"; otherwise go to "right".

For example, to route samples with AT >= 32 to a leaf with a tight range:
{"feature": 0, "threshold": 32, "operator": ">=", "left": {"id": "leaf_extreme_hot", "
range": [425, 442]}, "right": {... subtree for AT < 32 ...}}

...
```

This excerpt shows that a prior can contain both a high-level description of an OOD regime and executable constraints on different input regions. It also specifies how the constraints should be represented in the rule tree, rather than providing only a single global target value.