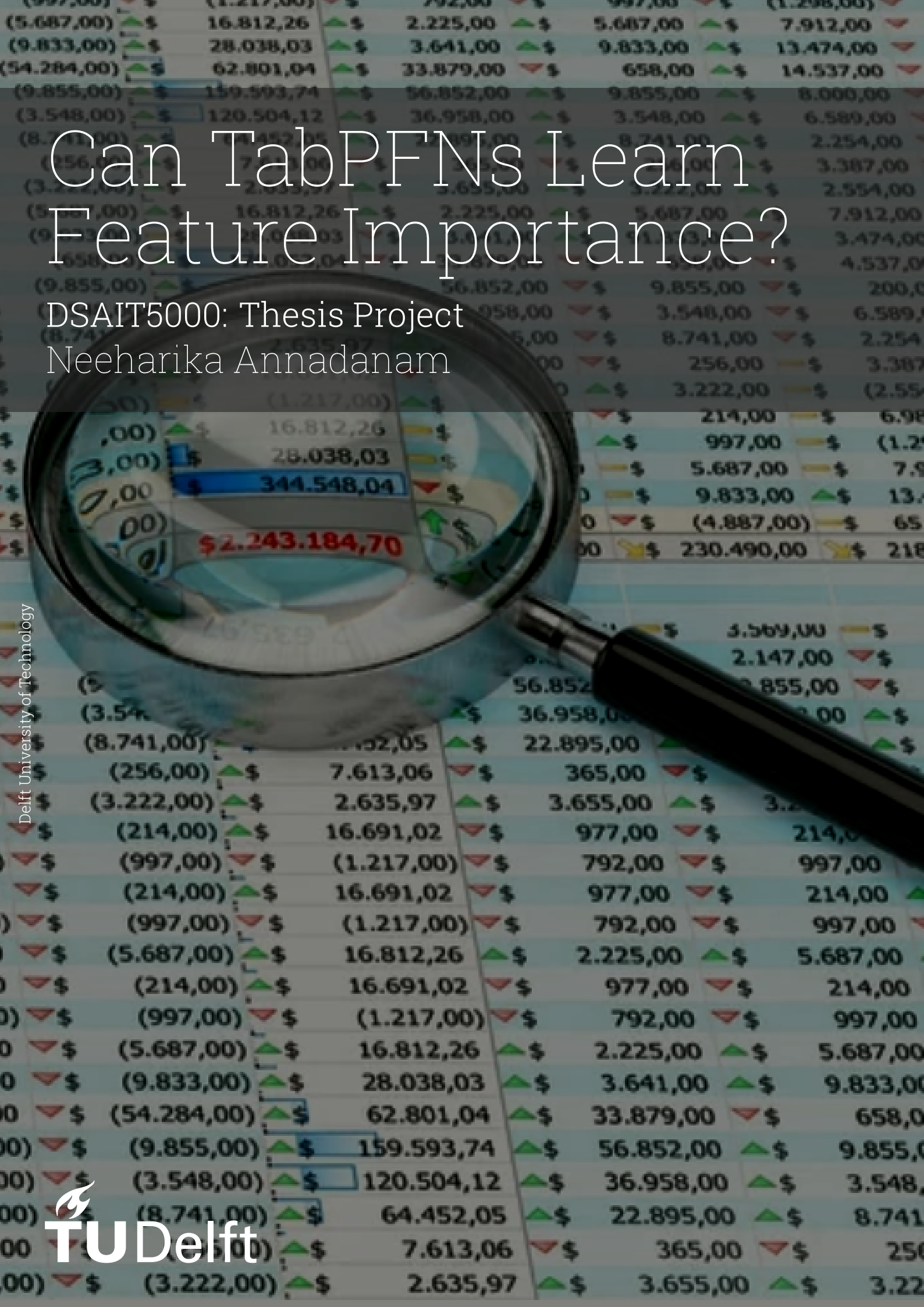
Can TabPFNs Learn Feature Importance?

DSAIT5000: Thesis Project

Neeharika Annadanam



Delft University of Technology

TU Delft

Can TabPFNs Learn Feature Importance?

by

Neeharika Annadanam

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday, September 30, 2026 at 08:45

Student Number:	6161111
Project Duration:	November 2025 - September 2026
Thesis Committee:	Dr. T.J. Viering, TU Delft, Daily Supervisor
	Dr. J.H. Krijthe, TU Delft, Thesis Advisor
	Dr. J. Yang, TU Delft, Committee Member

TU Delft

Acknowledgments

I would like to express my gratitude to Dr. Tom Viering and Dr. Jesse Krijthe for their supervision, and Dr. Jie Yang for being part of the thesis committee. I would also like to thank Cheng Yan for the help. I'm grateful to friends and family, near and far, for all their support.

Neeharika Annadanam
Delft, September 2026

Abstract

In this work, we extend nanoTabPFN, a Prior-Data Fitted Network (PFN) for tabular classification to produce Shapley attributions alongside its predictive distribution. The output is a bucketed distribution over signed feature attribution values. The class prediction is derived as the additive sum of a baseline and per-feature contributions, ensuring the explanation is faithful to the prediction by construction. The attribution head is supervised by TreeSHAP values from a decision tree prior, connecting the explanation to the data generating process (DGP). The resulting attributions are interpreted as a posterior over Shapley values. The PFN's distributional mechanism is transferred to feature attributions to yield calibrated explanation uncertainty. Across synthetic and real tabular benchmarks, the model recovers attributions with high fidelity ($R2 \approx 0.94$-0.95 against exact Shapley on synthetic data), but at a small cost to accuracy. It produces calibrated attribution uncertainty, with negative log-likelihood (NLL) below the uninformative uniform baseline.

Contents

1

Introduction

Tabular data is a prevalent modality in real world applications. In high-stakes domains such as healthcare and climate science, in addition to accuracy, analysis of explainability is important, as it may be a regulatory or ethical requirement [29]. Classical approaches such as gradient boosted trees have dominated tabular benchmarks [32], but they do not quantify uncertainty.

By meta training a transformer on synthetic datasets sampled from a prior distribution, PFNs learn to approximate Bayesian inference in a single forward pass [25]. They output a posterior predictive distribution (PPD), providing a measure of uncertainty based in Bayesian theory. TabPFN [11] extends the PFN framework to tabular data, leveraging prior knowledge learned from synthetic datasets to make predictions. It achieves competitive accuracy on tabular datasets with low inference costs.

However, PFNs remain opaque and can only be explained through post-hoc methods, which may be a limitation in domains where explainability is critical, since post-hoc interpretations are not guaranteed to be faithful to the model's actual computation and can therefore be misleading [17].

One way to address this limitation is to modify the model architecture so that an explanation is a native output, in addition to the prediction. This ensures that a post-hoc approximation would not be required. This work explores whether PFNs can be extended to produce explainability metrics as a posterior over its possible values, so that the explanation carries Bayesian uncertainty like the prediction, without sacrificing their predictive strength.

This matters because an importance close to zero may indicate that the feature is irrelevant, or that it cannot determine the feature's effect at all. A distribution separates a posterior concentrated at zero from a wide posterior whose mean is at zero. Reporting the full posterior would allow an end user to distinguish between reliable feature attribution scores and uncertain ones.

We ask whether the PFN architecture can be modified to produce feature importance scores alongside its predictive distribution, how well those scores agree with exact Shapley values, and whether they are faithful to the model's predictive behaviour, whether the architectural modification changes the classification accuracy of PFNs, and whether the attribution scores can be produced as distributions such that the model expresses calibrated uncertainty about its explanations.

2

Background

This chapter introduces the concepts used in Chapter 3.

2.1. Bayesian Inference

Bayesian inference is a method for reasoning under uncertainty over the underlying data-generating process. It treats model parameters, variables and predictions as probability distributions, based on Bayes' theorem:

$$p(\theta \mid D) = \frac{p(D \mid \theta)p(\theta)}{p(D)}$$

where θ are the parameters, D is the observed data, $p(D|\theta)$ is the likelihood and $p(\theta|D)$ is the posterior. The normalizing constant $p(D) = \int p(D|\theta)p(\theta)d\theta$ is the marginal likelihood. The posterior $p(\theta|D)$ is a distribution over all models weighted by likelihood $p(D|\theta)$ and how plausible it was before observing the data (prior $p(\theta)$).

This is in contrast to Maximum Likelihood Estimation (MLE), which finds a single $\hat{\theta}_{MLE}$ that maximizes $p(D|\theta)$. For a new input x, the predicted target is given by $\hat{y} = f(x; \hat{\theta}_{MLE})$.

With Bayesian Inference, for a new test point x, a prediction is made by marginalizing over the posterior:

$$p(y|x, D) = \int p(y|x, \theta)p(\theta|D)d\theta$$

This posterior predictive distribution (PPD) is a distribution over y, where y is the target to be predicted. It captures both aleatoric uncertainty, which arises from irreducible noise in the data generating process, and epistemic uncertainty, which arises from limited data and uncertainty in the model parameters. A model that outputs only a point estimate discards epistemic uncertainty, whereas a model that outputs the full PPD does not.

The prior $p(\theta)$ encodes assumptions about the data generating process before any observations are made. An informative prior can improve generalization in cases where there is low data by ruling out implausible models. A misleading prior can bias posterior inference towards regions of the parameter space that do not reflect the true process.

In tabular machine learning, the prior is over the data generating process. A generative procedure is used to produce synthetic datasets. Each generated dataset is treated as a sample from the prior.

2.2. Prior-Data Fitted Networks

Prior-Data Fitted Networks (PFNs) provide a way to amortize Bayesian inference. They learn to approximate the posterior predictive distribution (PPD) from data generated under the Bayesian prior.

In supervised learning, a training set $D_{\text{train}} = (x_i, y_i)_{i=1}^{n}$ is observed and the goal is to predict the label $\hat{y}$ for a new input x. In the Bayesian framework, a prior $p(\phi)$ is defined over a space of hypotheses Φ,

where each $\phi \in \Phi$ describes a data generating process (DGP) that determines how the synthetic data is generated.

The prior defines a generative process in which a hypothesis ϕ is first sampled according to $p(\phi)$. Then a dataset D is sampled according to the likelihood $p(D \mid \phi)$. The resulting marginal distribution over datasets is therefore

$$p(D) = \mathbb{E}_{\phi \sim p(\phi)}[p(D \mid \phi)]. \tag{2.1}$$

This distribution is the prior data distribution. It describes the synthetic datasets on which the model is trained. In this work, the hypothesis class Φ is the space of decision tree classifiers. Each ϕ is a decision tree classifier with randomly sampled depth, fitted on a fresh draw of random input points and random labels. The resulting tree defines a labeling function. A new set of inputs is then drawn and labeled by ϕ, yielding one synthetic dataset.

Given an observed dataset D, Bayesian prediction accounts for uncertainty over which hypothesis ϕ generated the data. Predictions for a test point x are obtained by integrating over all hypotheses weighted by their prior probability and likelihood:

$$p(y \mid x, D) \propto \int_{\Phi} p(y \mid x, \phi), p(D \mid \phi), p(\phi), \mathrm{d}\phi. \tag{2.2}$$

This quantity is the posterior predictive distribution (PPD). It represents the probability assigned to each possible label after taking into account both the prior uncertainty over hypotheses and the evidence provided by the observed dataset D.

Computing the PPD is often intractable because the integral requires marginalizing over the potentially very large hypothesis space Φ. Markov Chain Monte Carlo (MCMC) can provide accurate estimates of this integral but is computationally expensive [2], while Variational Inference introduces approximation bias [34]. Additionally, neither method amortizes Bayesian inference. Inference must be performed again for every new dataset.

Prior-Data Fitted Networks (PFNs) amortize this cost by training a transformer q_θ to approximate the PPD [25]. Instead of explicitly performing the integral in Equation 2.2 for every new dataset, the marginalization over hypotheses is learned during training and encoded in the parameters of the transformer.

To train the PFN, synthetic datasets are sampled from the prior data distribution in Equation 2.1. Each sampled dataset consists of input-label pairs (x_i, y_i) generated according to the sampled hypothesis. The PFN is then trained on these synthetic datasets to predict the label of a held out test point (x_{test}, y_{test}) given their training set D_{train}.

A transformer q_θ [33] is meta-trained to minimize the negative log-likelihood of held-out labels across synthetic datasets:

$$\mathcal{L}(\theta) = \mathbb{E}((x_{\text{test}}, y_{\text{test}}) \cup D_{\text{train}}) \sim p(D)\big[-\log q_\theta(y_{\text{test}} \mid x_{\text{test}}, D_{\text{train}})\big]. \tag{2.3}$$

Under the prior data distribution, minimizing this loss is equivalent to minimizing the KL divergence between q_θ and the true PPD. Consequently, the marginalization over hypotheses in Equation 2.2 is performed during training and, once the PFN has been trained, approximated at inference as a single forward pass.

For binary classification tasks, the PFN represents its prediction as a probability distribution over two buckets, corresponding to the two possible labels. We use a bucket to denote one discrete output category and denote the probability assigned to bucket b by

$$q_\theta(b \mid x_{\text{test}}, D_{\text{train}}). \tag{2.4}$$

The resulting bucketed output distribution is therefore a representation of the PPD, with probability mass distributed across the possible output labels. In addition to treating the PFN as a predictor of the most likely label, we also use these probabilities as a representation of the model's uncertainty.

2.3. Tabular Foundation Models (TFMs)

Tabular foundation models (TFMs) are pretrained models to generalize across a wide range of tabular prediction tasks. TFMs learn a prior over tabular data during pretraining and perform predictions on new datasets through in-context learning (ICL). This reduces the need for dataset specific parameter optimization and has recently emerged as an alternative to tree-based methods.

A development in this direction is TabPFN [12], which applies the Prior-data Fitted Network (PFN) framework to tabular classification. TabPFN is pretrained on synthetic datasets generated from a structural causal model (SCM) prior and learns to predict labels for unseen datasets directly from their training examples. It demonstrated strong performance against tuned XGBoost and random forest baselines on small datasets with up to 1000 rows and 100 features, although the original model was primarily limited to numerical features.

TabPFN v2 [11] extended the approach to larger datasets, supporting up to 10000 rows and 500 features. It introduced alternating column-wise and row-wise attention to improve scalability and added support for categorical features, missing values, and regression through a redesigned synthetic prior. These extensions made TabPFN more applicable to the heterogeneous structure of real-world tabular data.

nanoTabPFN [26] provides a lightweight and educational reimplementation of the TabPFN v2 architecture. It was designed to make tabular foundation models easier to understand and pretrain. It retains the core transformer architecture and PFN training framework. This makes nanoTabPFN a practical framework for experimenting with alternative architectures, priors, and training procedures, which is what this work does.

Input features are first normalized and clipped to bound outliers from the synthetic prior before a linear feature encoder embeds each cell x_{ij} into a d-dimensional space. A separate linear target encoder embeds the training labels y_i and adds them to the corresponding row embeddings. Test rows receive a learned placeholder in place of a label. The encoded table, of shape $(\text{rows} \times \text{features} \times d)$, is processed by a stack of transformer blocks. Each block applies row attention across rows within a feature/embedding channel and column attention across features within a row and finally a feed-forward network with each sub-layer wrapped with pre-layer-norm and a residual connection. After the final transformer block, embeddings of the query rows are pooled across features and passed through a decoder head that outputs bin logits, followed by a softmax to yield class probabilities, as shown in Figure 2.1.

TabICL [27] introduced a two-stage column-then-row architecture, enabling in-context learning on datasets with up to approximately 500,000 samples. By separating feature and sample processing, the model reduces the computational cost of applying attention to large tables. However, the original TabICL was focused on classification.

An important component of TabICL is its synthetic data prior, which determines the types of relationships encountered during pretraining. The original TabICL provides several SCM-based priors, namely MLP-SCM, Tree-SCM, and Mix-SCM [27]. MLP-SCM generates synthetic datasets using randomly parameterized multilayer perceptrons to model nonlinear relationships. Tree-SCM uses randomly generated tree-based models to produce piecewise and rule-based relationships. Mix-SCM combines the MLP and tree based generator to increase the diversity of the generated tasks. These priors generate a random dependency structure, sample root variables, and propagate them through randomly parameterized functions to obtain features and targets.

TabICLv2 [28] adds a graph-based SCM prior. Synthetic datasets are generated by first sampling a random graph, assigning features and targets to nodes, and then evaluating the graph using randomly selected node functions. The current TabICLv2 prior supports a diverse set of function types, including multilayer perceptrons (MLPs), tree ensembles, Gaussian processes (GPs), linear functions and product functions. This allows the generator to produce datasets with substantially different forms of smooth, nonlinear, piecewise and categorical relationships. Random sampling of graph structures, functions, parameters, feature distributions, and noise further increases the diversity of the resulting tasks. [28]. This work also extends the framework to both classification and regression and introduces repeated feature grouping to improve scalability and generalization. On benchmarks such as TabArena [6] and

TALENT [35, 19], TabICLv2 demonstrates competitive performance against tuned gradient-boosted tree models without requiring dataset specific hyperparameter tuning.

The main advances have focused on increasing dataset scale, supporting diverse feature types and prediction tasks, and improving the synthetic pretraining distributions.

2.4. Explainability and Shapley Values

Explainability refers to the degree to which a human can understand the causes of a model's decisions [16]. Broadly, two techniques exist, post-hoc and interpretability by design.

Post-hoc methods fit a separate explanatory model to a trained predictor's inputs and outputs. This is estimated alongside the model rather than read from its computation, so the explanation is not guaranteed to be faithful to what the predictor actually does [29].

Methods that are interpretable by design have explanations that reflect the model's actual computation. The explainability mechanism is deterministic, the same input always produces the same interpretation. Additionally, models that are interpretable by design are usually easier to debug and improve because we get insights into their inner workings [23].

Shapley values originate in cooperative game theory, where they provide payoff allocation satisfying efficiency, symmetry, dummy, and additivity properties [31, 23]. In this context, the game is the prediction for a single instance, the players are features, and the Shapley value ϕ_f of feature f is its average marginal contribution to the prediction over all 2^F subsets of the remaining features:

$$\phi_f = \sum_{S \subseteq \mathcal{F} \setminus \{f\}} \frac{|S|!\,(F - |S| - 1)!}{F!} \big[v(S \cup \{f\}) - v(S)\big]$$

where $v(S)$ denotes the model's expected output when only features in S are observed and the remaining features are marginalised out. Here, F is the total number of features and $\mathcal{F}$ is the set of features. These values satisfy the following axioms: they sum to the difference between the model's prediction and its global mean (efficiency), features with identical contributions receive equal values (symmetry), and a feature that never changes any prediction receives zero (dummy property), and for additive models such as random forests, Shapley values can be computed for each tree and averaged to obtain the overall explanation (additivity).

Exact computation Evaluating the value above requires 2^F coalitions, which is intractable for $F \gtrsim 20$. Exact computation is feasible only for small feature sets or for model classes whose structure can be exploited. Therefore, various approximation schemes have been developed, which we discuss next.

TreeSHAP For tree-based models such as decision trees, random forests, and gradient-boosted ensembles [21] derive an exact, polynomial-time algorithm. TreeSHAP traverses each tree and propagates, at each internal node, the proportion of training samples that would reach each branch when a coalition S is fixed at their observed values and the complementary features are integrated out over the training distribution. This yields exact (deterministic) Shapley values in $\mathcal{O}(TLD^2)$ time, where T is the number of trees,L the number of leaves, and D the maximum depth. This formulation marginalizes each missing feature independently, which coincides with the conditional expectation only when features are uncorrelated.

KernelSHAP For black-box models, [20] estimates Shapley values as a weighted linear regression problem over sampled coalitions. Each coalition $S \subseteq \mathcal{F}$ is represented as a binary masking vector $z \in \{0,1\}^F$. The Shapley kernel assigns weight

$$\pi(S) = \frac{F - 1}{\binom{F}{|S|}|S|(F - |S|)}$$

which upweights coalitions of extreme size because these provide the largest marginal signals. The coalition value $v(S)$ is estimated by replacing masked features with samples from a background dataset. KernelSHAP is model-agnostic. It introduces two sources of variance, from the sampling of coalitions, and how each coalition is estimated, $v(S)$.

Interventional Shapley Values Standard Shapley values treat all feature subsets as equal and ignore any causal structure among features. This raises the question of what it means to "remove" a feature from a coalition. [14] argue that feature relevance should be understood in terms of a model's computation. From this perspective, removing a feature means removing the information provided by that feature without allowing correlated features to reconstruct it. The interventional value function is:

$$v_{\text{int}}(S) = \mathbb{E}_{\mathbf{x}_{\bar{S}}}\left[f(\mathbf{x}_S, \mathbf{x}_{\bar{S}})\right]$$

Using this formula, the missing features are marginalized over their marginal distribution $p(\mathbf{x}_{\bar{S}})$. This removes statistical dependence between S and $\bar{S}$. The observed features are not used to infer the missing features. This formulation therefore explains the model's computation and not the data's correlation structure [14].

The motivation is that if a feature is removed, correlated features should not be able to provide information about the removed feature. Thus, a feature can receive zero attribution even if it is strongly correlated with a feature that the model uses, since correlation with a feature relevant to a model does not necessarily mean that the model itself uses that feature.

Conditional Shapley Values Here, the data's correlation structure can instead be represented by the conditional value function:

$$v_{\text{cond}}(S) = \mathbb{E}_{\mathbf{x}_{\bar{S}}|\mathbf{x}_S}\left[f(\mathbf{x}_S, \mathbf{x}_{\bar{S}})\right]$$

where $\mathbf{x} = (x_1, \ldots, x_F)$ is an instance and $S \subseteq \mathcal{F}$ is a coalition and $\bar{S} = \mathcal{F} \setminus S$.

Using this formula, the missing features are marginalized over $p(\mathbf{x}_{\bar{S}} \mid \mathbf{x}_S)$. Knowing $\mathbf{x}S$ updates beliefs about $\mathbf{x}\bar{S}$, and that update propagates into the expectation. Therefore, conditional Shapley values preserve the statistical dependencies between observed and missing features.

This leads to a different interpretation of feature relevance. The conditional formulation accounts for information about missing features that can be inferred from the observed features [10]. The interventional formulation asks which features the model relies on for its computation. Conditional Shapley values explicitly account for the causal relationships between features. Removing or intervening on a feature can therefore affect its downstream features according to the causal graph.

Rather than requiring the full causal graph, [8] modify the coalition structure to respect a causal graph. This ensures that upstream and downstream features in the causal graph are not treated as exchangeable.

These approaches therefore differ in how they define the value function $v(S)$, i.e. what it means to remove a feature from a coalition. The interventional formulation explains the model's computation by marginalizing missing features over their marginal distribution, while the conditional formulation preserves the data's correlation structure. Causal Shapley values incorporate the causal relationships between features and propagating the effects of interventions through the causal graph.

Beeswarm Plot A common way to summarize feature attributions across many instances at once is the beeswarm plot. Each row corresponds to one feature, ordered by mean absolute ϕ_j over the evaluated instances (most influential at the top). Each point corresponds to one (instance, feature) pair, positioned on the x-axis by its signed ϕ_j and colored by the instance's feature value (red = high, blue = low). This combines a global importance ranking with a dependence view of whether high or low feature values push the prediction up or down.

Figure 2.2 shows a beeswarm plot applied to the attributions from the modified nanoTabPFN model on a credit-scoring dataset [1], where ϕ_j is the model's additive contribution to predicted bad-client. The target is binary, with 1 indicating a bad (high-risk) client and 0 indicating a good (low-risk) client. The most influential features are `sex_male`, `family_status_Another`, and `month`. For instance, when `sex_male` is low (blue), it consistently pushes the prediction toward higher risk while high values (red) push it lower.

[1] `https://www.openml.org/search?type=data&sort=runs&id=43442&status=active`

Figure 2.3 shows a beeswarm plot applied to the diabetes dataset[2], where ϕ_j is the model's additive contribution to predicted diabetes risk. The target is binary, with 1 indicating a diabetic (high-risk) patient and 0 indicating a non-diabetic (low-risk) patient. The most influential features are `Glucose`, `Age`, and `BMI`. When `Glucose` is high (red), it consistently pushes the prediction toward higher risk while low values (blue) push it lower.

These illustrate the qualitative information a beeswarm plot conveys.

Waterfall Plot Figure 2.4 and Figure 2.5 show the waterfall plots for a single instance for two different datasets. They show how one specific prediction is built up feature by feature from a baseline.

The two per-class logits $(\text{logit}_0, \text{logit}_1)$ are passed through a softmax to obtain class probabilities, and the predicted label is the argmax.

In Figure 2.4, an applicant starts from a class-1 ("bad client") baseline of $\text{base}_1 = -1.092$. The features with the largest magnitude of $|\phi_j|$ of 42 one-hot encoded features total contribute $+0.079$, $+0.057$, $+0.054$, $+0.051$, $+0.025$, -0.013, -0.012, $+0.012$, and the remaining 34 features contribute a further -0.045 in aggregate, giving $f(x) = \text{base}_1 + \sum_f \phi_{f,1} = -0.882$.

Softmax over both class logits yields $P(\text{class } 1) = 0.249 < 0.5$. Although several features (`month`, `family_status_Another`, `product_type_Cell phones`, `sex_male`) individually push toward higher predicted risk, their combined evidence is not enough to overcome the strongly negative baseline, and the model predicts class 0 ("good client").

In Figure 2.5, a patient starts from a class-1 ("diabetes-positive") baseline of $\text{base}_1 = -0.294$. The 8 features of the diabetes dataset contribute $\phi_{j,1}$ values of $+0.124$ (`Glucose`), -0.075 (`BMI`), -0.035 (`DiabetesPedigreeFunction`), $+0.033$ (`Age`), $+0.024$ (`Pregnancies`), $+0.006$ (`Insulin`), -0.004 (`BloodPressure`), and $+0.001$ (`SkinThickness`), giving

$$f_1(x) = \text{base}_1 + \sum_j \phi_{j,1} = -0.294 + 0.075 = -0.219 \tag{2.5}$$

The additive decomposition is applied independently to both class logits, so class 0 ("tested negative") has its own baseline value base_0:

$$f_0(x) = \text{base}_0 + \sum_f \phi_{f,0} = 0.177 + (-0.487) = -0.310 \tag{2.6}$$

The predicted class probability is then the softmax of the two logits,

$$P(\text{class } 1) = \frac{e^{f_1(x)}}{e^{f_0(x)} + e^{f_1(x)}} = \frac{e^{-0.219}}{e^{-0.310} + e^{-0.219}} = \frac{0.804}{0.734 + 0.804} \approx 0.523 \tag{2.7}$$

equivalently $P(\text{class } 1) = 1/(1 + e^{-(f_1(x) - f_0(x))}) = 1/(1 + e^{-0.091}) \approx 0.523$

Although `BMI` and `DiabetesPedigreeFunction` individually push toward lower predicted risk, the dominant positive contribution from `Glucose` and smaller positive contributions from `Age` and `Pregnancies` are enough to overcome both the negative baseline and the negative features, and the model predicts class 1 ("diabetes-positive").

[2] `https://www.openml.org/search?type=data&sort=runs&id=43582&status=active`

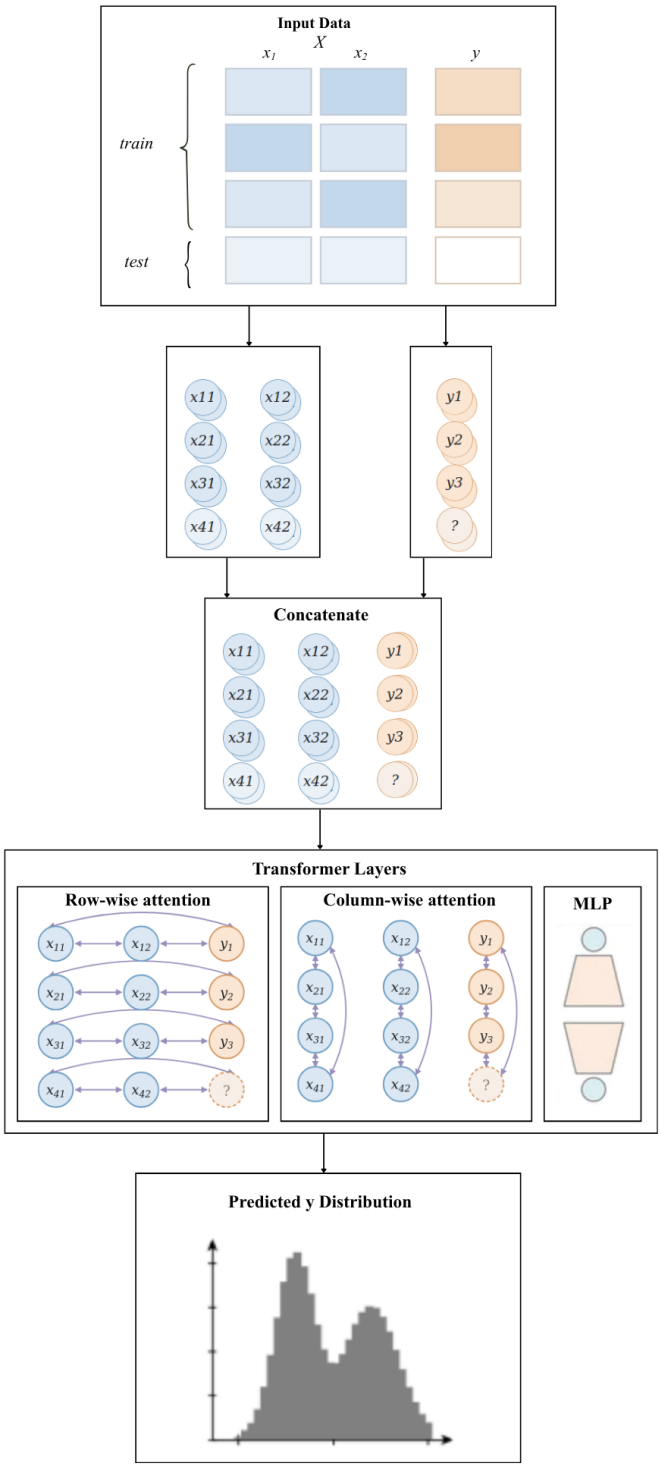


Figure 2.1: Architecture of nanoTabPFN.

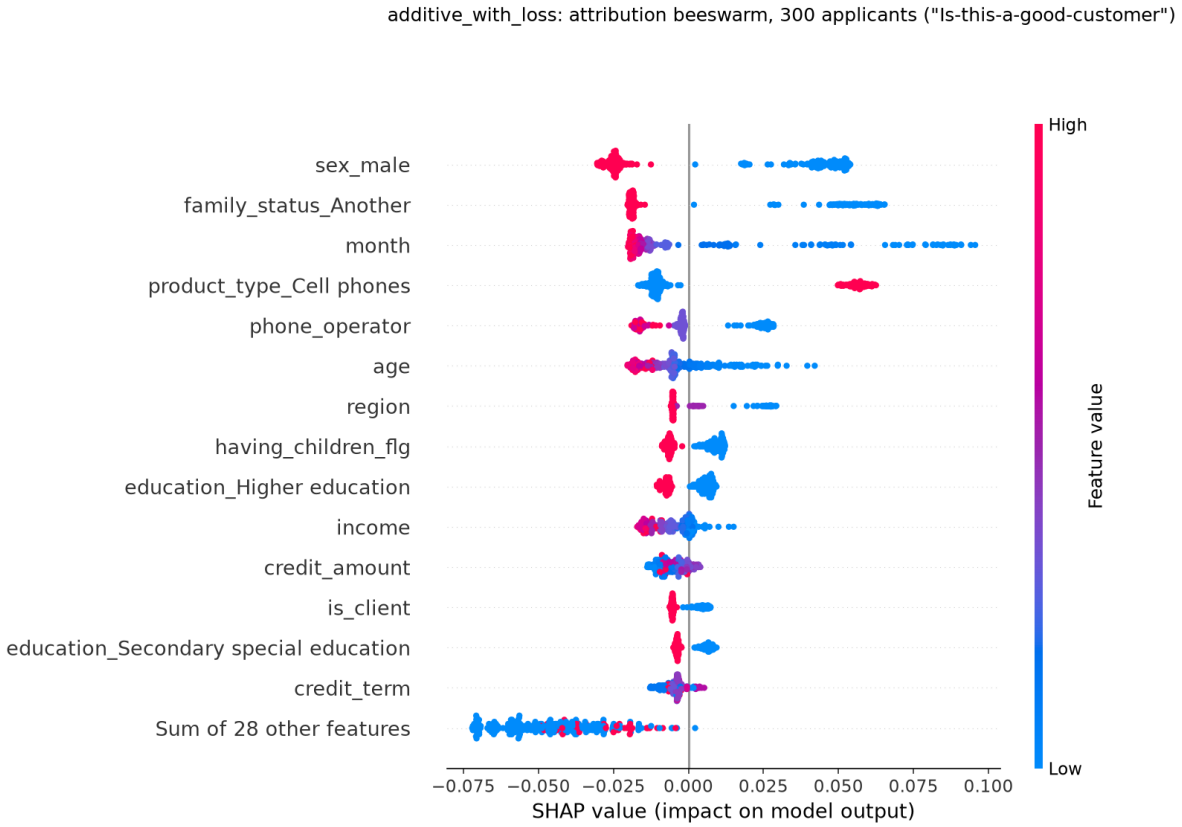


Figure 2.2: Beeswarm plot: signed contribution ϕ_j to predicted bad client risk.

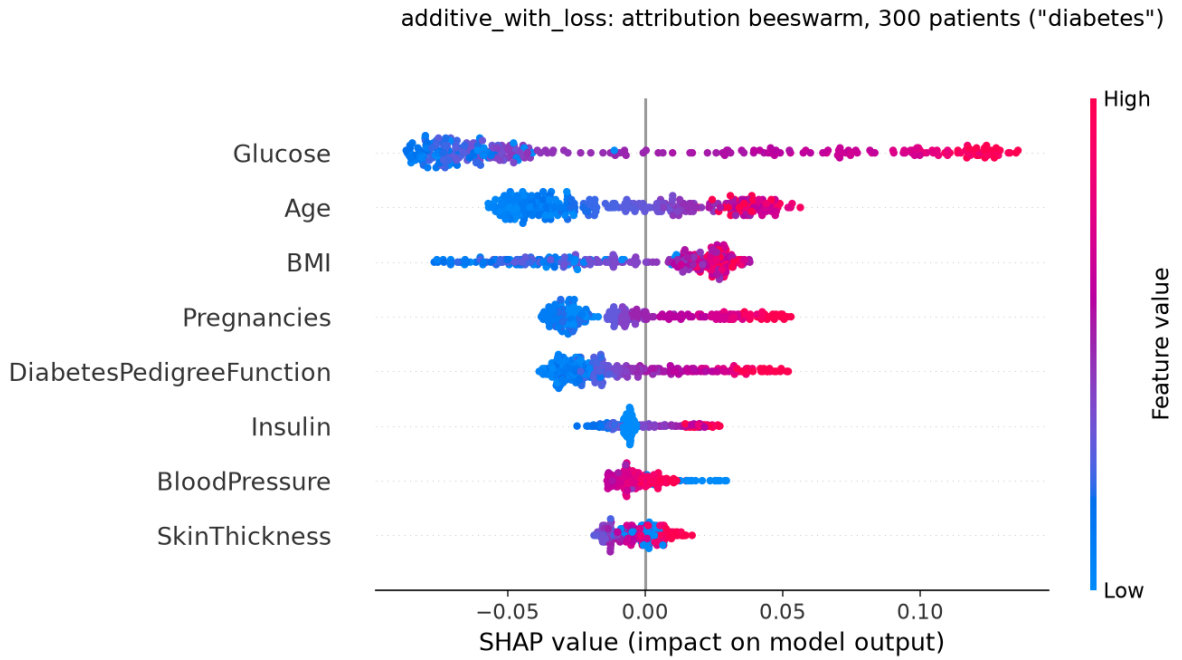


Figure 2.3: Beeswarm plot: signed contribution ϕ_j to predicted diabetes risk.

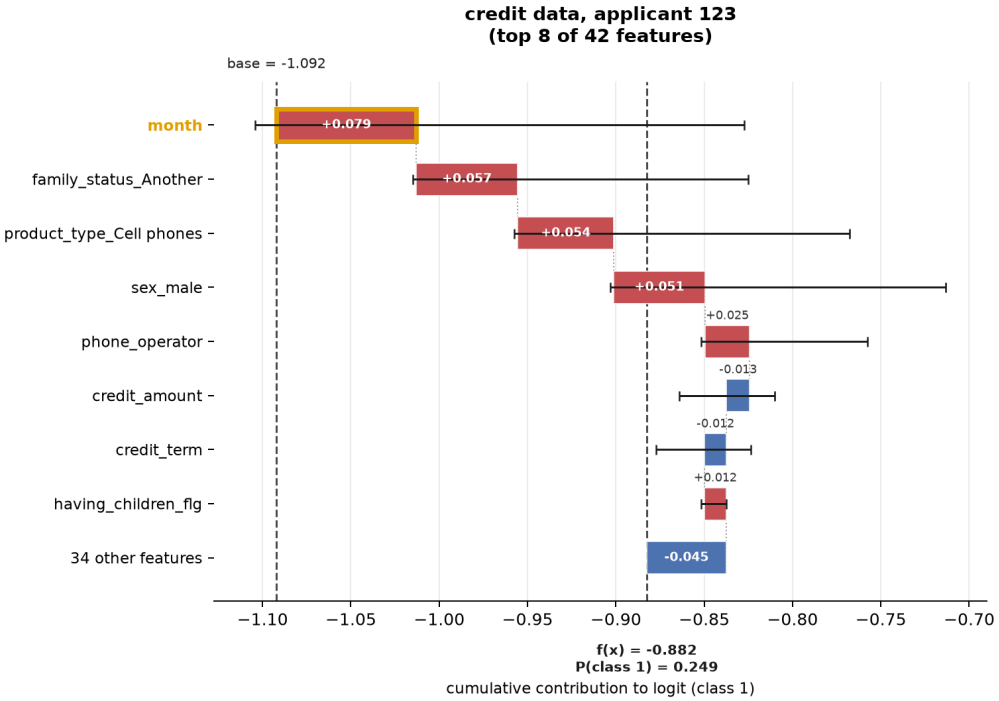


Figure 2.4: Additive decomposition of the model's class-1 logit for one applicant from the credit dataset: $\text{base} + \sum_j \phi_j = f(x)$. Whiskers show each feature's credible interval from its predicted bucket distribution.

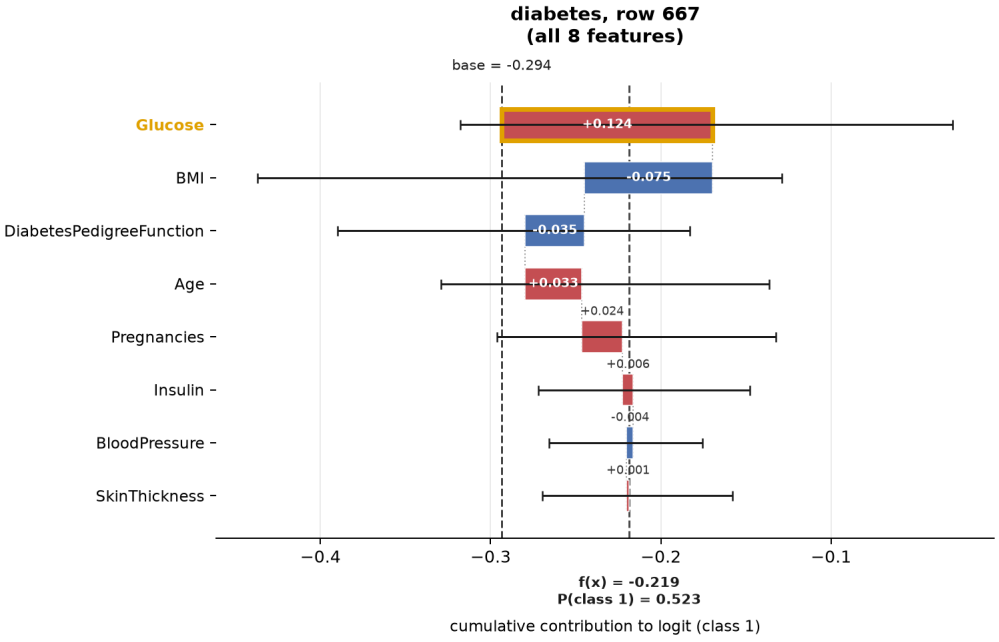


Figure 2.5: Additive decomposition of the model's class-1 logit for one applicant from the diabetes dataset: $\text{base} + \sum_j \phi_j = f(x)$. Whiskers show each feature's credible interval from its predicted bucket distribution.

3

Scientific Article

Can PFNs Learn Feature Importance?

Neeharika Annadanam [1] **Tom Viering** [1] **Cheng Yan** [1]

Abstract

We extend nanoTabPFN, a Prior-Data Fitted Network (PFN) for tabular classification to produce Shapley attributions alongside its predictive distribution. The output is a bucketed distribution over signed feature attribution values. The class prediction is derived as the additive sum of a baseline and per-feature contributions, ensuring the explanation is faithful to the prediction by construction. The attribution head is supervised by TreeSHAP values from a decision tree prior, connecting the explanation to the data generating process (DGP). The resulting attributions are interpreted as a posterior over Shapley values. The PFN's distributional mechanism is transferred to feature attributions to yield calibrated explanation uncertainty. Across synthetic and real tabular benchmarks, the model recovers attributions with high fidelity ($R^2 \approx 0.94$-0.95 against exact Shapley on synthetic data), but at a small cost to accuracy. It produces calibrated attribution uncertainty, with negative log-likelihood (NLL) below the uninformative uniform baseline. An ablation trained only on the additive self-consistency constraint, without TreeSHAP supervision, degrades sharply on both attribution fidelity and uncertainty calibration.

Code is available via `https://github.com/akiraheen/PFN-Feature-Importance`

1. Introduction

Tabular data is ubiquitous in various domains, where explainability may be a requirement (Rudin, 2019). While gradient boosted trees have dominated tabular benchmarks (Shwartz-Ziv & Armon, 2021), they do not quantify uncertainty. PFNs address this by meta-training a transformer on synthetic datasets sampled from a prior, approximating Bayesian inference in a single forward pass (Müller et al., 2024) and outputting a posterior predictive distribution. TabPFN (Hollmann et al., 2025) demonstrates that this approach achieves competitive accuracy at low inference cost. Yet PFNs remain opaque. Post-hoc explanations applied after the fact are not guaranteed to reflect the model's actual computation (Lipton, 2016). This work extends PFNs to produce feature attributions as native outputs alongside predictions, representing each attribution as a distribution. This matters because a narrow posterior near zero and a wide posterior centered at zero both have the same point estimate, but carry different information about whether a feature's effect can be determined from the available data.

The research questions are:

1. Can the PFN architecture be modified so that the model produces feature importance scores along with its predictive distribution?
2. How well do the feature importance scores agree with the exact Shapley values, and are they faithful to the model's predictive behaviour?
3. Does the architectural modification required to produce importance scores change the accuracy of PFNs?
4. Can the feature attribution scores be produced as distributions so that the model expresses uncertainty about its explanations, and is this uncertainty calibrated?

2. Background

2.1. Prior-Data Fitted Networks and Tabular Foundation Models

In supervised learning, a training set $D_{\text{train}} = (x_i, y_i)$ is observed and the goal is to predict the label for a new input x. In the Bayesian framework, a prior $p(\phi)$ is defined over hypotheses Φ, where each $\phi \in \Phi$ describes a possible data generating process. A hypothesis is sampled first, followed by a synthetic dataset $D \sim p(D \mid \phi)$, defining the prior-data distribution

$$p(D) = \mathbb{E}\phi \sim p(\phi)[p(D \mid \phi)] \tag{1}$$

Predictions are obtained by marginalizing over hypotheses:

$$p(y \mid x, D) \propto \int_{\Phi} p(y \mid x, \phi) p(D \mid \phi) p(\phi), d\phi, \tag{2}$$

[1]Delft University of Technology

known as the posterior predictive distribution (PPD). Computing this integral is often expensive. Approximation techniques such as Markov Chain Monte Carlo (MCMC) and Variational Inference (VI) can be used, but MCMC is computationally costly (Andrieu et al., 2003), while VI introduces approximation bias (Wainwright & Jordan, 2008).

Prior-Data Fitted Networks (PFNs) amortize this computation by training a transformer q_θ to approximate the PPD (Müller et al., 2024). The model is trained on synthetic datasets using the equation:

$$\mathcal{L}(\theta) = \mathbb{E}\left[-\log q_\theta(y_{\text{test}} \mid x_{\text{test}}, D_{\text{train}})\right]. \tag{3}$$

Thus, Bayesian marginalization is learned during training and approximated at inference with a single forward pass.

The PFN output is a distribution over buckets, corresponding to the two labels. We denote the probability of bucket b by $q_\theta(b \mid x, D)$. We investigate whether changes in these bucket probabilities can be used to represent feature attributions.

nanoTabPFN (Pfefferle et al., 2025) provides a lightweight and educational reimplementation of the TabPFN v2 architecture. It was designed to make tabular foundation models easier to understand and pretrain. It retains the core transformer architecture and PFN training framework. This makes nanoTabPFN a practical framework for experimenting with alternative architectures, priors, and training procedures, which is what this work does.

2.2. Explainability

Explainability refers to the degree to which the causes of a model's decisions can be understood (Kim et al., 2016). Broadly, two techniques exist, post-hoc and interpretability by design. Our method is interpretable by design as the attribution is a native output of the same forward pass that produces the prediction. The prediction is read off from it additively, so no separate explainer is fit to the trained model. Of the post-hoc Shapley methods described next, TreeSHAP is used to generate training targets from the decision trees sampled from the prior. These supervise the PFN's attribution head during training. KernelSHAP and exact Shapley values are applied to the trained baseline model, nanoTabPFN at evaluation time to get reference attributions.

Shapley Values Shapley values (Shapley, 1953; Molnar, 2025) quantify each feature's average marginal contribution to a model prediction across all feature subsets:

$$\phi_j = \sum_{S \subseteq \mathcal{F} \setminus \{j\}} \frac{|S|!(F - |S| - 1)!}{F!} \big[v(S \cup \{j\}) - v(S)\big].$$

Exact computation requires evaluating 2^F coalitions. Therefore, it is often intractable, motivating approximation methods such as TreeSHAP and KernelSHAP.

TreeSHAP TreeSHAP (Lundberg et al., 2019) computes exact Shapley values in polynomial time for tree-based models by traversing each tree and marginalizing missing features using the training distribution. Its complexity is $\mathcal{O}(TLD^2)$, where T is the number of trees, L the number of leaves, and D the maximum depth. This approach assumes features are marginalized independently.

KernelSHAP For black-box models, KernelSHAP (Lundberg & Lee, 2017) estimates Shapley values by fitting a weighted linear regression over sampled feature coalitions. It is model-agnostic, but introduces variance from both coalition sampling and estimating the value of each coalition from a background dataset.

3. Related Work

KernelICL Miftachov et al. (2026) replaces the prediction head of tabular foundation models with explicit kernel functions, so that the predictions are weighted averages of training labels with readable weights. This gives instance level explainability, identifying which training examples influenced a prediction the most. Unlike KernelICL, which computes just instance level importance from training examples, the proposed method targets feature level importance as well, to identify which input features drive predictions across the dataset as a whole.

Generalised additive models (GAMs) represent an approach to interpretability by design by decomposing predictions into a sum of per-feature shape functions that are directly readable (Caruana et al., 2015). **GAMFormer** Mueller et al. (2026) brings this to tabular foundation models by running a separate transformer per feature, estimating GAM shape functions via in-context learning in a single forward pass. This makes each feature's contribution readable without post-hoc explainability. Like GAMFormer, our model is interpretable by design. It learns in context and outputs per-feature contributions that sum to the prediction. Unlike GAMFormer, it attends across features, so each $\phi_j(x)$ is a Shapley value specific to an instance that depends on the whole input. Interactions are therefore shared among the features involved. The contributions are also trained directly on TreeSHAP targets and predicted as distributions.

ShapPFN Sena & Azevedo (2026) adds a baseline decoder and a per-feature Shapley decoder to a PFN. The logit is the baseline plus the feature contributions. This gives predictions and additive attributions in a single forward pass. Attribution quality comes only from a ViaSHAP-style consistency loss (Alkhatib et al., 2025), which compares masked coalition predictions with the additive estimate. Our model has the same additive structure but differs in two ways. First,

it supervises the attributions directly with TreeSHAP values of the trees that generated the prior data, rather than relying on self-consistency alone. Second, it predicts a distribution over each ϕ_j rather than a point estimate. The spread of that distribution captures uncertainty in the attribution given the context.

Evaluating feature attributions is difficult because the true attributions for real data are unknown. So most evaluations compare against approximations such as KernelSHAP. **XAI-Bench** Liu et al. (2021) instead provides synthetic binary classification datasets whose feature distribution (a multivariate Gaussian) and labelling function are fully specified. Because the distribution is Gaussian, the conditional distribution of any feature subset is available in closed form. Shapley values can therefore be estimated by sampling from the true conditional distribution and not a background. This gives a ground-truth reference that real data cannot provide. We use three of its datasets with linear, piecewise and nonlinear decision boundaries. On each, we compute these Shapley values for the trained PFN's own predictions. We use them as the reference for two checks, how accurate the model's point attributions are, and how well calibrated its predicted ϕ distributions are (measured by NLL). Their known relevant and irrelevant features also support dummy-feature tests.

4. Methodology

We extend nanoTabPFN (Pfefferle et al., 2025) so that it produces both predictions and per-feature attributions with uncertainty estimates in a single forward pass. The prediction is derived from it through the additive structure (Section 4.2).

Each feature attribution is represented as a distribution over signed values. Since a PFN is an amortized approximation to Bayesian inference, it outputs a PPD over the label. This mechanism is applied to the prediction, and now to the feature attributions. So, they are reported as a posterior predictive distribution with uncertainty estimates, given the observed data.

The attributions are supervised with per-instance Shapley values computed by TreeSHAP on the decision trees that generate the prior. Thus, the explanation is based on the data-generating process.

4.1. Prior Generation

The model is trained on synthetic datasets generated from a decision tree prior adapted from TabForestPFN (den Breejen et al., 2025).

For each dataset, a decision tree of random depth (1-5) is fit to randomly sampled features with random binary labels. This produces a random axis-aligned partition of the feature space. The fitted tree then assigns labels to a new set of points with independent features, as shown in Figure 9. The tree is fit before the dataset is sampled because the relationship between the features and the target is defined independently of the dataset. The model's task is to recover this relationship.

Since the features are independent, $p(x_{\bar{S}}) = p(x_{\bar{S}} \mid x_S)$, the interventional and conditional Shapley values coincide. For each synthetic dataset, we compute the interventional TreeSHAP values of the generating tree on every row, using that dataset's training split as the background distribution. TreeSHAP (Lundberg et al., 2019) is used for three reasons. First, it computes Shapley values exactly from the tree structure, without coalition sampling. The only approximation is the finite background sample used for the marginal feature distribution. Second, since the labels are produced by the tree itself, TreeSHAP explains exactly the function that generated the data. The feature attributions are those of the labelling mechanism. Third, TreeSHAP produces signed, per-instance, per-feature attributions, which are the targets used to supervise the contribution head. Implementation details can be found in Appendix A.

4.2. Architecture

The backbone is nanoTabPFN (Pfefferle et al., 2025). A feature encoder normalizes each feature value using the mean and standard deviation of the training data, clips against outliers, and projects the resulting scalar to a d-dimensional embedding via a shared linear layer. A target encoder applies the same normalization, clipping, and linear projection to the targets. Test-row label slots are filled with the mean of the training labels before embedding. The transformer blocks alternate attention over features and over datapoints. Training rows attend to each other, while test rows attend only to training rows. Training rows do not attend to test rows and test rows do not attend to each other. Each test prediction depends only on the training context and the test row itself. The single classification decoder of nanoTabPFN is replaced by two heads, a baseline head and a contribution head. The architecture is shown in Figure 1.

Baseline head A two-layer multilayer perceptron (MLP) maps the mean of the training target embeddings to a per-class scalar baseline $\mathrm{base}(x)$ to represent $\mathbb{E}[f(X)]$.

Contribution head A small MLP is applied independently to each test-row feature embedding, with the same weights for every feature. For each feature j and each class, it outputs a distribution over $B = 32$ buckets of signed contribution values. The bucket mean is the point attribution $\phi_j(x)$ and its spread is the uncertainty. Bucket edges are set from the quantiles of the prior's TreeSHAP values. At infer-

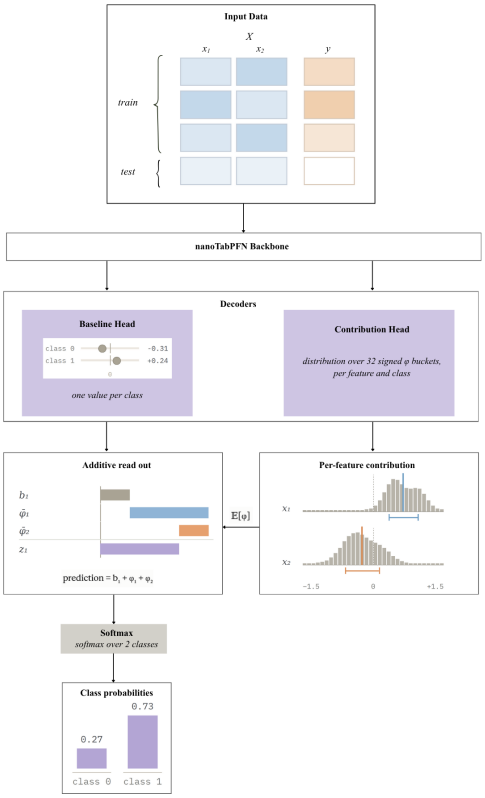


Figure 1. Modified nanoTabPFN architecture with heads for classification and per-feature attributions as outputs.

ence, the bucketed output approximates the posterior over feature j's contribution given the training data. This is the distribution of that feature's attribution across the labelling functions in the prior, weighted by their probability.

Additive prediction The logit is

$$f_\theta(x) = \text{base}(x) + \sum_{j=1}^{F} \phi_j(x),$$

applied separately for each class, and the predicted class probabilities are its softmax. So the explanation and the prediction cannot disagree by design. This guarantees that the attributions are consistent with the prediction. Whether each individual ϕ_j matches the feature's Shapley value is what the attribution losses below encourage.

4.3. Training objective

The model is trained on three losses per synthetic dataset:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda\,\mathcal{L}_{\text{TreeSHAP}} + \mu\,\mathcal{L}_{\text{ViaSHAP}}$$

$\mathcal{L}_{\text{CE}}$ is the cross-entropy classification loss on the model's prediction. $\mathcal{L}_{\text{TreeSHAP}}$ supervises the contribution head with TreeSHAP attributions from the prior, using a bucketed (bar-distribution) cross-entropy. $\mathcal{L}_{\text{ViaSHAP}}$ enforces consistency between the attributions and the model's own predictions on masked inputs. TreeSHAP supervision pins down each ϕ_f against a known target, while the classification loss only constrains their sum. ViaSHAP enforces additive self-consistency (components sum to the prediction) with no external target.

Classification loss $\mathcal{L}_{\text{CE}}$ is the standard cross-entropy between $\text{softmax}(f_\theta(x))$ and the label of each test row. Because the prediction is additive, its gradient flows through every ϕ_j, forcing the contributions to be collectively predictive.

TreeSHAP loss The contribution head is supervised by the per-instance, per-feature TreeSHAP values of the generating tree (Section 4.1). Each scalar target is mapped to its bucket index, and the head is trained with cross-entropy.

A point-estimate loss would discard uncertainty information. Training with cross-entropy against bucket targets instead incentivizes the head to output calibrated probabilities over attributions. Here, calibrated means that if the head assigns probability p to a bucket, the true TreeSHAP value falls in that bucket a fraction p of the time across datasets in the prior. This is the same mechanism by which PFNs approximate the PPD over labels. The spread of the attribution distribution reflects posterior uncertainty over which labelling function generated data.

ViaSHAP loss $\mathcal{L}_{\text{ViaSHAP}}$ (Alkhatib et al., 2025) penalizes disagreement between the model's predictions on masked inputs and its own additive approximation. Efficiency, $\sum_{j=1}^{F} \phi_j(x) = f_\theta(x) - \text{base}(x)$, already holds by construction. ViaSHAP targets the property that characterises Shapley values. For any coalition s of features kept, $\text{base}(x) + \sum_{j \in s} \phi_j(x)$ should match the model's expected prediction when the features outside s are marginalised out. With Shapley-kernel weights w_s, the Shapley values are a solution of this weighted least-squares fit when the baseline equals the model's prediction with all features masked. $\mathcal{L}_{\text{ViaSHAP}}$ therefore encourages each ϕ_j towards the Shapley value of the model's own prediction.

For each synthetic dataset, S coalitions s are sampled. A size is drawn uniformly from $\{1, \ldots, F-1\}$ and that many features are kept at random. For each test row, the removed features are replaced by values from K randomly chosen training rows (background). Let x^{s_k} be the k-th such

masked version of x. The model's masked prediction is:

$$\hat{f}_\theta(x^s) = \frac{1}{K}\sum_{k=1}^{K} f_\theta(x^{s_k})$$

and its additive approximation is:

$$\bar{f}_\theta(x^s) = \text{base}(x) + \sum_{j\in s}\phi_j(x),$$

where $\phi_j(x)$ is the bucket mean on the unmasked input. The loss is the mean squared error:

$$\mathcal{L}_{\text{ViaSHAP}} = \frac{1}{S}\sum_{s\in\mathcal{S}} w_s\big(\bar{f}_\theta(x^s) - \hat{f}_\theta(x^s)\big)^2$$

weighted by the Shapley-kernel

$$w_s = \frac{F-1}{\binom{F}{|s|}\,|s|\,(F-|s|)}.$$

The weight w_s is largest for very small and nearly complete coalitions, where a single feature's effect is easiest to isolate. Both attribution losses are held at zero for the first 1200 steps and phased in together afterwards, since a prediction can only be meaningfully decomposed once the model can make it.

At inference, given the training context $(X_{\text{train}}, y_{\text{train}})$ and a test row x^*, a single forward pass returns the predicted class probabilities and a per-feature, per-class attribution distribution.

5. Experiments

5.1. TabArena Benchmark

5.1.1. Setup

We evaluate predictive accuracy of binary classification and alignment with KernelSHAP explanation of 8 tasks of the TabArena benchmark (Erickson et al., 2025), spanning 4 to 20 features and 748 to 10000 instances. Each PFN model is evaluated under 8-fold cross-validation over 3 seeds, with folds shared across methods, and scored by ROC AUC. Per-dataset scores are the mean over folds and seeds. Baselines are taken from TabArena's published results using the default configuration variant of each method. We report the mean AUC and standard error over the eight datasets. Per-dataset estimates were stable across the three seeds.

5.1.2. Classification Accuracy

The tabular foundation models form the top group (TabPFNv2 0.814, TabICL 0.813, CatBoost 0.811), followed by the gradient-boosted trees and classical baselines. Our model, using the additive architecture together with the TreeSHAP loss and ViaSHAP loss, attains a mean AUC of 0.744. The additive architecture with only the TreeSHAP loss $\mathcal{L}_{TreeSHAP}$ has an AUC of 0.750, and with only the ViaSHAP loss $\mathcal{L}_{ViaSHAP}$ has an AUC of 0.762. The additive architecture alone achieves 0.740, as shown in Figure 2. The model with only ViaSHAP loss edges out the other three variants, suggesting the two losses trade off slightly against predictive accuracy.

5.1.3. Evaluation of Explanations

To evaluate alignment of attribution scores on real datasets, we measure the agreement between each model's per-feature attributions and the attributions obtained by running KernelSHAP on that same model's own predictions. On real data there is no known DGP. So exact or closed form Shapley is not available. Thus KernelSHAP, a model-agnostic post-hoc method is used.

For each dataset, 150 rows are drawn as the background for KernelSHAP, and 50 test rows are explained. The experiment is repeated over three seeds. Agreement is measured by cosine similarity, Spearman rank correlation and R^2.

Figure 3 reports attribution agreement (R^2) with KernelSHAP across the eight datasets. ShapPFN attains the strongest agreement ($R^2 = 0.873 \pm 0.026$), consistent with its richer training prior. Among our additive variants, the full attribution loss (TreeSHAP + ViaSHAP) yields the closest agreement to KernelSHAP ($R^2 = 0.772 \pm 0.043$). Training with the ViaSHAP consistency term alone, comes close behind ($R^2 = 0.765 \pm 0.053$). Supervising with the TreeSHAP loss alone, without the ViaSHAP consistency term, recovers most but not all of this faithfulness ($R^2 = 0.678 \pm 0.070$). Training without either attribution loss collapses agreement ($R^2 = 0.164 \pm 0.070$).

Cosine similarity and Spearman ρ show the same pattern as R^2 across all eight datasets (Appendix Table 5). All four variants and ShapPFN deliver a similar computational speedup over KernelSHAP, roughly 560–730$\times$ on average (Table 5). The single-pass ϕ head has essentially the same forward cost regardless of which attribution loss supervised it.

So from Figure 3 and Figure 2, the ViaSHAP consistency term accounts for most of the faithfulness gain over the unsupervised additive decomposition. The TreeSHAP loss alone still recovers a substantial share of this gain, but less than ViaSHAP does on its own. This gain does not come at a cost to accuracy either.

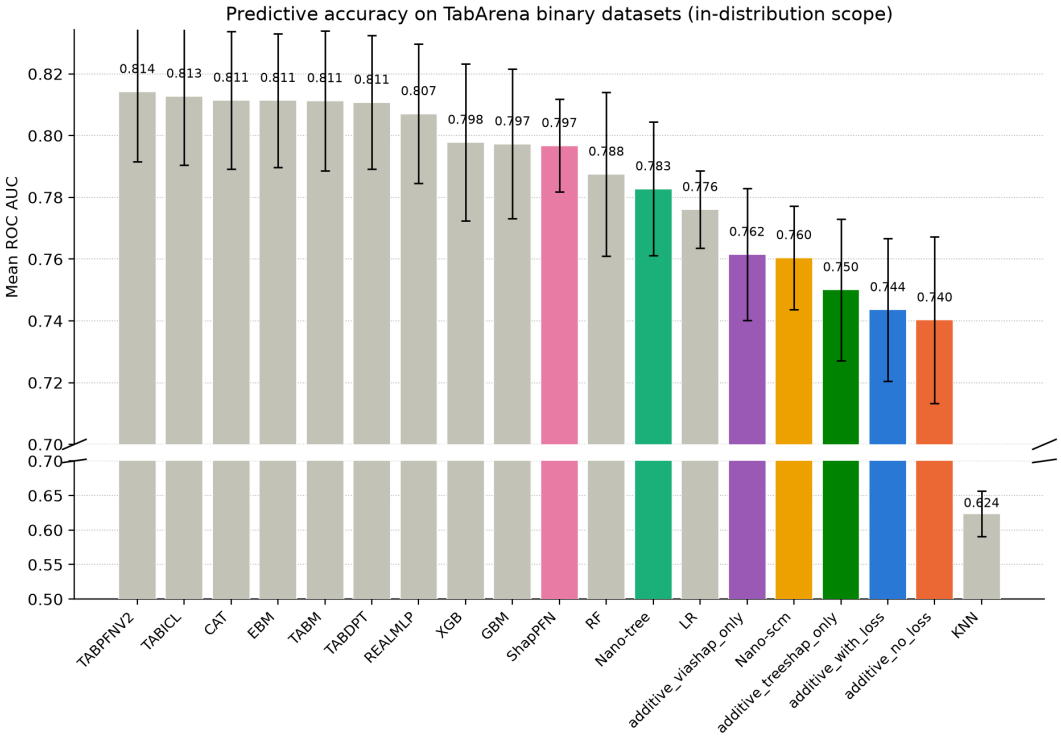


Figure 2. Predictive accuracy on the eight in-distribution binary TabArena tasks (8-fold ROC AUC, three seeds). Mean AUC and standard error are across datasets. Average rank is over 19 methods. Colored bars denote loss-ablated model variants, the vanilla NanoTabPFN baselines (Nano-tree, Nano-scm) and ShapPFN. Gray bars are TabArena's cached baselines.

5.2. Evaluation on Synthetic Datasets

5.2.1. Setup

We evaluate classification accuracy and attribution quality on eight synthetic datasets described in Appendix D.1, with known ground-truth feature importance. Loss ablation variants of the additive PFN were trained with both TreeSHAP and ViaSHAP attribution losses, with neither, with the TreeSHAP loss only and with the ViaSHAP loss only. This was done alongside ShapPFN and vanilla NanoTabPFN (tree prior).

5.2.2. Classification Accuracy

For each (dataset, model) pair, we run 10 independent train/test splits ($N = 300$, 50% test fraction) and evaluate ROC-AUC and accuracy at a 0.5 threshold on the held-out split, reporting the mean $\pm$ SE over seeds and an aggregate averaged across datasets.

All models achieve similarly high accuracy, with aggregate ROC-AUC ranging from 0.966 to 0.976 (Table 1). ShapPFN attains the highest aggregate AUC and accuracy.

Predictive accuracy does not distinguish the loss-ablation variants from each other or from vanilla NanoTabPFN, as seen in Table 6. Attribution loss supervision shapes explanation quality, not classification accuracy. Since the prediction depends only on $\sum_j \phi_j$, the classification loss constrains the sum but not the individual ϕ_j values, which only the attribution loss $\mathcal{L}_{TreeSHAP}$ and $\mathcal{L}_{ViaSHAP}$ supervise. The shared accuracy drop on the piecewise constant dataset (AUC $0.84-0.88$) reflects its harder decision boundary, independent of model.

5.2.3. Evaluation of Explanations: Shapley Alignment

Using the models and datasets described in Section 5.2.1, we compare six methods' point-estimate feature attributions against exact Shapley values. The additive PFN variants are each explained with their native $E[\phi]$, and ShapPFN with its native point estimate of ϕ. NanoTabPFN, which does not have a native attribution head, is instead explained via KernelSHAP on its own predictions. Each method is scored against the exact Shapley value of its own predictions (10 seeds, 60 explained test instances per seed). We report R^2 (Figure 4).

Five of the six models achieve high R^2 across nearly every dataset ($0.89 - 0.99$). These models include the three

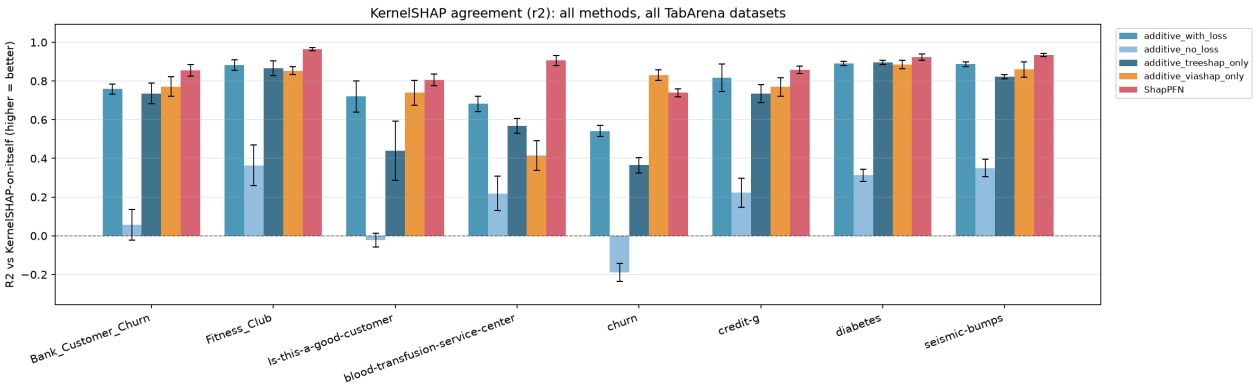


Figure 3. Agreement between each model's attributions and KernelSHAP across 8 TabArena datasets (mean ± SE over 3 seeds), for R^2.

Table 1. Predictive accuracy comparison across three models on all eight synthetic datasets (mean ± SE over 10 seeds, $N = 300$ samples per split).

Dataset	additive_with_loss		ShapPFN		NanoTabPFN-tree	
	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy
single_relevant	1.000 ± 0.000	0.996 ± 0.001	1.000 ± 0.000	0.980 ± 0.003	1.000 ± 0.000	0.994 ± 0.003
two_unequal	0.980 ± 0.003	0.904 ± 0.010	0.989 ± 0.002	0.923 ± 0.006	0.987 ± 0.002	0.937 ± 0.005
all_relevant	0.958 ± 0.003	0.853 ± 0.010	0.985 ± 0.002	0.929 ± 0.007	0.960 ± 0.005	0.882 ± 0.011
tie	0.979 ± 0.004	0.841 ± 0.018	0.989 ± 0.002	0.930 ± 0.009	0.985 ± 0.002	0.928 ± 0.007
many_dummies	0.979 ± 0.002	0.899 ± 0.008	0.985 ± 0.002	0.921 ± 0.006	0.980 ± 0.004	0.909 ± 0.008
gaussian_linear	0.976 ± 0.006	0.907 ± 0.012	0.984 ± 0.005	0.928 ± 0.012	0.973 ± 0.006	0.919 ± 0.010
gaussian_piecewise_constant	0.881 ± 0.010	0.783 ± 0.015	0.879 ± 0.008	0.811 ± 0.008	0.844 ± 0.011	0.768 ± 0.012
gaussian_nonlinear_additive	0.978 ± 0.007	0.905 ± 0.015	0.994 ± 0.002	0.953 ± 0.011	0.998 ± 0.001	0.968 ± 0.009
Average	$\mathbf{0.966 \pm 0.013}$	$\mathbf{0.886 \pm 0.022}$	0.976 ± 0.014	0.922 ± 0.017	0.966 ± 0.018	0.913 ± 0.024

additive-PFN loss variants (TreeSHAP only, ViaSHAP only, or both), ShapPFN, and vanilla NanoTabPFN with KernelSHAP. The variant trained with neither attribution loss is the exception, collapsing to negative R^2 on 7 of 8 datasets and reaching $R^2 = -4.6$.

The loss and not the additive architecture alone, is what makes the point estimate accurate. On synthetic, independent-feature data ViaSHAP slightly outperforms TreeSHAP alone ($R^2 = 0.978$ vs. 0.944) and even edges out the checkpoint trained with both losses combined (0.946). This is more prominent in real, correlated data. ViaSHAP alone beats TreeSHAP alone ($R^2 = 0.765$ vs. 0.678, Section 5.1.3) and comes close to the full-loss checkpoint (0.772).

TreeSHAP only's accuracy drops sharply when moving from synthetic to real data ($0.944 \rightarrow 0.678$), while ViaSHAP-only's accuracy drops only slightly ($0.978 \rightarrow 0.765$). Direct per-feature value supervision (TreeSHAP) is effective when features are independent but generalizes poorly once they correlate. The self-consistency constraint (ViaSHAP) is more robust to that shift. This mirrors ShapPFN, whose only supervision is a self-consistency loss, it is the strongest of all six methods on real data ($R^2 = 0.873$). ViaSHAP supervision degrades less under real-world feature correlation than direct TreeSHAP value supervision does.

5.2.4. Evaluation of Explanations: Remove and Retrain (ROAR)

We test whether each model's native feature-importance ranking is faithful to its own predictions using ROAR (Hooker et al., 2019). Features are dropped one at a time, most important first and least important first, and AUC is tracked. A faithful ranking should show a large gap between the two curves. AUC collapses quickly when important features are removed, staying nearly unchanged when unimportant ones are removed. We run this for the four additive-PFN loss variants and ShapPFN across all eight synthetic datasets (10 seeds each).

The four loss supervised methods behave almost identically, as shown in Figure 5. The mean gap ranges from 0.38 to 0.40. Removing the most important features first causes the AUC to collapse to near-chance levels (≈ 0.55) within one or two removal steps. This indicates that these features cap-

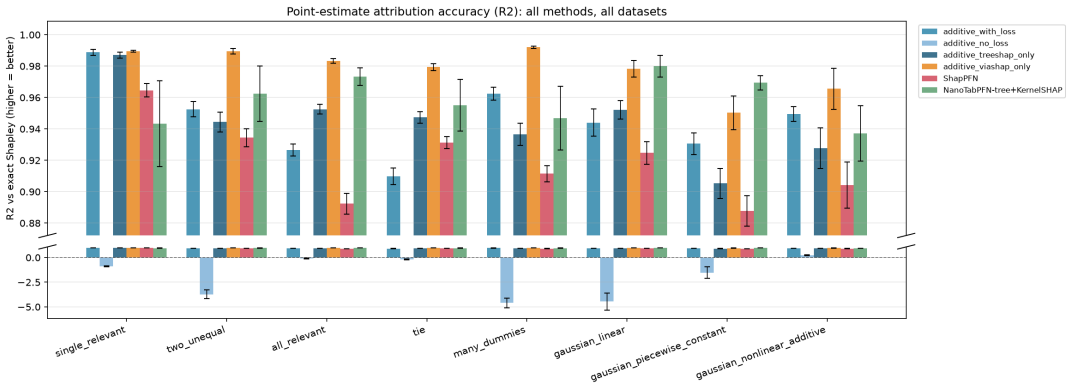


Figure 4. Point-estimate attribution comparison (R^2 vs. exact Shapley, higher is better) for all six methods across all eight synthetic datasets.

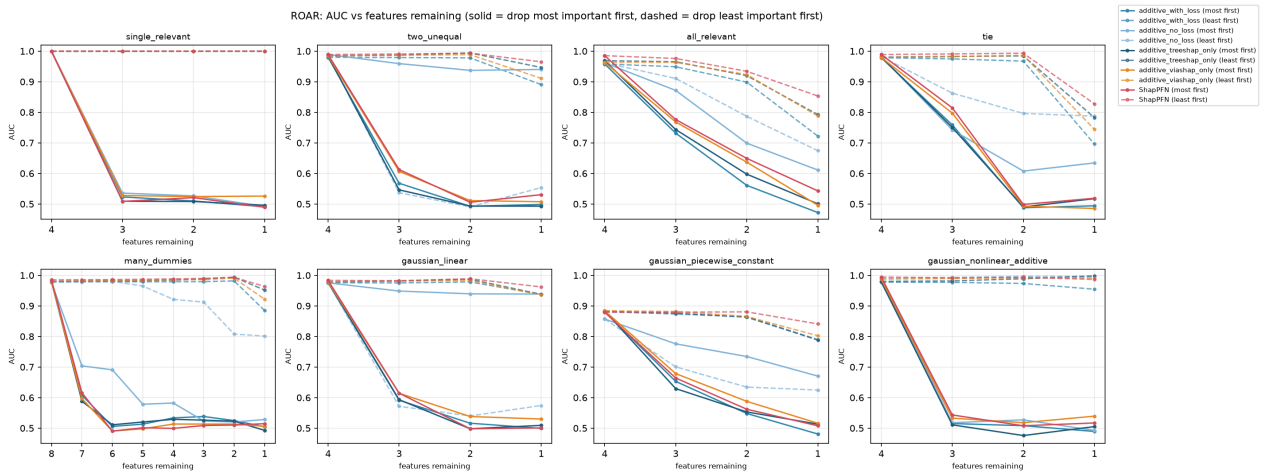


Figure 5. ROAR curves per dataset: mean AUC vs. number of features remaining, averaged over 10 seeds, for the four loss-ablated additive-PFN variants and ShapPFN. Solid lines drop the most important features first. Dashed lines drop the least important features first.

ture most of the predictive signal. In contrast, removing the least important features has little effect on datasets with genuine dummy features. On synthetic datasets where nearly every feature carries signal, removing the least important features still reduces AUC by approximately 0.18-0.22.

Any attribution loss supervision, TreeSHAP or ViaSHAP, alone or combined, produces a feature importance ranking the model's own predictions faithfully depend on. The specific loss choice barely matters for this check. Without attribution supervision, the additive architecture's importance ranking is occasionally actively misleading, consistent with its point estimate and calibration results.

5.3. Uncertainty Calibration of Attributions

Setup The additive readout predicts each feature attribution ϕ_j as a distribution. We ask whether the predicted distribution is informative. Does it place high density on the KernelSHAP reference attribution and does its spread increase when the attribution is harder to predict? We evaluate distributional accuracy using negative log-likelihood (NLL) and uncertainty-error alignment using the predicted standard deviation.

Each attribution head outputs probabilities over $K{=}32$ attribution buckets. For a bucket of width w_k containing the reference attribution ϕ_j^* with predicted probability mass p_k, the prediction is interpreted as $q(\phi) = p_k/w_k$ within the bucket. The per-cell NLL is therefore

$$\mathrm{NLL} = -\log q(\phi_j^*) = -\log p_k + \log w_k \qquad (4)$$

We use NLL$= 0$ as the reference for an uninformative uniform distribution. Negative values indicate higher density at the reference attribution than this baseline, while positive values indicate lower density. The reference attribution ϕ_j^* is the mean of five independent KernelSHAP estimates of

the model's own predictions.

We report NLL for four loss-ablation models across all eight synthetic datasets, averaged over ten data-generation seeds. We use per (instance, feature) cells across all datasets to examine the NLL distribution and the relationship between predicted uncertainty $\sigma = \mathrm{std}(\phi_j)$ and reference attribution error $|\mathbb{E}[\phi_j] - \phi_j^*|$.

Results Figure 6 shows that the two models trained with TreeSHAP value supervision achieve nearly identical mean NLL (-2.62 and -2.61) and outperform the model without TreeSHAP supervision on every dataset. That is the only model with positive mean NLL, indicating that its predicted distributions are less concentrated around the reference attribution than the uninformative uniform baseline. The model trained with ViaSHAP supervision alone (no TreeSHAP loss) reaches an aggregate mean NLL of -1.15, negative on every one of the eight datasets and consistently better than the no-supervision model, but worse than both the TreeSHAP supervised models.

Figure 7 shows that this difference is not driven only by the mean. The two TreeSHAP supervised models have concentrated negative-NLL distributions, with little mass above zero. In contrast, the model with neither TreeSHAP nor ViaSHAP supervision has a broader distribution reaching NLL$= 9.53$, indicating individual cases where very little density is assigned to the reference attribution. The ViaSHAP-only model's distribution is intermediate. Its mean (-1.19) and maximum (1.25) both sit between the tight TreeSHAP supervised distributions and the wide, heavy-tailed no supervision distribution.

Figure 8 shows a positive association between predicted uncertainty and reference attribution error. The Spearman correlations are $+0.469$ for the model trained with both losses, $+0.574$ for the model trained with TreeSHAP supervision alone, and $+0.412$ for the model trained without TreeSHAP or ViaSHAP. Thus, higher predicted uncertainty generally corresponds to larger attribution errors, with the strongest association observed for the TreeSHAP only model. The ViaSHAP-only model also shows a positive but comparatively weak association ($+0.328$).

Inference Together, the experiments show that TreeSHAP supervision substantially improves the quality and informativeness of the predicted attribution distributions. The additive self-consistency objective constrains base$(x) + \sum_j \phi_j$, so it does not uniquely determine the individual feature attributions or their uncertainty. TreeSHAP supervision provides a per-feature reference, producing distributions that assign more density to the reference values. The NLL results for the two TreeSHAP-supervised models further indicate that most of the benefit comes from per-feature supervision. The ViaSHAP-only model indicates that per-feature supervision alone is not sufficient to reproduce the TreeSHAP-supervised models' calibration.

6. Discussion and Limitations

Representing feature attribution values ϕ_j as a distribution provides additional information for practical applications. Point attribution methods provide only a single estimate and therefore cannot quantify uncertainty. Two features may have similar point estimates while having substantially different predictive distributions. Thus, a small attribution may indicate either a confidently small contribution or a highly uncertain contribution that averages to a small value. Distributions make this distinction explicit and reveal when a feature's contribution is uncertain. This could have potential downstream implications for decision making and feature selection.

A feature attribution could either explain the model or recover the underlying data generating process (DGP). This distinction is what makes the ground truth attribution meaningful. If the goal is to explain the model, then the attribution target should be computed from the model being explained. If the goal is to recover the underlying feature contributions, the reference should be derived from the DGP. Our work grounds the attribution head in the DGP (TreeSHAP on the same tree that generated the labels), giving exact per-feature targets. Evaluation then explains the model itself. On real data, the DGP is no longer available. DGP recovery teaches decomposition, and model-explanation tests whether it generalizes. Since TreeSHAP supervision is a direct target derived from this prior's tree-based structure and ViaSHAP's interventional background sampling does not generally capture feature dependencies, it introduces the limitation that both losses are reliable only under feature independence.

Other limitations are as follows. Attributions can appear fair and auditable without being verified outside the conditions they were tested under (here, feature independence). Easily generated per-feature attributions also risk enabling reverse engineering of a model's weighted features, aiding adversarial manipulation. The environmental cost of all training runs, ablations, and experiments is also a limitation.

7. Conclusion and Future Work

This work extends nanoTabPFN to produce feature attributions as a native distribution alongside its predictions, using a baseline head and a per-feature contribution head. Predictive accuracy is largely unaffected by this change, and varies little across loss-ablation variants, since the classification loss constrains the sum of the attributions but not their individual values. The attributions are accurate when the

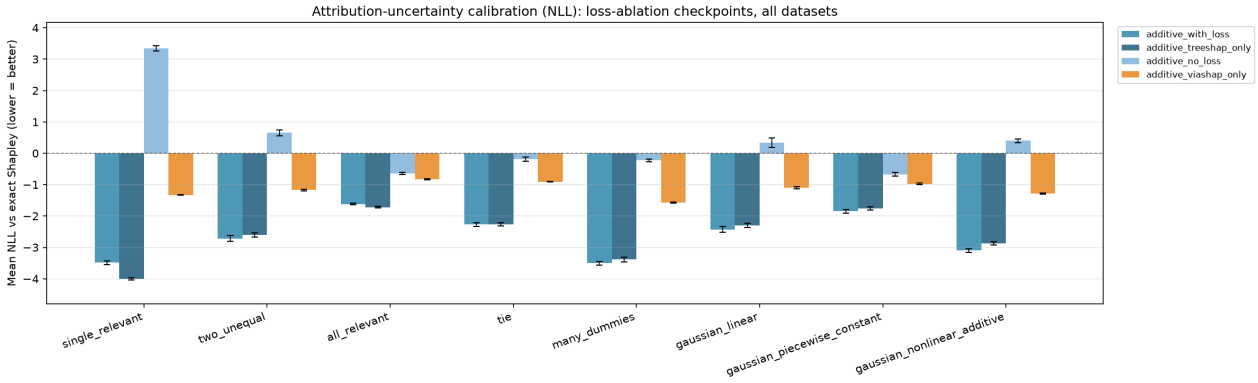


Figure 6. Mean attribution NLL by dataset and model (error bars: standard error over ten seeds). Lower is better. The dashed line marks the uniform distribution, NLL$= 0$.

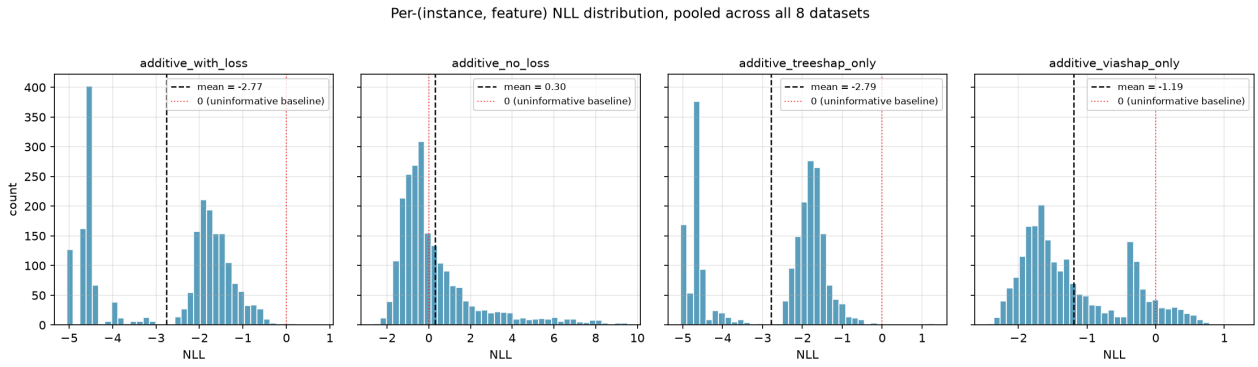


Figure 7. Distribution of per-(instance, feature) attribution NLL, across all eight synthetic datasets. The dashed line marks each model's mean. The dotted line marks the uniform distribution, NLL$= 0$.

loss terms are present. They agree closely with exact Shapley values on synthetic data and with KernelSHAP on real data. ROAR confirms the model's own predictions depend on the features its attributions rank highest. The predicted attribution distributions are calibrated, but only under direct per-feature (TreeSHAP) supervision. Self-consistency (ViaSHAP) alone gives weaker calibration, and without either loss the model is miscalibrated on average. Overall, TreeSHAP supervision drives point-estimate accuracy and calibration and ViaSHAP drives robustness under real-data feature correlation, at a small cost to predictive accuracy.

Future directions include modifying the prior to generate correlated features and replacing interventional TreeSHAP targets with causal Shapley values that respect correlation. This can also be extended beyond binary classification to test whether calibrated attributions hold across other tasks.

References

Alkhatib, A., Bresson, R., Boström, H., and Vazirgiannis, M. Prediction via shapley value regression, 2025.

Andrieu, C., de Freitas, N., Doucet, A., and Jordan, M. I. An introduction to mcmc for machine learning. *Machine Learning*, 50(1):5–43, Jan 2003. ISSN 1573-0565.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., and Elhadad, N. Intelligible models for HealthCare: Predicting pneumonia risk and hospital 30-day readmission. pp. 1721–1730. Association for Computing Machinery, 2015.

Defazio, A., Yang, X., Mehta, H., Mishchenko, K., Khaled, A., and Cutkosky, A. The road less scheduled. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 9974–10007. Curran Associates, Inc., 2024.

den Breejen, F., Bae, S., Cha, S., and Yun, S.-Y. Fine-tuned in-context learning transformers are excellent tabular data classifiers, 2025.

Erickson, N., Purucker, L., Tschalzev, A., Holzmüller, D., Desai, P. M., Salinas, D., and Hutter, F. Tabarena: A

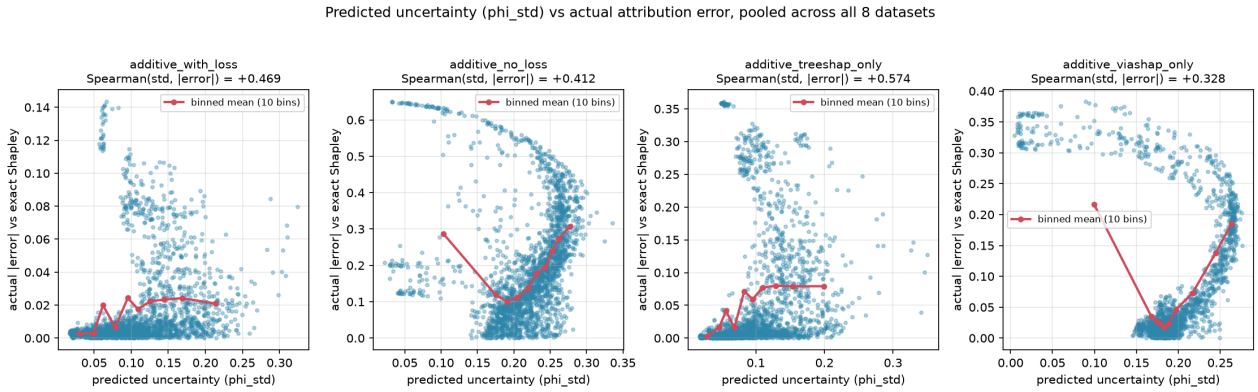


Figure 8. Predicted uncertainty against reference attribution error, across all eight synthetic datasets. The red line shows the mean error within ten bins of predicted uncertainty.

living benchmark for machine learning on tabular data, 2025.

Fonseca, J. and Stoyanovich, J. Explainerpfn: Towards tabular foundation models for model-free zero-shot feature importance estimations, 2026.

Frye, C., Rowat, C., and Feige, I. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability, 2021.

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3681–3688, Jul. 2019.

Heskes, T., Sijben, E., Bucur, I. G., and Claassen, T. Causal shapley values: Exploiting causal knowledge to explain individual predictions of complex models, 2020.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations 2023*, 2023.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 01 2025. doi: 10.1038/s41586-024-08328-6.

Hooker, S., Erhan, D., Kindermans, P.-J., and Kim, B. A benchmark for interpretability methods in deep neural networks, 2019.

Janzing, D., Minorics, L., and Blöbaum, P. Feature relevance quantification in explainable ai: A causal problem, 2019.

Jethani, N., Sudarshan, M., Covert, I., Lee, S.-I., and Ranganath, R. Fastshap: Real-time shapley value estimation, 2022.

Kim, B., Khanna, R., and Koyejo, O. Examples are not enough, learn to criticize! criticism for interpretability. pp. 2288–2296. Curran Associates Inc., 2016.

Lipton, Z. C. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016.

Liu, S.-Y., Cai, H.-R., Zhou, Q.-L., Yin, H.-H., Zhou, T., Jiang, J.-P., and Ye, H.-J. Talent: A tabular analytics and learning toolbox. *Journal of Machine Learning Research*, 26(226):1–16, 2025.

Liu, Y., Khandagale, S., White, C., and Neiswanger, W. Synthetic benchmarks for scientific research in explainable machine learning, 2021.

Lundberg, S. and Lee, S.-I. A unified approach to interpreting model predictions, 2017.

Lundberg, S. M., Erion, G. G., and Lee, S.-I. Consistent individualized feature attribution for tree ensembles, 2019.

Miftachov, R., Charron, B., and Valentin, S. Interpretable tabular foundation models via in-context kernel regression, 2026.

Molnar, C. *Interpretable Machine Learning*. 3 edition, 2025. ISBN 978-3-911578-03-5.

Mueller, A., Siems, J., Nori, H., Salinas, D., Zela, A., Caruana, R., and Hutter, F. Gamformer: Bridging tabular foundation models and interpretable machine learning, 2026.

Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference, 2024.

Pfefferle, A., Hog, J., Purucker, L., and Hutter, F. nanotabpfn: A lightweight and educational reimplementation of tabpfn. *arXiv preprint arXiv:2511.03634*, 2025.

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. Tabicl: A tabular foundation model for in-context learning on large data, 2025.

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. Tabiclv2: A better, faster, scalable, and open tabular foundation model, 2026.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, May 2019.

Sena, L. B. T. R. and Azevedo, F. G. Real-time explanations for tabular foundation models, 2026.

Shapley, L. S. A value for n-person games. In Kuhn, H. W. and Tucker, A. W. (eds.), *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematical Studies*, pp. 307–317. Princeton University Press, Princeton, NJ, 1953.

Shwartz-Ziv, R. and Armon, A. Tabular data: Deep learning is not all you need, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. pp. 6000–6010. Curran Associates Inc., 2017.

Wainwright, M. J. and Jordan, M. I. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305, 12 2008.

Ye, H.-J., Liu, S.-Y., Cai, H.-R., Zhou, Q.-L., and Zhan, D.-C. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024.

A. Prior Generation

Each synthetic dataset is generated as follows. The number of features F is drawn from $[2, 20]$, biased toward smaller feature counts. The dataset size is capped at $\min(150, \lfloor 75000/F \rfloor)$ rows.

A DecisionTreeClassifier with random maximum depth drawn from a uniform distribution of $[1, 5]$ is fit on N standard normal points with random binary labels. Then fresh $N \times F$ standard normal matrix is then sampled and labelled by this tree, so the labelling function is independent of the observed points. The train/test split position is drawn uniformly from $[0.1N, 0.9N]$.

A random subset of columns is discretized following (den Breejen et al., 2025). Selected columns are quantile-transformed and bucketed. The transform is applied independently per column. Interventional TreeSHAP is computed on all rows using the training data as background, resulting in signed attributions $\phi \in \mathbb{R}^{N \times F}$. Datasets are stored in a compressed HDF5 file.

Figure 9 shows an example dataset with two features in the prior.

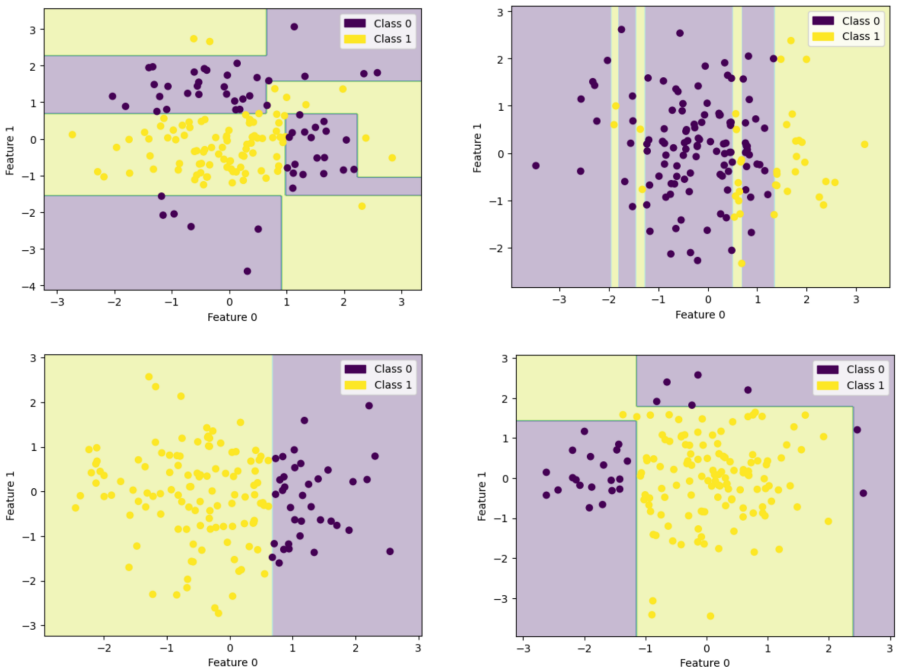


Figure 9. Generated data (prior). Decision boundaries are orthogonal, and there is no feature correlation. Every box is a generated dataset with 150 rows, 2 features, and 2 classes.

B. Hyperparameter Optimization

B.1. Number of Training Steps

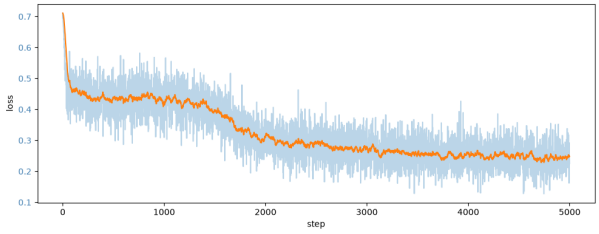
Training step count was optimized on the model with additive architecture and decision tree prior, where TreeSHAP weight $= 0$, viaSHAP weight $= 0$. The model was trained using the AdamW Schedule-Free optimizer (Defazio et al., 2024) with learning rate 10^{-4}, weight decay 0, and gradient clipping at max norm 1.0. Loss was recorded at every optimization step and a smoothed trace was computed via exponential moving average (EMA) with decay $\alpha = 0.95$:

$$\mathcal{L}_t^{\text{EMA}} = 0.95 \cdot \mathcal{L}_{t-1}^{\text{EMA}} + 0.05 \cdot \mathcal{L}_t.$$

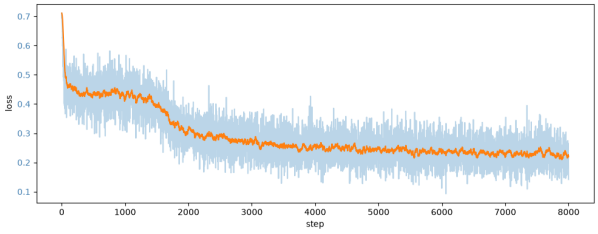
Five runs were conducted using $5000, 8000, 10000, 15000$, and 20000 training steps.

The loss curve followed a consistent pattern across the five runs as shown in Figure 10. Descent to ≈ 0.40–0.45 within the first 500 steps, then gradual decrease through ≈ 0.30 over steps 500–4,000, and a plateau at ≈ 0.21–0.22. The 5000 run

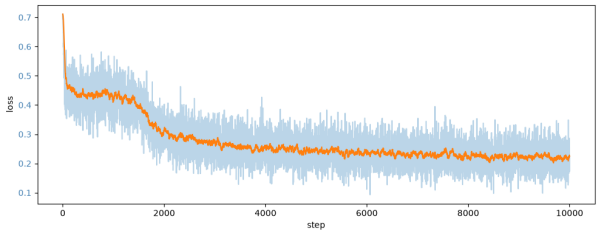
terminates mid descent at ≈ 0.25. The 8000 run reaches the plateau by step $\sim$6,000 and is flat for the remainder. Runs at 10000, 15000, and 20000 steps are indistinguishable past step $\sim$8000.



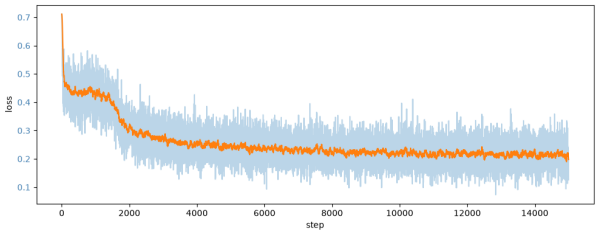
(a) 5000 steps



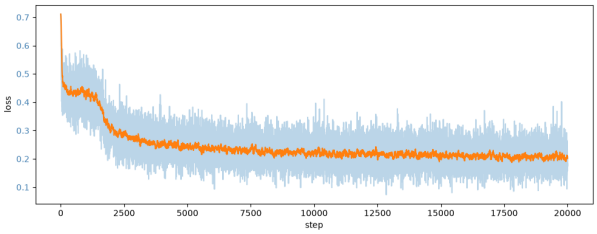
(b) 8000 steps



(c) 10000 steps



(d) 15000 steps



(e) 20000 steps

Figure 10. Training loss curves for the model across five step budgets $(5000 - 20000)$. Blue: per-step loss. Orange: EMA-smoothed loss ($\alpha = 0.95$). Loss plateaus at $\approx$ 0.21–0.22 by step $\sim$6000, with no improvement beyond 8000 steps.

Based on these results, 8000 training steps were selected as the shortest budget that reaches convergence, with training steps beyond 8000 offering no measurable improvement at additional compute cost. This is consistent with ShapPFN (Sena & Azevedo, 2026), which also trained for 8000 steps.

B.2. Learning Rate

The learning rate was optimized over five values, $\{10^{-4}, 5 \times 10^{-4}, 10^{-3}, 2 \times 10^{-3}, 4 \times 10^{-3}\}$, each trained for 8000 steps on the model with additive architecture and decision tree prior, where TreeSHAP weight $= 0$, viaSHAP weight $= 0$. Final model quality was assessed via ROC-AUC, accuracy, and balanced accuracy on a held out synthetic set.

Learning rate	ROC-AUC	Accuracy	Balanced accuracy
10^{-4}	0.850	0.812	0.812
5×10^{-4}	0.920	0.824	0.823
10^{-3}	0.902	0.828	0.828
2×10^{-3}	**0.923**	**0.860**	**0.859**
4×10^{-3}	0.500	0.488	0.500

Table 2. Evaluation metrics at the end of 8000 training steps for each learning rate value. Best values in bold.

The loss curves (Figure 11) show that 10^{-4} converges too slowly, still descending at step 8000 and plateauing higher than all other converged runs. 5×10^{-4}, 10^{-3}, and 2×10^{-3} all converge cleanly, with faster convergence and lower plateaus at higher values. 4×10^{-3} does not converge. The loss initially descends but then spikes sharply and plateaus. This is consistent with a large gradient update destroying learned representations, despite gradient clipping at max norm 1.0. This is reflected in the evaluation metrics (ROC-AUC $= 0.50$).

2×10^{-3} achieves the highest accuracy and balanced accuracy. It converges to the lowest training loss among stable runs. This is consistent with ShapPFN (Sena & Azevedo, 2026), which also identified 2×10^{-3} as optimal under an identical model configuration (3 layers, 4 attention heads, embedding size 96, hidden size 192). So, 2×10^{-3} is selected as the learning rate for all experiments.

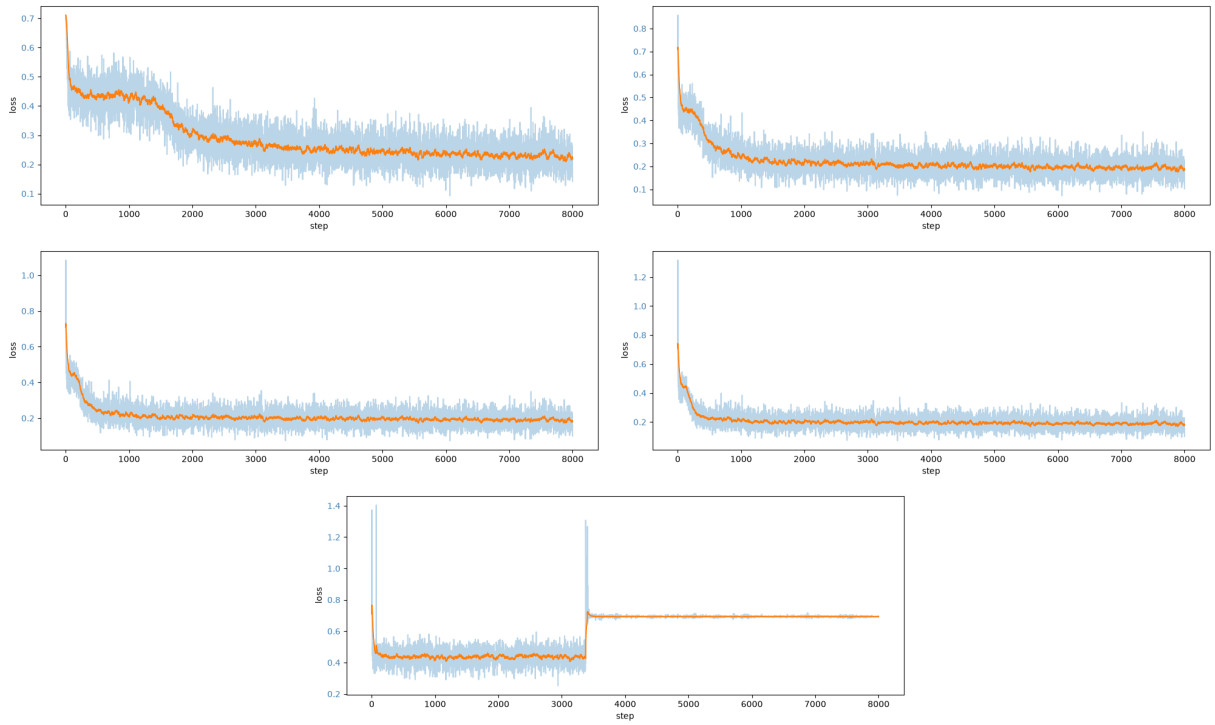


Figure 11. Training loss curves for learning rates 10^{-4} through 4×10^{-3} (top to bottom), each trained for 8000 steps. Blue: per-step loss. Orange: EMA-smoothed loss ($\alpha = 0.95$). Rates 5×10^{-4} - 2×10^{-3} converge. 10^{-4} converges slowly and plateaus. 4×10^{-3} does not converge.

C. Training Settings

Hyperparameter	Value
Embedding size	96
Number of attention heads	4
MLP hidden size	192
Number of layers	3 (`TransformerEncoderLayer` blocks)
Number of outputs	2 (binary classification)
Number of ϕ buckets	32

Table 3. Hyperparameter configuration used for the model.

Setting	Value
Optimizer	AdamWScheduleFree
Learning rate	2×10^{-3} (constant. ScheduleFree replaces an explicit LR schedule)
Weight decay	0.0
Batch size	32
Training steps	8000
Warmup steps	1200 (classification only steps before `loss_phi` is added)

Table 4. Training configuration used for the reported experiments.

D. Evaluation

D.1. Synthetic Datasets

All evaluations in this work that require an exact or closed form ground truth Shapley value use 8 synthetic binary classification datasets with i.i.d. standard normal features. Five hand constructed datasets and three drawn from the xai-bench (Liu et al., 2021) benchmark library. Independent features make exact Shapley computation tractable (or closed-form, for the xai-bench datasets), which is not possible on real, correlated-feature data. This is why the synthetic suite is used wherever a genuine ground truth is needed.

Hand-constructed datasets Each is a linear-threshold function $y = \mathbb{1}[X \cdot \text{coeffs} > \text{median}]$ over 4 or 8 i.i.d. Gaussian features, with a coefficient vector chosen to isolate one specific attribution-difficulty pattern:

- `single_relevant`: one relevant feature, three dummies. The minimal sanity check that a method assigns all importance to the one feature that matters and none to the rest.
- `two_unequal`: two relevant features with unequal weight (1 vs. 3). Tests whether attributions correctly reflect *relative* magnitude, not just relevance and irrelevance.
- `all_relevant`: four relevant features, no dummies. Tests behavior when there is no sparsity to exploit.
- `tie`: two relevant features with equal weight. Tests whether truly tied importance is attributed as a tie.
- `many_dummies`: eight features, only two relevant. A higher dimensional, sparse setting testing whether importance is correctly reflected in a larger pool of irrelevant features.

xai-bench datasets `gaussian_linear`, `gaussian_piecewise_constant`, and `gaussian_nonlinear_additive` are drawn from the xai-bench library. Each comes with a closed-form ground-truth Shapley value under conditional Gaussian sampling (exploiting the known generative distribution). The main value here is varying the functional form of the label.

- `gaussian_linear`: $y = \mathbb{1}[X \cdot w + \varepsilon \geq 0]$ with $w = (0, 1, 3, 0)$. Linear threshold datasets.
- `gaussian_piecewise_constant`: The label is a sum of fixed piecewise-constant (step) functions of the first three feature columns. The fourth column is pure noise. Importance is not proportional to any coefficient.
- `gaussian_nonlinear_additive`: The label is a sum of fixed nonlinear transforms (sine, absolute value, identity, exponential) applied to each of the four feature columns. The function is smooth but nonlinear per feature, with no interaction terms between features.

Table 5. KernelSHAP agreement on 8 benchmark datasets (mean ± SE over 3 seeds, 50 explained instances each) and computational efficiency. Time = single-pass ϕ wall-clock; speedup = KernelSHAP time / ϕ time. Background $= 150$ samples.

Dataset	ShapPFN				
	Time (s)	Speedup	R^2	Cos	ρ
Bank_Customer_Churn	0.006 ± 0.002	$916 \times \pm 255$	0.855 ± 0.031	0.926 ± 0.017	0.904 ± 0.013
Fitness_Club	0.002 ± 0.000	$151 \times \pm 12$	0.964 ± 0.008	0.982 ± 0.004	0.868 ± 0.026
Is-this-a-good-customer	0.004 ± 0.000	$2891 \times \pm 16$	0.806 ± 0.030	0.902 ± 0.015	0.867 ± 0.030
blood-transfusion-service-center	0.006 ± 0.001	$104 \times \pm 17$	0.906 ± 0.026	0.953 ± 0.014	0.828 ± 0.006
churn	0.005 ± 0.000	$3329 \times \pm 24$	0.740 ± 0.020	0.868 ± 0.012	0.642 ± 0.021
credit-g	0.005 ± 0.000	$3353 \times \pm 6$	0.858 ± 0.020	0.927 ± 0.010	0.849 ± 0.008
diabetes	0.002 ± 0.000	$433 \times \pm 7$	0.924 ± 0.016	0.961 ± 0.008	0.876 ± 0.002
seismic-bumps	0.002 ± 0.000	$250 \times \pm 11$	0.934 ± 0.009	0.967 ± 0.005	0.881 ± 0.015
Average	0.004 ± 0.001	$688 \times \pm 526$	0.873 ± 0.026	0.936 ± 0.013	0.839 ± 0.029

Dataset	additive_no_loss					additive_treeshap_only				
	Time (s)	Speedup	R^2	Cos	ρ	Time (s)	Speedup	R^2	Cos	ρ
Bank_Customer_Churn	0.010 ± 0.003	$673 \times \pm 306$	0.056 ± 0.080	0.277 ± 0.094	0.511 ± 0.040	0.006 ± 0.001	$1057 \times \pm 142$	0.735 ± 0.054	0.865 ± 0.028	0.829 ± 0.026
Fitness_Club	0.003 ± 0.000	$135 \times \pm 3$	0.363 ± 0.105	0.597 ± 0.083	0.534 ± 0.066	0.003 ± 0.000	$141 \times \pm 3$	0.865 ± 0.039	0.931 ± 0.020	0.773 ± 0.021
Is-this-a-good-customer	0.004 ± 0.000	$2858 \times \pm 10$	-0.021 ± 0.036	0.108 ± 0.065	0.363 ± 0.066	0.004 ± 0.000	$2944 \times \pm 15$	0.439 ± 0.153	0.676 ± 0.097	0.665 ± 0.034
blood-transfusion-service-center	0.007 ± 0.002	$76 \times \pm 36$	0.219 ± 0.088	0.464 ± 0.101	0.247 ± 0.072	0.003 ± 0.000	$149 \times \pm 2$	0.569 ± 0.038	0.766 ± 0.019	0.760 ± 0.017
churn	0.006 ± 0.000	$3371 \times \pm 37$	-0.189 ± 0.047	-0.068 ± 0.027	0.440 ± 0.004	0.006 ± 0.000	$3427 \times \pm 22$	0.365 ± 0.040	0.661 ± 0.013	0.506 ± 0.033
credit-g	0.006 ± 0.000	$3428 \times \pm 20$	0.222 ± 0.075	0.473 ± 0.069	0.429 ± 0.039	0.006 ± 0.000	$3426 \times \pm 13$	0.735 ± 0.046	0.861 ± 0.026	0.787 ± 0.012
diabetes	0.003 ± 0.000	$411 \times \pm 16$	0.313 ± 0.030	0.560 ± 0.026	0.495 ± 0.046	0.003 ± 0.000	$437 \times \pm 2$	0.896 ± 0.011	0.947 ± 0.006	0.880 ± 0.041
seismic-bumps	0.003 ± 0.000	$231 \times \pm 3$	0.350 ± 0.045	0.604 ± 0.027	0.121 ± 0.045	0.003 ± 0.000	$238 \times \pm 3$	0.823 ± 0.011	0.911 ± 0.003	0.822 ± 0.008
Average	0.005 ± 0.001	$619 \times \pm 540$	0.164 ± 0.070	0.377 ± 0.088	0.393 ± 0.051	0.004 ± 0.001	$729 \times \pm 536$	0.678 ± 0.070	0.827 ± 0.040	0.753 ± 0.042

Dataset	additive_viashap_only					additive_with_loss				
	Time (s)	Speedup	R^2	Cos	ρ	Time (s)	Speedup	R^2	Cos	ρ
Bank_Customer_Churn	0.050 ± 0.047	$315 \times \pm 368$	0.770 ± 0.051	0.881 ± 0.026	0.856 ± 0.010	0.073 ± 0.065	$285 \times \pm 496$	0.759 ± 0.026	0.875 ± 0.015	0.880 ± 0.009
Fitness_Club	0.003 ± 0.000	$123 \times \pm 17$	0.853 ± 0.021	0.924 ± 0.011	0.663 ± 0.038	0.003 ± 0.001	$123 \times \pm 16$	0.882 ± 0.027	0.940 ± 0.014	0.800 ± 0.033
Is-this-a-good-customer	0.004 ± 0.000	$2995 \times \pm 83$	0.739 ± 0.065	0.868 ± 0.032	0.744 ± 0.013	0.004 ± 0.000	$3087 \times \pm 75$	0.720 ± 0.080	0.862 ± 0.040	0.698 ± 0.004
blood-transfusion-service-center	0.006 ± 0.004	$71 \times \pm 34$	0.415 ± 0.077	0.687 ± 0.045	0.717 ± 0.059	0.008 ± 0.005	$99 \times \pm 49$	0.682 ± 0.039	0.842 ± 0.027	0.624 ± 0.014
churn	0.006 ± 0.000	$3617 \times \pm 133$	0.830 ± 0.026	0.928 ± 0.010	0.599 ± 0.013	0.007 ± 0.001	$3161 \times \pm 250$	0.542 ± 0.029	0.762 ± 0.013	0.520 ± 0.041
credit-g	0.006 ± 0.000	$3691 \times \pm 185$	0.769 ± 0.047	0.877 ± 0.027	0.661 ± 0.014	0.006 ± 0.000	$3700 \times \pm 184$	0.816 ± 0.072	0.905 ± 0.039	0.816 ± 0.015
diabetes	0.003 ± 0.000	$391 \times \pm 33$	0.886 ± 0.023	0.941 ± 0.012	0.850 ± 0.026	0.003 ± 0.000	$398 \times \pm 32$	0.891 ± 0.012	0.944 ± 0.006	0.906 ± 0.024
seismic-bumps	0.003 ± 0.000	$225 \times \pm 18$	0.859 ± 0.041	0.928 ± 0.020	0.873 ± 0.017	0.003 ± 0.000	$203 \times \pm 28$	0.887 ± 0.013	0.943 ± 0.007	0.911 ± 0.014
Average	0.010 ± 0.006	$560 \times \pm 593$	0.765 ± 0.053	0.879 ± 0.029	0.745 ± 0.037	0.013 ± 0.008	$564 \times \pm 571$	0.772 ± 0.043	0.884 ± 0.022	0.770 ± 0.050

Table 6. Predictive accuracy comparison across six models on all eight synthetic datasets (mean ± SE over 10 seeds, $N = 300$ samples per split).

Dataset	additive_with_loss		additive_no_loss		additive_treeshap_only	
	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy
single_relevant	1.000 ± 0.000	0.996 ± 0.001	1.000 ± 0.000	0.994 ± 0.003	1.000 ± 0.000	0.989 ± 0.003
two_unequal	0.980 ± 0.003	0.904 ± 0.010	0.989 ± 0.002	0.944 ± 0.006	0.984 ± 0.002	0.905 ± 0.008
all_relevant	0.958 ± 0.003	0.853 ± 0.010	0.962 ± 0.006	0.883 ± 0.008	0.969 ± 0.003	0.894 ± 0.005
tie	0.979 ± 0.004	0.841 ± 0.018	0.982 ± 0.002	0.916 ± 0.004	0.980 ± 0.004	0.902 ± 0.012
many_dummies	0.979 ± 0.002	0.899 ± 0.008	0.977 ± 0.004	0.912 ± 0.008	0.980 ± 0.003	0.898 ± 0.008
gaussian_linear	0.976 ± 0.006	0.907 ± 0.012	0.975 ± 0.007	0.915 ± 0.010	0.978 ± 0.009	0.909 ± 0.014
gaussian_piecewise_constant	0.881 ± 0.010	0.783 ± 0.015	0.857 ± 0.012	0.787 ± 0.013	0.883 ± 0.010	0.789 ± 0.013
gaussian_nonlinear_additive	0.978 ± 0.007	0.905 ± 0.015	0.989 ± 0.003	0.936 ± 0.013	0.980 ± 0.005	0.911 ± 0.017
Average	**0.966 ± 0.013**	**0.886 ± 0.022**	0.966 ± 0.016	0.911 ± 0.021	0.969 ± 0.013	0.900 ± 0.019

Dataset	additive_viashap_only		ShapPFN		NanoTabPFN-tree	
	ROC-AUC	Accuracy	ROC-AUC	Accuracy	ROC-AUC	Accuracy
single_relevant	1.000 ± 0.000	0.994 ± 0.002	1.000 ± 0.000	0.980 ± 0.003	1.000 ± 0.000	0.994 ± 0.003
two_unequal	0.985 ± 0.002	0.907 ± 0.007	0.989 ± 0.002	0.923 ± 0.006	0.987 ± 0.002	0.937 ± 0.005
all_relevant	0.964 ± 0.004	0.880 ± 0.007	0.985 ± 0.002	0.929 ± 0.007	0.960 ± 0.005	0.882 ± 0.011
tie	0.980 ± 0.005	0.898 ± 0.010	0.989 ± 0.002	0.930 ± 0.009	0.985 ± 0.002	0.928 ± 0.007
many_dummies	0.980 ± 0.003	0.905 ± 0.006	0.985 ± 0.002	0.921 ± 0.006	0.980 ± 0.004	0.909 ± 0.008
gaussian_linear	0.979 ± 0.006	0.903 ± 0.014	0.984 ± 0.005	0.928 ± 0.012	0.973 ± 0.006	0.919 ± 0.010
gaussian_piecewise_constant	0.884 ± 0.008	0.796 ± 0.011	0.879 ± 0.008	0.811 ± 0.008	0.844 ± 0.011	0.768 ± 0.012
gaussian_nonlinear_additive	0.987 ± 0.004	0.927 ± 0.014	0.994 ± 0.002	0.953 ± 0.011	0.998 ± 0.001	0.968 ± 0.009
Average	0.970 ± 0.013	0.901 ± 0.019	0.976 ± 0.014	0.922 ± 0.017	0.966 ± 0.018	0.913 ± 0.024

4

Reflection

This work sits at the intersection of two lines of research. Explainability methods have treated attribution as a deterministic point estimate, while uncertainty quantification research has focused almost entirely on prediction intervals and calibration of the output itself, rarely the explanation. Framing attributions as calibrated distributions is a bridge between the two, and it lands amid a broader push in the field toward uncertainty aware ML. This is visible in work on conformal prediction for tabular models and in regulatory pressure such as the EU AI Act's transparency and "right to explanation" provisions, both of which are increasingly skeptical of explanations that imply false precision. The PFN backbone ties this to in-context learning and amortized Bayesian inference for tabular data. Most tabular foundation model papers report accuracy and calibration of predictions but say little about the reliability of any accompanying explanation. Grounding attributions in a synthetic DGP with an exact, closed-form target also addresses a longstanding evaluation gap in XAI. Without a known ground truth, faithfulness is usually assessed only indirectly, through proxies like ROAR or human agreement. Having an exact target here is a way of testing what those proxies actually measure.

For a practitioner doing feature selection or model debugging, the practical payoff is being able to distinguish "this feature reliably doesn't matter" from "I can't tell if this feature matters".This could change which features get dropped or scrutinized further. For someone subject to an automated decision, e.g. a loan applicant, a distributional attribution is only useful if the uncertainty is itself trustworthy. Here that trust rests entirely on feature independence, which real applicant data cannot guarantee by construction (income and education correlate, for instance). For a regulator or auditor, a system that can point to "calibrated, verified" attributions looks more auditable, but the verification here only holds under conditions unlikely to be met by the data. So the method's honesty depends on how carefully that scope is communicated downstream. For stakeholders, per-feature attributions are a double-edged sword, because the same signal that aids understanding also helps an attacker infer which features to manipulate. A full distribution arguably leaks more than a point estimate would. The environmental cost is also real collective concern, given how much of the explainability literature depends on this kind of repeated retraining.

5

Use Case

Predictive attribution distributions are essential because they distinguish a contribution that is confidently small from one that merely has a small point estimate but remains highly uncertain.

Figure 5.1 shows feature attributions of two instances explained by our model. The additive decomposition is shown from a shared baseline of $\text{base}_1 = -0.294$. In both cases every feature pushes toward class 0 ("tested negative"), and both patients are classified as Class 0.

The two features `BloodPressure` in patient 1 and `Glucose` in patient 2 have almost identical point estimates for feature attribution $\phi_{\text{BloodPressure},1} = -0.007$ versus $\phi_{\text{Glucose},1} = -0.011$. Read off the waterfall plot alone, or off any point-valued attribution method (KernelSHAP, ShapPFN), these two contributions look interchangeable (small pushes to class 0).

The model's predictive distribution over each ϕ (Figure 5.2) shows that in patient 1, `BloodPressure`'s distribution is sharply concentrated around its point estimate, with a 90% credible interval of $[-0.093, +0.062]$ (width 0.155). The model is certain this feature's true contribution is small and near zero.

The distribution of`Glucose` in patient 2 is much more diffuse. It has substantial probability spread across a broad range of attribution values, with a 90% credible interval of $[-0.267, +0.651]$ (width 0.919, roughly six times wider). The model expresses greater uncertainty about Glucose's contribution for this patient, and that contribution could plausibly be as large as $+0.65$. Using the model's uncertainty, `Glucose` could either increase or decrease the predicted logit for class 1. Under a hypothetical attribution of $+0.65$, while holding all other feature contributions fixed, the resulting logit would increase from -0.432 to $+0.23$, changing the predicted class.

This distinction matters because `Glucose` the beeswarm plot (Figure 2.3) shows it is the single most influential feature on average across the dataset. Yet the model is roughly six times less certain about `Glucose`'s contribution for patient 2 than about `BloodPressure`'s contribution for patient 1. `BloodPressure` is a feature the model otherwise treats as unimportant in general.

An independent reference, NanoTabPFN (tree prior) + KernelSHAP, supports this picture. For patient 1's `BloodPressure` it lands at $+0.001$, inside the narrow band. For patient 2's `Glucose` it lands at -0.049, far from the -0.011 point estimate but still inside the wide band. A point estimate comparison would flag that gap as disagreement. The model's own credible interval already predicts and accounts for it. A single scalar ϕ cannot tell "the model is confident this contribution is small" apart from "the model is not certain, and it happens to average out to something small." The full predictive distribution over ϕ makes that difference visible.

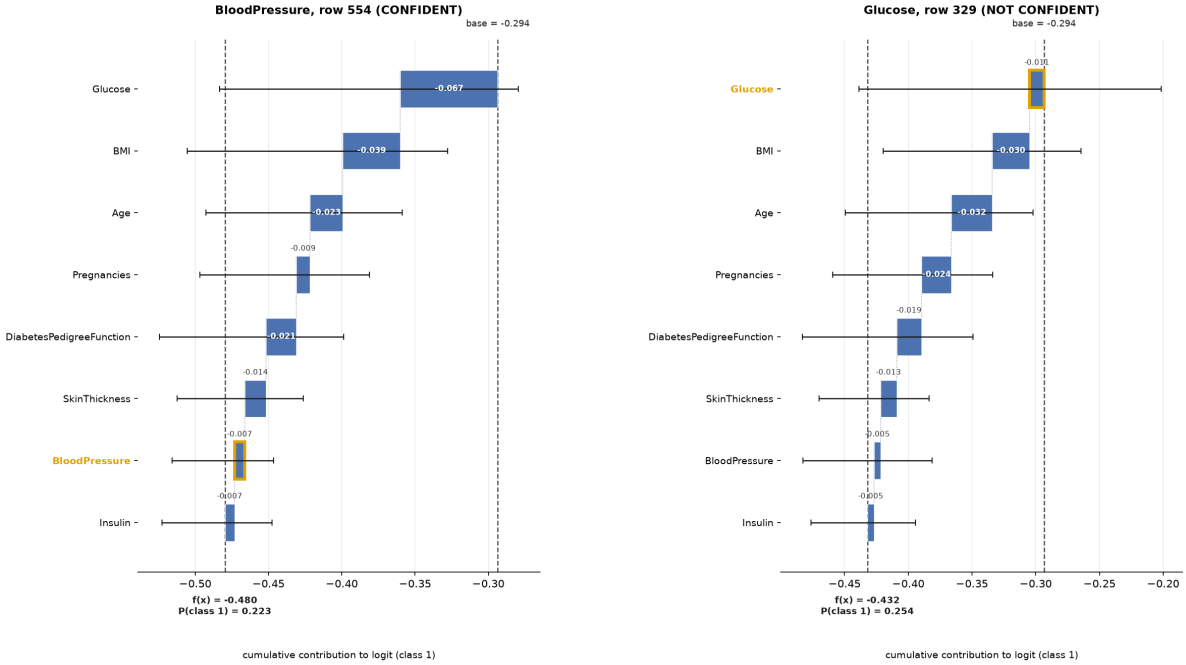


Figure 5.1: Additive decomposition for two patients. **Left:** patient 1, zoomed on `BloodPressure` (gold outline), a narrow-CI ("confident") attribution. **Right:** patient row 2, zoomed on `Glucose`, a wide-CI ("not confident") attribution. Black whiskers show each feature's credible intervals. Both patients share the same class-1 baseline, $\text{base}_1 = -0.294$. Both are predicted class 0 ("tested negative").

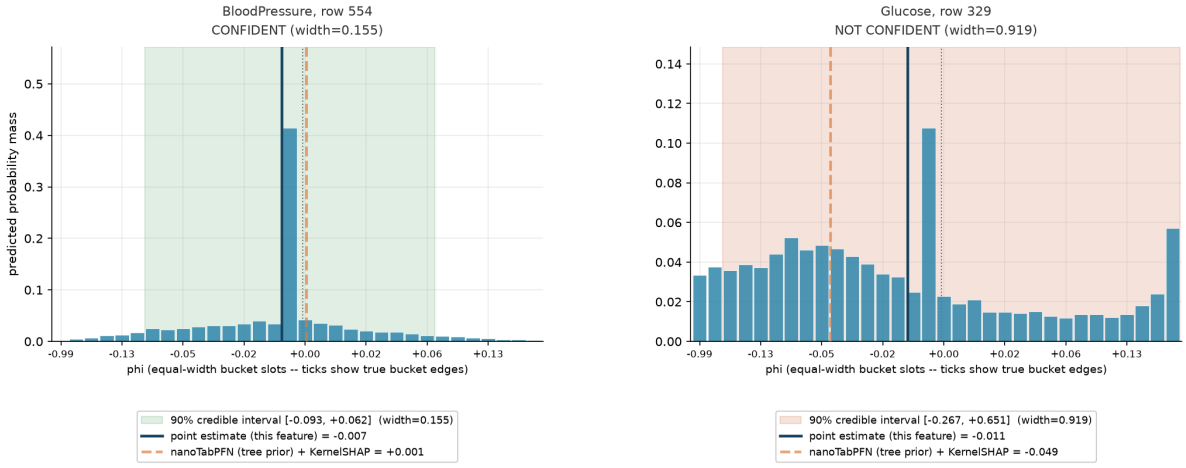


Figure 5.2: Predicted bucket distribution over ϕ for the two features in Figure 5.1, shown as a histogram (x-axis ticks show the bucket edges). **Left:** `BloodPressure` (patient 1) is sharply concentrated near its point estimate, 90% credible interval $[-0.093, +0.062]$ (width 0.155). **Right:** `Glucose` (patient 2) is nearly flat across the full range, 90% credible interval $[-0.267, +0.651]$ (width 0.919), despite a similar-looking point estimate (-0.011 vs. -0.007). Dashed orange line: NanoTabPFN (tree prior) + KernelSHAP reference.

References

[1] Amr Alkhatib et al. *Prediction via Shapley Value Regression*. 2025. arXiv: 2505.04775 [cs.LG].

[2] Christophe Andrieu et al. “An Introduction to MCMC for Machine Learning”. In: *Machine Learning* 50.1 (Jan. 2003), pp. 5–43. ISSN: 1573-0565. DOI: 10.1023/A:1020281327116. URL: https://doi.org/10.1023/A:1020281327116.

[3] Felix den Breejen et al. *Fine-tuned In-Context Learning Transformers are Excellent Tabular Data Classifiers*. 2025. arXiv: 2405.13396 [cs.LG].

[4] Rich Caruana et al. “Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission”. In: Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1721–1730.

[5] Aaron Defazio et al. “The Road Less Scheduled”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Vol. 37. Curran Associates, Inc., 2024, pp. 9974–10007. DOI: 10.52202/079017-0320.

[6] Nick Erickson et al. *TabArena: A Living Benchmark for Machine Learning on Tabular Data*. 2025. arXiv: 2506.16791 [cs.LG].

[7] Joao Fonseca and Julia Stoyanovich. *ExplainerPFN: Towards tabular foundation models for model-free zero-shot feature importance estimations*. 2026. arXiv: 2601.23068 [cs.LG].

[8] Christopher Frye, Colin Rowat, and Ilya Feige. *Asymmetric Shapley values: incorporating causal knowledge into model-agnostic explainability*. 2021. arXiv: 1910.06358 [stat.ML].

[9] Amirata Ghorbani, Abubakar Abid, and James Zou. “Interpretation of Neural Networks Is Fragile”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 3681–3688. DOI: 10.1609/aaai.v33i01.33013681. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4252.

[10] Tom Heskes et al. *Causal Shapley Values: Exploiting Causal Knowledge to Explain Individual Predictions of Complex Models*. 2020. arXiv: 2011.01625 [cs.AI].

[11] Noah Hollmann et al. “Accurate predictions on small data with a tabular foundation model”. In: *Nature* (Jan. 2025). DOI: 10.1038/s41586-024-08328-6.

[12] Noah Hollmann et al. “TabPFN: A transformer that solves small tabular classification problems in a second”. In: *International Conference on Learning Representations 2023*. 2023.

[13] Sara Hooker et al. *A Benchmark for Interpretability Methods in Deep Neural Networks*. 2019. arXiv: 1806.10758 [cs.LG].

[14] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. *Feature relevance quantification in explainable AI: A causal problem*. 2019. arXiv: 1910.13413 [stat.ML].

[15] Neil Jethani et al. *FastSHAP: Real-Time Shapley Value Estimation*. 2022. arXiv: 2107.07436 [stat.ML].

[16] Been Kim, Rajiv Khanna, and Oluwasanmi Koyejo. “Examples are not enough, learn to criticize! criticism for interpretability”. In: Barcelona, Spain: Curran Associates Inc., 2016, pp. 2288–2296.

[17] Zachary Chase Lipton. “The Mythos of Model Interpretability”. In: *CoRR* abs/1606.03490 (2016). arXiv: 1606.03490. URL: http://arxiv.org/abs/1606.03490.

[18] Yang Liu et al. *Synthetic Benchmarks for Scientific Research in Explainable Machine Learning*. 2021. arXiv: 2106.12543 [cs.LG].

[19] Si-Yang Liu et al. “Talent: A Tabular Analytics and Learning Toolbox”. In: *Journal of Machine Learning Research* 26.226 (2025), pp. 1–16. URL: http://jmlr.org/papers/v26/25-0512.html.

[20] Scott Lundberg and Su-In Lee. *A Unified Approach to Interpreting Model Predictions*. 2017. arXiv: 1705.07874 [cs.AI].

[21] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee. *Consistent Individualized Feature Attribution for Tree Ensembles*. 2019. arXiv: 1802.03888 [cs.LG].

[22] Ratmir Miftachov, Bruno Charron, and Simon Valentin. *Interpretable Tabular Foundation Models via In-Context Kernel Regression*. 2026. arXiv: 2602.02162 [cs.LG].

[23] Christoph Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 3rd ed. 2025. ISBN: 978-3-911578-03-5.

[24] Andreas Mueller et al. *GAMformer: Bridging Tabular Foundation Models and Interpretable Machine Learning*. 2026. arXiv: 2410.04560 [cs.LG].

[25] Samuel Müller et al. *Transformers Can Do Bayesian Inference*. 2024. arXiv: 2112.10510 [cs.LG].

[26] Alexander Pfefferle et al. "nanoTabPFN: A Lightweight and Educational Reimplementation of TabPFN". In: *arXiv preprint arXiv:2511.03634* (2025).

[27] Jingang Qu et al. *TabICL: A Tabular Foundation Model for In-Context Learning on Large Data*. 2025. arXiv: 2502.05564 [cs.LG].

[28] Jingang Qu et al. *TabICLv2: A better, faster, scalable, and open tabular foundation model*. 2026. arXiv: 2602.11139 [cs.LG].

[29] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (May 2019), pp. 206–215.

[30] Luan Borges Teodoro Reis Sena and Francisco Galuppo Azevedo. *Real-Time Explanations for Tabular Foundation Models*. 2026. arXiv: 2603.29946 [cs.LG].

[31] Lloyd S. Shapley. "A Value for n-Person Games". In: *Contributions to the Theory of Games II*. Ed. by Harold W. Kuhn and Albert W. Tucker. Vol. 28. Annals of Mathematical Studies. Princeton, NJ: Princeton University Press, 1953, pp. 307–317.

[32] Ravid Shwartz-Ziv and Amitai Armon. *Tabular Data: Deep Learning is Not All You Need*. 2021. arXiv: 2106.03253 [cs.LG].

[33] Ashish Vaswani et al. "Attention is all you need". In: Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010.

[34] Martin J. Wainwright and Michael I. Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* 1.1-2 (Dec. 2008), pp. 1–305. ISSN: 1935-8237. DOI: 10.1561/2200000001. eprint: https://www.emerald.com/ftmal/article-pdf/1/1-2/1/11153930/2200000001en.pdf. URL: https://doi.org/10.1561/2200000001.

[35] Han-Jia Ye et al. "A Closer Look at Deep Learning on Tabular Data". In: *arXiv preprint arXiv:2407.00956* (2024).

A

Use of Generative AI

AI tools were used in the following ways for this work:

1. Conducting Research
   (a) Idea Testing: Exploring alternative approaches to strengthen arguments.
   (b) Data Analysis Support: Generating scripts for visualization, while ensuring that all results and interpretations are understood and verified.
2. Writing
   (a) Writing Quality Enhancement: Checking grammar, spelling, style, and improving readability.
   (b) Formatting Assistance: Producing tables and converting text into LaTeX.

All AI generated output was reviewed and edited. I retain full responsibility for the contents of this work.