Annals of Actuarial Science (2026), 1–24
doi:10.1017/S1748499526100402

ORIGINAL RESEARCH PAPER

Tree-like pairwise interaction networks

Ronald Richman[1], Salvatore Scognamiglio[2] and Mario Wüthrich[3]

[1]InsureAI, South Africa; [2]Management and Quantitative Sciences, University of Naples – Parthenope, Italy; and [3]Department of Mathematics, ETH Zurich, Switzerland
Corresponding author: Salvatore Scognamiglio; Email: salvatore.scognamiglio@uniparthenope.it

(Received 22 August 2025 UTC; revised 04 June 2026 UTC; accepted 05 August 2026 UTC)

Abstract

Modeling feature interactions in tabular data remains a key challenge in predictive modeling, for example, as used for insurance pricing. This paper proposes the tree-like pairwise interaction network (PIN), a novel neural network architecture that explicitly captures pairwise feature interactions through a shared feed-forward neural network architecture that mimics the structure of decision trees. PIN enables intrinsic interpretability by design, allowing for direct inspection of interaction effects. Moreover, it allows for efficient SHapley's Additive exPlanation computations because it only involves pairwise interactions. We highlight connections between PIN and established models such as GA^{2}Ms, gradient boosting machines, and graph neural networks. Empirical results on the popular French motor insurance dataset show that PIN outperforms both traditional and modern neural network benchmarks in predictive accuracy, while also providing insight into how features interact with each another and how they contribute to the predictions.

Keywords: generalized additive models; insurance pricing; interaction effects; neural networks; Shapley explanation

1. Introduction

Understanding and modeling the interactions among input features represent a core challenge in predictive modeling on tabular data. These interactions refer to how multiple variables jointly influence the target outcome in ways that go beyond their individual, isolated effects. In many real-world cases, the relationship between predictors and the response variable is not simply additive and may involve complex interdependencies. This is particularly relevant in insurance pricing, where factors such as driver age, location, and driving behavior can interact in nonobvious ways to affect risk assessment and premium calculation. Overlooking or misspecifying such interactions can lead to suboptimal models, resulting in price distortions and potentially biased interpretations.

Traditionally, the insurance industry has relied on generalized linear models (GLMs) due to their interpretability, strong statistical foundation, and also for computational reasons; we refer to Nelder and Wedderburn (1972) and McCullagh and Nelder (1983). In the GLM framework, the expected value of the response variable is modeled as a linear combination of predictors, linked to the response through a specified link function. While GLMs allow for the estimation of individual covariate effects, they offer limited capability to capture interactions among variables. In fact, interactions can be integrated, but each one needs to be engineered manually, which clearly limits their scope and the model complexity and accuracy.

Generalized additive models (GAMs) were introduced in Hastie and Tibshirani (1986) as an extension of GLMs relaxing the assumption of linearity see also Wood (2017). They allow each

© The Author(s), 2026. Published by Cambridge University Press on behalf of The Institute and Faculty of Actuaries. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

predictor to have a nonlinear effect on the response variable. This is accomplished by replacing the original covariates with smooth functions (typically splines) applied individually to each input variable. GAMs have often been used in the context of actuarial science as a foundation for developing more sophisticated methods to solve actuarial problems; notable examples include Chang *et al.* (2024) and Maillart and Robert (2024). Building upon GAMs with Pairwise Interactions (GA^2Ms) extend the framework by incorporating a selected set of pairwise feature interactions, see Lou *et al.* (2013). These additional components enable the model to capture the joint effects of feature pairs on the response variable. Similarly to interactions in GLMs, GA^2Ms typically require manual identification and specification of interaction terms, which demands domain expertise and time-consuming feature engineering. This reliance on expert judgment can introduce subjectivity and biases, and it may make the model sensitive to the modeler's choices, potentially affecting both predictive performance and generalization capability.

Deep neural networks (DNNs) have attracted a growing attention in actuarial science as powerful tools for predictive modeling; for an extended discussion see Richman (2021a,b). These models can be viewed as an extension of GLMs that allow for the modeling of nonlinear relationships; see Chapter 5 of Wüthrich *et al.* (2025). In particular, based on a successful training algorithm, DNNs perform automated feature engineering. Through the mechanism of representation learning, DNNs transform original input features into new, higher-level representations via multiple layers of nonlinear functions, which are optimized to predict the response variable. This hierarchical structure allows the model to automatically construct features that capture higher-order interactions among the original covariates. Despite their strong predictive performance, DNNs present significant challenges, particularly in identifying, quantifying and interpreting feature interactions. The complexity of the learned representations makes it difficult to understand how input variables jointly contribute to the model's predictions. For these reasons, DNNs are sometimes called black box models.

Modern machine learning literature grapples with the challenge of effectively modeling and interpreting feature interactions within neural networks. Song *et al.* (2019) address this issue by proposing a specialized DNN architecture designed to automatically learn feature interactions in presence of tabular data. The model employs a multi-head self-attention mechanism, inspired by the transformer architecture of Vaswani *et al.* (2017), to explicitly capture interactions between features, thereby eliminating the need for manual feature engineering. Tsang *et al.* (2018) introduce a method for detecting statistical interactions in trained DNNs by analyzing the values of the learned weights. Li *et al.* (2019) propose modeling categorical feature interactions by representing covariates using a graph, where each node corresponds to a distinct feature field. Interactions among features are effectively captured through the propagation mechanisms of a graph neural network (GNN). Enouen and Liu (2022) develop a DNNs-based approach that extends classical GAMs by introducing an additional component to capture higher-order feature interactions. Havrylenko and Heger (2024) revisit the CANN proposal of Wüthrich and Merz (2019) to identify the most significant interacting pairs missing in a GLM.

The contribution of this paper is to introduce the tree-like pairwise interaction network (PIN), a novel neural network architecture designed to model and identify pairwise feature interactions in tabular data. The PIN architecture embeds each input feature into a learned latent space and then explicitly models all pairwise interactions through a shared feed-forward neural network (FNN). Sharing this FNN implies an efficient way of dealing with model parameters, and to still allow for variability between the different interactions, each interaction makes use of a dedicated set of learnable parameters – playing a similar role to the classification (CLS) token used in Devlin *et al.* (2018) – which is aimed at modeling the dependence structure between each pair of features. The output of each interaction is passed through a centered hard sigmoid activation function that mimics the discrete partitioning behavior of decision trees, yet in a continuous and differentiable form. This results in tree-like structures (using binary splits), and this model allows for direct inspection of interaction terms allowing for model explainability. The proposed PIN also exhibits

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

connections with several well-established machine learning models, including gradient boosting, GA^{2}M, and GNNs. These relationships are explored in detail throughout the manuscript. We assess the performance of the tree-like PIN on two widely studied datasets, the French motor third-party liability (MTPL) claims frequency dataset and the Belgian MTPL dataset, both available from of Dutang *et al.* (2024). Our experiments demonstrate that the PIN architecture delivers strong predictive accuracy in terms of Poisson losses, outperforming not only classical benchmarks but also more recent neural network architectures, such as the credibility transformer introduced by Richman *et al.* (2025). Moreover, unlike most other network architectures, the PIN architecture allows for an efficient computation of SHapley's Additive exPlanations (SHAP). The crucial point is that the PIN architecture only involves pairwise interactions, and it has been shown in Lundberg (2018) and Mayer and Wüthrich (2025, Proposition 3.1) that this allows for a very efficient computation of the Shapley values using paired permutation SHAP sampling.

Concluding this introduction: Our novel architecture of the tree-like PIN provides superior performance on tabular data over currently used network architectures, and at the same time, it provides explainability. We believe that this makes the tree-like PIN an interesting proposal for actuarial pricing.

The paper is organized as follows. In Section 2, we introduce the PIN architecture, starting with feature embeddings and proceeding through the PIN. Section 3 details the application of the PIN to claims frequency modeling. In Section 4.1, we propose interaction importance measures for model interpretation, and we illustrate how the SHAP explanation can be computed within the PIN architecture using paired-sampling permutation SHAP. In Section 5, we study a second real-world dataset to confirm the excellent predictive performance of the PIN proposal. Finally, Section 6 concludes with a summary and future research directions.

2. Tree-like pairwise interaction network

2.1 Feature embedding

We start from tabular input data $x = (x_1, \ldots, x_q)^\top$. In a first step, we tokenize this tabular input data to an input tensor $\boldsymbol{\phi} = \boldsymbol{\phi}(x) = [\phi_1, \ldots, \phi_q] \in \mathbb{R}^{d \times q}$; this step is done in complete analogy to Gorishniy *et al.* (2021), Brauer (2024), and Richman *et al.* (2025). For this first step, we select a fixed embedding dimension $d \in \mathbb{N}$; this is a hyperparameter selected by the modeler.

Categorical features.
For categorical feature components $x_j \in \mathcal{X}_j = \{1, \ldots, n_j\} \subset \mathbb{N}$, having n_j levels, consider a d-dimensional entity embedding $\phi_j : \mathcal{X}_j \to \mathbb{R}^d$

$$x_j \mapsto \phi_j(x_j) = \sum_{k=1}^{n_j} W_{j,k}^{(0)} \mathbb{1}_{\{x_j = k\}}, \tag{2.1}$$

where $W_j^{(0)} = [W_{j,1}^{(0)}, \ldots, W_{j,n_j}^{(0)}]^\top \in \mathbb{R}^{n_j \times d}$ is an embedding matrix, which is a learnable parameter.

Continuous features.
For continuous feature components $x_j \in \mathbb{R}$, we consider a d-dimensional FNN embedding $\phi_j : \mathbb{R} \to \mathbb{R}^d$

$$x_j \mapsto \phi_j(x_j) = W_j^{(2)} \tanh\left(W_j^{(1)} x_j + b_j^{(1)}\right) + b_j^{(2)} \tag{2.2}$$

with biases $b_j^{(1)} \in \mathbb{R}^{d'}$ and $b_j^{(2)} \in \mathbb{R}^d$, weight matrices $W_j^{(1)} \in \mathbb{R}^{d' \times 1}$ and $W_j^{(2)} \in \mathbb{R}^{d \times d'}$, for a fixed number of units d', and where the hyperbolic tangent function is applied element-wise.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

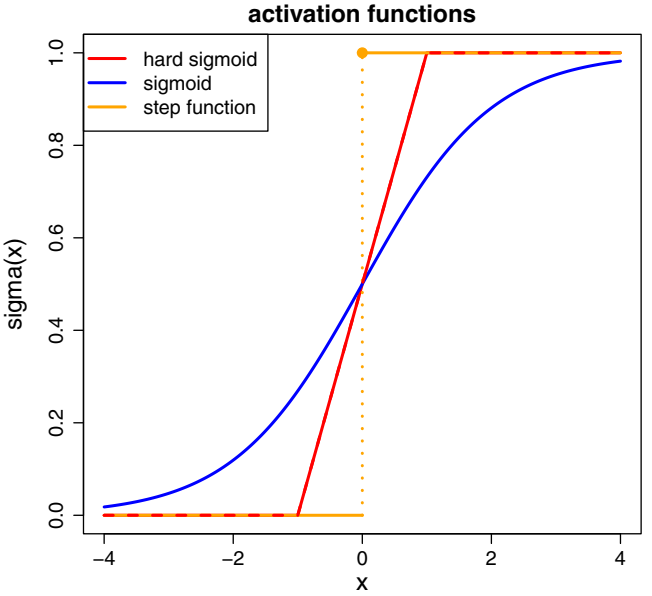


Figure 1. Activation functions: (centered) hard sigmoid, sigmoid, and step function.

Collecting all embedded components $(\phi_j)_{j=1}^q = (\phi_j(x_j))_{j=1}^q$ of the input x gives us the tensor

$$\boldsymbol{\phi} = \boldsymbol{\phi}(x) = [\phi_1, \ldots, \phi_q] = [\phi_1(x_1), \ldots, \phi_q(x_q)] \in \mathbb{R}^{d \times q}. \tag{2.3}$$

This tensor $\boldsymbol{\phi}$ involves the embedding weights $W_j^{(0)} \in \mathbb{R}^{n_j \times d}$ from the categorical features (of dimension $n_j d$), and the FNN biases and weights $(b_j^{(1)}, b_j^{(2)}, W_j^{(1)}, W_j^{(2)})$ from the continuous feature embeddings (of dimension $2d' + (d'+1)d$). All these biases and weights are learned from the available training data.

The categorical entity embeddings (2.1) and the continuous feature embeddings (2.2) do not necessarily live on the same scale. If a common scale is desired, this can be achieved either through ridge regularization or by applying separate normalization layers to the two types of embeddings. In our examples, the embeddings are not directly extracted, but they are further processed by interaction tokens introduced in the next section. These interaction tokens alleviate the potential scale discrepancies. Therefore, we do not regularize or normalize the embeddings.

2.2 Pairwise interaction layer

The tensor $\boldsymbol{\phi} = \boldsymbol{\phi}(x)$, given in (2.3) and containing the tokens $(\phi_j(x_j))_{j=1}^q$, is expressing the individual feature components of $x \in \mathcal{X}$ before they have been interacting with each other. In the next step, we are going to let these tokens interact in a pairwise manner. This can be interpreted in a similar way as the query–key interaction concept in the attention layer of Vaswani *et al.* (2017).

We recall the (centered version of the) *hard sigmoid activation*

$$\sigma_{\text{hard}} : \mathbb{R} \to [0,1], \qquad x \mapsto \sigma_{\text{hard}}(x) = \max\left(0, \min\left(1, \frac{1+x}{2}\right)\right). \tag{2.4}$$

This is the centered version of the hard sigmoid activation; there is also a noncentered version with $(1+x)/2$ being replaced by x in (2.4). The centered version is directly comparable to the sigmoid activation function and the step function activation, see Figure 1.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

We now let ϕ_j and ϕ_k interact, $j \le k$. To support these interactions, we introduce *interaction tokens* $e_{j,k} \in \mathbb{R}^{d_0}$, for a given hyper-parameter $d_0 \in \mathbb{N}$. The tokens consist of a set of learnable parameters that are optimized jointly with the rest of the model during network training. Before interacting, these tokens do not encode any information and are randomly initialized, like the other network parameters. This is analogous to the CLS token used in Devlin *et al.* (2018) and Richman *et al.* (2025). After training, the tokens are calibrated to support pairwise interactions between the j-th and k-th covariates. We concatenate the three vectors to a row-vector $(\phi_j, \phi_k, e_{j,k})^\top \in \mathbb{R}^{2d+d_0}$, $j \le k$. The goal of the interaction tokens is to learn the interaction structure within the tensor $\boldsymbol{\phi}$ for predicting the response Y, in particular, different pairs (ϕ_j, ϕ_k) may have a different interaction behavior which is reflected in different learned interaction tokens $e_{j,k}$, $j \le k$. Unlike the CLS token of Devlin *et al.* (2018) and Richman *et al.* (2025), we do not use these interaction tokens to strip out information, but we add them to allow for the necessary modeling flexibility to be able to account for differences in the different interaction pairs.

Define the *shared interaction network* $f_\theta : \mathbb{R}^{2d+d_0} \to \mathbb{R}$ as a composition of three FNN layers

$$f_\theta(\phi_j, \phi_k, e_{j,k}) = W^{(3)}\, \mathrm{ReLU}\Big(W^{(2)}\, \mathrm{ReLU}\Big(W^{(1)}(\phi_j, \phi_k, e_{j,k})^\top + b^{(1)}\Big) + b^{(2)}\Big) + b^{(3)}, \tag{2.5}$$

with:

- the 1st FNN layer has d_1 units, rectified linear unit (ReLU) activation, network weights $W^{(1)} \in \mathbb{R}^{d_1 \times (2d+d_0)}$, and bias $b^{(1)} \in \mathbb{R}^{d_1}$;
- the 2nd FNN layer has d_2 units, ReLU activation, network weights $W^{(2)} \in \mathbb{R}^{d_2 \times d_1}$, and bias $b^{(2)} \in \mathbb{R}^{d_2}$;
- the output layer has 1 unit, linear activation, network weights $W^{(3)} \in \mathbb{R}^{1 \times d_2}$, and bias $b^{(3)} \in \mathbb{R}$;

and $\theta = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, W^{(3)}, b^{(3)})$ collects all network parameters. This interaction network f_θ is going to be shared by all pairwise interactions between the components of $\boldsymbol{\phi}$, i.e., there is one single network parameter θ that is shared by all ordered pairs (ϕ_j, ϕ_k), $1 \le j \le k \le q$. This can also be seen as a so-called time-distributed network as being used, e.g., in recurrent neural networks. To differentiate different forms of interactions of the different ordered pairs, $j \le k$, we let the interaction tokens $e_{j,k}$ learn the pairwise interaction structures.

Definition 2.1 (pairwise interaction layer). *For feature tokens $\phi_j(\cdot)$ and $\phi_k(\cdot)$, $1 \le j \le k \le q$, and interaction tokens $e_{j,k}$, we define the pairwise interaction units by*

$$h_{j,k}(x) = \sigma_{\mathrm{hard}}(f_\theta(\phi_j(x_j), \phi_k(x_k), e_{j,k})) \in [0, 1]. \tag{2.6}$$

The pairwise interaction layer for $x \in \mathcal{X}$ is defined by the upper-right triangular matrix

$$\big(h_{j,k}(x)\big)_{1 \le j \le k \le q} = \begin{pmatrix} h_{1,1}(x) & h_{1,2}(x) & \cdots & h_{1,q-1}(x) & h_{1,q}(x) \\ \mathrm{n/a} & h_{2,2}(x) & \cdots & h_{2,q-1}(x) & h_{2,q}(x) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathrm{n/a} & \mathrm{n/a} & \cdots & h_{q-1,q-1}(x) & h_{q-1,q}(x) \\ \mathrm{n/a} & \mathrm{n/a} & \cdots & \mathrm{n/a} & h_{q,q}(x) \end{pmatrix}, \tag{2.7}$$

where the lower-left part is undefined indicated by n/a.

We remark that the diagonal entries represent the "self-interaction" of each covariate component and can be interpreted as the corresponding individual (main) effects. Including these diagonal terms allows the model to learn nonlinear functions that capture the marginal contribution of each variable, while preserving the same unified representation used for modeling pairwise interactions.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

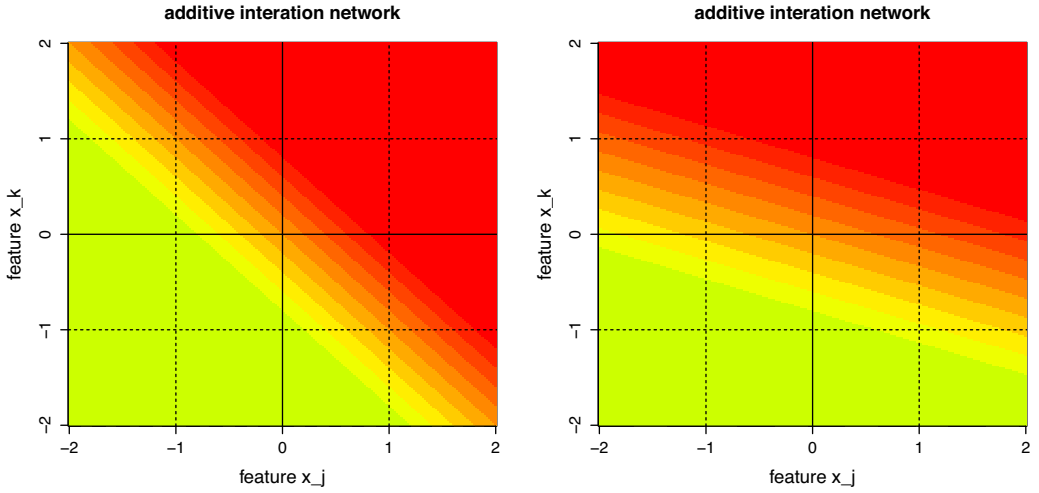


Figure 2. Interaction layers: (lhs) $\sigma_{\text{hard}}(x_j + x_k)$ and (rhs) $\sigma_{\text{hard}}(x_j/3 + x_k)$ for $j < k$.

Example 2.2. *We give a simple example of the pairwise interaction layer. We assume only continuous features, that is, $x \in \mathbb{R}^q$. We select one-dimensional linear feature embeddings (tokenization) $x_j \mapsto \phi_j(x_j) = x_j$ for all $1 \le j \le q$. This gives us the tensor $\boldsymbol{\phi} = \boldsymbol{\phi}(x) = x \in \mathbb{R}^{1\times q}$. For the shared interaction network f_θ, we select a simple additive function and the interaction tokens are set off in this example. This gives the simplified version*

$$h^{\text{add}}_{j,k}(x) = \sigma_{\text{hard}}(x_j + x_k) \qquad \text{for } j \le k. \tag{2.8}$$

This is illustrated on the left-hand side of Figure 2 for $j < k$.

The right-hand side of Figure 2 shows a linear additive interaction network with the first component x_j being scaled with $1/3$. We note that these linear additive interaction networks are rather similar to binary decision tree partitions of the feature space. We allow for additional flexibility by letting the interaction network f_θ be nonlinear and by letting the interaction token $e_{j,k}$ learn different pairwise interaction structures. It is precisely this ability to learn to modify the inputs that gives the pairwise interaction layers their power.

2.3 Pairwise interaction network

We are now ready to define the tree-like PIN.

Definition 2.3. *Choose an output activation function g. The PIN) is defined by $f_{\text{PIN}} : \mathcal{X} \to \mathbb{R}$*

$$x \mapsto f_{\text{PIN}}(x) = g\left(\sum_{1\le j\le k\le q} w_{j,k}\, h_{j,k}(x) + b\right), \tag{2.9}$$

with learnable (real-valued) output weights $(w_{j,k})_{1\le j\le k\le q}$ and bias $b \in \mathbb{R}$.

There is structural similarity between a PIN and a classical gradient boosting machine (GBM)

$$f_{\text{GBM}}(x) = \sum_{m=1}^{M}\sum_{l=1}^{L} \gamma_{m,l}\, \mathbb{1}_{R_{m,l}}(x),$$

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

with $M \in \mathbb{N}$ GBM iterations considering binary split decision trees with L leaves giving the partition $(R_{m,l})_{l=1}^{L}$ of the feature space $\mathcal{X}$ and assigning the leaf values $(\gamma_{m,l})_{l=1}^{L}$ to this partition. We start by discussing the structural correspondence between the two regression functions.

GBM component	PIN analog
Indicators $\mathbb{1}_{R_{m,l}}(x)$	Bounded activation $\sigma_{\text{hard}}(f_\theta(\cdot))$
Leaf values $\gamma_{m,l}$	Output weights $w_{j,k}$
Tree structure	Network f_θ with interaction tokens $e_{j,k}$

We list some key differences:

1. *Smooth activation*: The hard binary decisions $\mathbb{1}_{R_{m,l}}(x)$ are replaced by continuous functions involving the hard sigmoid function.
2. *Learned representations*: There is an automatic feature transformations through embeddings resulting in the tensor $\boldsymbol{\phi}$, whereas GBMs split on the original feature values.
3. *Parameter sharing*: Common network f_θ across all interactions, and the interaction tokens learn the differences between the pairwise interactions (ϕ_j, ϕ_k).
4. *Joint optimization*: All components trained simultaneously, where as the GBM learns the parameters recursively in stage-wise adaptive way.

The other close connection of the PIN is to a GA^{2}M. A GA^{2}M is obtained from a GAM by adding interaction terms to the regression function

$$g^{-1}(\mu(X)) = b + \sum_{j=1}^{q} f_j(X_j) + \sum_{1 \le j < k \le q} f_{j,k}(X_j, X_k), \tag{2.10}$$

where f_j and $f_{j,k}$ are selected from a given class of splines; see Lou *et al.* (2013). A classical GAM only considers the terms $(f_j)_{j=1}^{q}$, and the GA^{2}M adds the pairwise interaction terms $(f_{j,k})_{1 \le j < k \le q}$. We note that this GA^{2}M (2.10) has the same structure as our PIN (2.9), the difference is that we replace the splines $(f_j)_{j=1}^{q}$ and $(f_{j,k})_{1 \le j < k \le q}$ by pairwise interaction layers $(h_{j,k})_{1 \le j \le k \le q}$ being based on FNNs.

2.4 Connection to graph neural networks

The pairwise interaction structure of the PIN can be naturally interpreted through the lens of GNNs. We formalize this connection and demonstrate how our model can be viewed as a special case of message passing on a complete graph with learned edge features.

Definition 2.4 (feature graph). *For input $x \in \mathcal{X}$, we define the corresponding feature graph $G_x = (V, E)$ as follows:*

- *Vertices $V = \{1, \ldots, q\}$ correspond to feature components;*
- *Edges $E = \{(j,k) : 1 \le j \le k \le q\}$ form a complete graph;*
- *Node features $\phi_j = \phi_j(x_j) \in \mathbb{R}^d$ for $j \in V$;*
- *Edge features $e_{j,k} \in \mathbb{R}^{d_0}$ for $(j,k) \in E$.*

Our PIN can be reformulated as a message passing neural network with a single update step.

Definition 2.5 (message function). *The message function $\mathcal{M} : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^{d_0} \to \mathbb{R}$ is defined as*

$$\mathcal{M}(\phi_j, \phi_k, e_{j,k}) = \sigma_{\text{hard}}(f_\theta(\phi_j, \phi_k, e_{j,k})),$$

where f_θ is the shared interaction network (2.5).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

It is obvious that the PIN (2.9) is equivalent to the GNN defined by

$$f_{\mathrm{GNN}}(x) = g\left(\sum_{(j,k)\in E} w_{j,k}\, \mathcal{M}(\phi_j, \phi_k, e_{j,k})\right).$$

Our architecture differs from traditional GNNs in the following ways:

1. *Edge-centric*: While most GNNs focus on learning node representations, our model primarily focuses on edge interactions and the interaction token $e_{j,k}$ to distinguish different interaction patterns.
2. *Learned edge features*: The interaction tokens $e_{j,k}$ are learned parameters rather than derived features.
3. *Single-pass architecture*: Unlike typical GNNs that use multiple layers of message passing, our model uses a single pass with a more expressive message function.
4. *Bounded messages*: The hard sigmoid activation ensures bounded message values, similar to the binary nature of GBM binary decision tree splits.

Having this connection to GNNs provides the following additional theoretical insights.

Proposition 2.6 (permutation invariance). *For any permutation π of the feature indices, the model output is invariant up to a reindexing of the output weights $w_{j,k}$.*

This proposition immediately follows by analyzing permutations of the components/vertices $V = \{1, \dots, q\}$.

Proposition 2.7 (expressiveness). *The PIN can express any message-passing GNN with binary messages, single-layer architecture and complete graph structure.*

Using GNN operations, the GNN perspective suggests several natural extensions to our base model.

Definition 2.8 (multi-layer PIN). *We can define a K-layer model through iterative message passing, that is,*

$$\phi_v^{(k+1)} = \mathrm{UPDATE}\left(\phi_v^{(k)}, \sum_{u\in\mathcal{N}(v)} \mathcal{M}^{(k)}\big(\phi_u^{(k)}, \phi_v^{(k)}, e_{u,v}\big)\right) \tag{2.11}$$

where $\phi_v^{(0)} = \phi_v(x_v)$, $v \in V$, $k = 0, \dots, K-1$, and $\mathcal{N}(v)$ are the vertices adjacent to v. The UPDATE *can be various functions such as the maximum or the mean of the two components.*

Remark 2.9 (attention mechanism). The model can be enhanced with an attention mechanism by replacing the fixed output weights $w_{j,k}$ with learned functions of the node features

$$w_{j,k} = \alpha_\vartheta(\phi_j, \phi_k),$$

where α_ϑ is a (self-)attention layer with parameters ϑ; see Vaswani *et al.* (2017). If we apply the attention weights only to pairs $j < k$, and if we apply a thresholding, we obtain

$$x \mapsto g\left(b + \sum_{j=1}^{k} w_j\, h_{j,j}(x) + \sum_{1\le j<k\le q} \mathrm{ReLU}(\alpha_\vartheta(\phi_j, \phi_k) - \tau)\, h_{j,k}(x)\right), \tag{2.12}$$

for an attention threshold $\tau \in \mathbb{R}$. The thresholding in (2.12) gives a natural way of pairwise interaction selection.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

The connection to GNNs not only provides theoretical insights but also suggests practical extensions and implementation strategies for the PIN architecture. Moreover, for efficient implementations, existing GNN libraries are available that allow for batched processing (graph batching for parallel processing), optimized GNN kernels on GPUs, and optional sparsification of the interaction graph, similar to (2.12).

3. Claims frequency example

We present a first example that we take in several steps to illustrate the functioning of the tree-like PIN. We consider the popular French MTPL claims frequency dataset available from Dutang *et al.* (2024). We apply the same data cleaning procedure as described in Wüthrich and Merz (2023);[1] for a detailed discussion of the French MTPL dataset, we refer to Appendix B of the latter reference.

Our goal is to predict the MTPL claims frequencies $Y_i = N_i/v_i$ of the insurance policyholders $i \in \{1, \ldots, n\}$, where N_i is the number of claims of policyholder i, $v_i > 0$ is the time exposure (in yearly units) of that policyholder, and x_i describes their features. We fit different tree-like PINs to this data by using the Poisson deviance (PD) loss as the objective function; the PD loss is the canonical strictly consistent loss function choice for mean estimation for claims count problems in insurance; Gneiting and Raftery (2007) and Wüthrich *et al.* (2025). For given learning data $\mathcal{L} = (Y_i, x_i, v_i)_{i=1}^n$, we aim at minimizing the following (average) PD loss to find the optimal weights of the tree-like PIN

$$L_{\mathrm{PD}}(\mathcal{L}, f_{\mathrm{PIN}}) = \frac{1}{n}\left[\sum_{i:\, Y_i=0} 2v_i f_{\mathrm{PIN}}(x_i) + \sum_{i:\, Y_i\neq 0} 2v_i \left(f_{\mathrm{PIN}}(x_i) - Y_i - Y_i \log\left(\frac{f_{\mathrm{PIN}}(x_i)}{Y_i}\right)\right)\right], \quad (3.1)$$

where $f_{\mathrm{PIN}}(\cdot)$ is the PIN given in (2.9), and we select the exponential output function $g(\cdot) = \exp(\cdot)$, which is the canonical link of the Poisson model.

For model training (minimizing the PD loss L_{PD} in the network weights and biases), we use the Adam optimizer on batches of size 128, with an initial learning rate of 0.001. We then apply a learning rate reduction on plateau (factor 0.9, patience 5), and early stopping is exercised based on a 10% validation set of the learning data $\mathcal{L}$; the size of the learning data is $n = 610,206$. Moreover, we hold an independent test sample $\mathcal{T} = (Y_j, x_j, v_j)_{j=1}^m$ of sample size $m = 67,801$, that is going to be used for out-of-sample model testing and model comparison. The learning and test samples, $\mathcal{L}$ and $\mathcal{T}$, are identical to Brauer (2024) and Richman *et al.* (2025), which makes all results directly comparable to these references.

The case of a linear PIN, with two continuous features only.
Similar to Example 2.2, we start with a linear PIN, and we only consider the two continuous features "driver age" (DrivAge) and "log-Density". Moreover, we only consider the interaction term $h_{1,2}(x)$ and we drop the diagonal terms $h_{1,1}(x)$ and $h_{2,2}(x)$ by setting $w_{1,1} = w_{2,2} = 0$. This allows us to nicely illustrate the results. For this linear off-diagonal PIN, we select the (single, $q = 2$) pairwise interaction

$$h_{1,2}^{\mathrm{linear}}(x) = \sigma_{\mathrm{hard}}\big(W^{(1)}(x_1, x_2, e_{1,2})^\top + b^{(1)}\big).$$

This only considers the most inner operation of the shared interaction network (2.5), with embedding dimension $d = 1$, interaction token dimension $d_0 = 1$ and output dimension $d_1 = 1$. This gives us a weight matrix $W^{(1)} \in \mathbb{R}^{1\times 3}$ and a bias $b^{(1)} \in \mathbb{R}$. This is similar to the additive interaction (2.8), but additionally allowing for an affine transformation. Note that in this example of only one

[1] The cleaned data are available from https://aitools4actuaries.com/.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

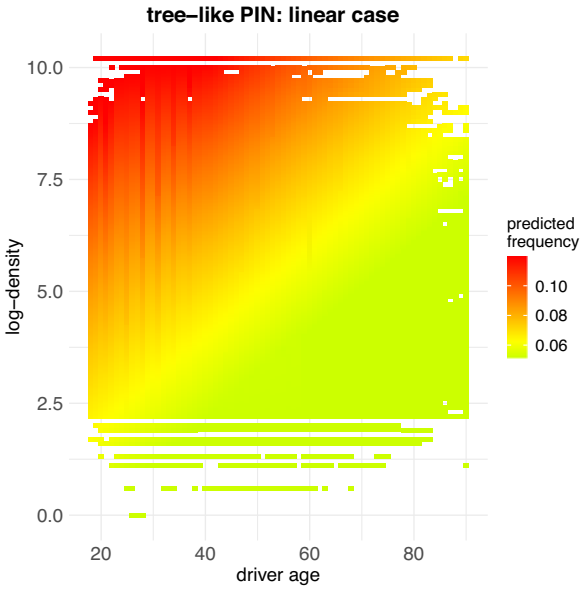


Figure 3. Heatmap of predicted frequencies of the linear off-diagonal PIN only considering the features "DrivAge" and "log-density".

interaction pair ("DrivAge", "log-Density"), the interaction token $e_{1,2}$ would not be necessary. By inserting this linear interaction network into the PIN in (2.9) gives us a first illustrative example.

We fit this linear off-diagonal PIN to the learning data $\mathcal{L}$ using the PD loss (3.1), and the result is presented in Figure 3. The colored dots show the estimated frequencies on the insurance portfolio (for the white parts there are no insurance policies available for these feature combinations). We observe that this 1-dimensional ($d = 1$) linear off-diagonal PIN suggests a diagonal pattern. Related to decision trees, this corresponds to a feature space partition along the diagonal (for driver age and log-density scaled as in Figure 3). In the next example, we are going to refine this prediction by allowing for more general functional forms.

Tree-like PIN, with two continuous features only.
In our next example, we consider the non-linear tree-like PIN as introduced in (2.9). To improve the modeling step-by-step on the previous example, we again only consider the two features "DrivAge" and "log-Density" and we let them interact nonlinearly, and again we only consider the off-diagonal interaction term $h_{1,2}(x)$ by setting $w_{1,1} = w_{2,2} = 0$. We start with the 1-dimensional embedding case $d = 1$, thus, we consider 1-dimensional FNN embeddings $x_j \mapsto \phi_j(x_j) \in \mathbb{R}$, which we then let interact by the interaction network as defined in (2.5); note that because we only have two features in this example, $j \in \{1, 2\}$, this network is not shared by different interactions and the interaction token is not necessary.

We fit this (non-linear) off-diagonal PIN as described above by using the Adam optimizer and early stopping. The result of this 1-dimensional case with two feature components is shown in Figure 4 (lhs), and it is compared to a GLM considering the same two features "DrivAge" and "log-Density", the first variable is encoded categorically and the second one continuously in this GLM, see Wüthrich and Merz (2023, Section 5.2.4) for more details. The GLM in the right-hand side of Figure 4 shows a multiplicative structure in the two feature components (because of the log-link choice), and the PIN on the left-hand side gives a refinement indicating that the vertical age class coloring of the GLM is not fully appropriate. From these two plots, we also observe that the expected frequency is non-monotone in the age of driver variable.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

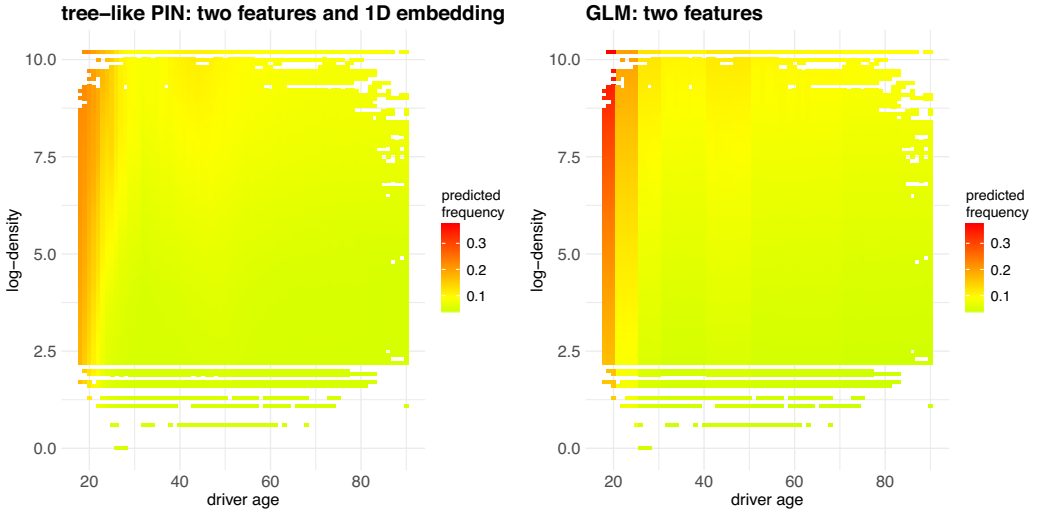


Figure 4. Heatmaps of predicted frequencies of (lhs) the off-diagonal PIN with 1D embedding ($d = 1$), and (rhs) GLM only considering the two features "DrivAge" and "log-density".

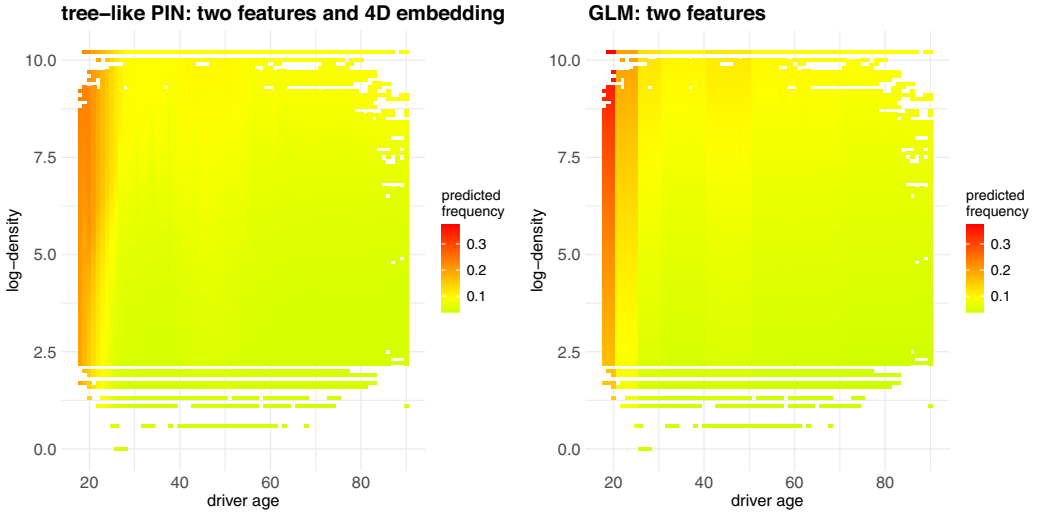


Figure 5. Heatmaps of predicted frequencies of (lhs) the off-diagonal PIN with 4D embedding ($d = 4$), and (rhs) GLM only considering the two features "DrivAge" and "log-density".

In a next step, a further refinement is received by increasing the dimension of the embedding to $d = 4$ in $x_j \mapsto \phi_j(x_j) \in \mathbb{R}^d$. Similarly to GBMs, this overlaps multiple tree-like structures. Figure 5 (lhs) shows the resulting fitted off-diagonal PIN, which looks like a refinement of the GLM on the right-hand side, also more clearly reflecting the expected frequency increase in the driver age bucket between 40 and 50. Note that this age bucket has not been indicated to the PIN, but this finding of a higher frequency in this age bucket is purely data driven. Out-of-sample on the test data $\mathcal{T}$, the PIN has a slightly better performance than the GLM w.r.t. the PD loss; we do not show the figures because in this example we only showcased the functioning of the tree-like PIN on two feature components and only in the off-diagonal case ($w_{1,1} = w_{2,2} = 0$); the general case is presented next.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

Table 1. Selected PIN architecture used on the French MTPL dataset

Module	# Weights
Embedding dimension $d = 10$:	
Continuous features (7) with $d' = 20$	1, 750
Categorical features (2) with $n_j = 11, 22$	330
Pairwise interaction layer:	
Interaction tokens ($10 \times 9/2$) with $d_0 = 10$	450
1st FNN layer with $d_1 = 30$	930
2nd FNN layer with $d_2 = 20$	620
Output layer	21
Output:	
Output weights $(w_{j,k})_{j \le k}$ including biases	46
Total	4, 147

Tree-like PIN on all feature components.
We now include all available feature components and we also consider the diagonal elements of the pairwise interaction layer (2.7). There are seven continuous and binary input variables to which we apply separate network embeddings (2.2), and two categorical variables to which we apply two entity embeddings (2.1). We select a hidden layer of dimension $d' = 20$ in (2.2), and an entity embedding dimension of $d = 10$. Regarding the binary input variables, in a standard GLM framework, treating them as dummy coded factors or as continuous covariates is equivalent. In deep learning models, binary variables are often incorporated directly as continuous inputs. This parameterization yields in our case $2d' + (d' + 1)d = 250$ parameters. By contrast, using entity embeddings would require only $2d = 20$ parameters. Therefore, if model complexity is a concern, the embedding-based approach is preferable.

This gives us a tensor $\boldsymbol{\phi} = \boldsymbol{\phi}(x) \in \mathbb{R}^{10 \times 9}$, and in total $10 \times 9/2 = 45$ interaction tokens $e_{j,k} \in \mathbb{R}^{10}$, $1 \le j \le k \le 9$; we select dimension $d_0 = 10$ for the interaction tokens. The interaction tokens are randomly initialized using the uniform initializer provided by `Keras`, with values sampled from the interval $(-0.05, 0.05)$. Finally, for the shared interaction network (2.5), we select hidden layer dimensions $d_1 = 30$ and $d_2 = 20$. Table 1 summarizes the selected PIN architecture, indicating the number of weights/parameters to be fitted.

This fully specifies the selected PIN architecture, which is now trained on the learning sample $\mathcal{L}$ using the PD loss (3.1) as the objective function. Because early stopped stochastic gradient descent (SGD) training involves several elements of randomness, the presented results are averages over 10 SGD runs using different seeds (starting points) for the SGD algorithm, see also Wüthrich *et al.* (2025, Section 5.3.8). That is, the loss figures are averages over 10 different fits (and in brackets we provide the observed standard deviations across these 10 runs). Moreover, we ensemble the predictors (by averaging over the predictors). This ensembling typically improves the predictive model, which is also verified in our PIN architecture, all these results are presented in Table 2.

Let us discuss the results of Table 2. The benchmark models are taken from Wüthrich and Merz (2023, Table 7.9), Brauer (2024, Tables 2 and 4) and Richman *et al.* (2025, Table 2). The null model does not consider any features, but it is just based on the empirical mean. Poisson GLM3 is a feature-engineered GLM having 50 parameters; this model is linear on the canonical scale and does not consider any interaction terms. The linearity of the GLM is challenged by the GAM using natural cubic splines. The complexity of this GAM cannot be expressed in terms of the number of parameters involved, but the GAM routine in R determines the effective degrees of freedom (edf) using generalized cross-validation. The resulting edf in our example are 66.7,

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

Table 2. Number of parameters, in-sample and out-of-sample Poisson deviance losses (units are in 10^{-2}); benchmark models are taken from Wüthrich and Merz (2023, Table 7.9), Brauer (2024, Tables 2 and 4) and Richman et al. (2025, Table 2); for the ensemble models we state the number of parameters of a single model because each one is trained individually. The best out-of-sample performance is highlighted in bold

Model	#Param.	In-sample Poisson loss	Out-of-sample Poisson loss
Null model (intercept-only) (Wüthrich & Merz, 2023)	1	25.213	25.445
Poisson GLM3 (Wüthrich & Merz, 2023)	50	24.084	24.102
Poisson GAM	(66.7)	23.920	23.956
Plain-vanilla FNN (Wüthrich & Merz, 2023)	792	23.728 (±0.026)	23.819 (±0.017)
Ensemble plain-vanilla FNN (Wüthrich & Merz, 2023)	792	23.691	23.783
CAFFT (Brauer, 2024)	27,133	23.715 (±0.047)	23.807 (±0.017)
Ensemble CAFFT (Brauer, 2024)	27,133	23.630	23.726
Credibility Transformer (Richman *et al.*, 2025)	1,746	23.641 (±0.053)	23.788 (±0.040)
Ensemble Credibility Transformer (Richman *et al.*, 2025)	1,746	23.562	23.711
Tree-like PIN	4,147	23.593 (±0.046)	**23.740** (±0.025)
Ensemble tree-like PIN	4,147	23.522	**23.667**

which means that suitably chosen natural cubic splines should be able to accommodate with roughly 67 parameters. These three fitted models have been fitted by minimizing the PD loss which is equivalent to perform MLE in the corresponding Poisson claims count model. We conclude that the GAM has the best out-of-sample performance among these three classical actuarial models, resulting in an out-of-sample loss of 23.956×10^{-2}. This is the best benchmark that we can get if we do not consider any interaction terms between the feature components.

If we take a deep (plain-vanilla) FNN architecture of depth 3 with (20, 15, 10) units and with two-dimensional embedding layers for the two categorical covariates, we receive a network architecture with 792 network parameters. If we fit this architecture with an early stopped SGD with 10 initializations, we receive an average out-of-sample loss of 23.819×10^{-2}, and ensembling these 10 networks gives an out-of-sample performance of 23.783×10^{-2}, thus, interaction modeling can clearly improve the GAM.

This deep plain-vanilla FNN architecture has been improved by using different versions of the transformer architecture of Vaswani *et al.* (2017) modified to tabular input data. The powerful adoptions of Brauer (2024, Table 4) and Richman *et al.* (2025, Table 2) provide out-of-sample losses of 23.726×10^{-2} and 23.711×10^{-2}, respectively, for the ensemble versions. From Table 2, we conclude that our tree-like PIN proposal outperforms all these other methods. Thus, this data example verifies the superior predictive performance of the tree-like PIN over some of the most recent proposals in the literature. On top, as will be illustrated in the remainder of this article, it also gives us an intrinsic explainability, in contrast to most other network architectures.

Figure 6 shows the resulting heatmaps of the predicted frequencies, with driver's age on the x-axis and log-density on the y-axis; this is a two-dimensional marginal plot (M plot); see Apley and Zhu (2020) and Wüthrich *et al.* (2025, Section 9.3.4). Remark that we do not expect a smooth graph because the predicted frequencies were computed on nine feature components, and the graphs show a projected version to two of them (M plot), i.e., these are marginalized plots. For example, for high driver's age, we see some sprinkling of red dots that can only be explained by analyzing the other features, e.g., it may be severely impacted by a high bonus-malus level;

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

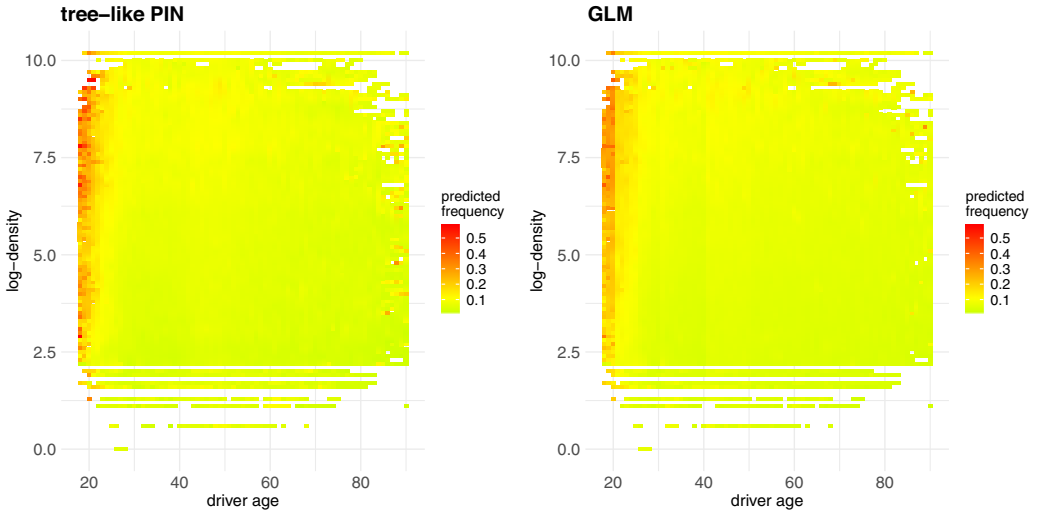


Figure 6. Heatmaps of predicted frequencies of (lhs) tree-like PIN and (rhs) GLM on all available features.

remark also that the portfolio is comparably sparse for large driver's age values, that is why extreme expected frequency values do not average out.

4. Model explainability

4.1 PIN interaction importance

A notoriously difficult problem in machine learning models is explainability. Scholars have developed various post hoc measures of variable importance, e.g., permutation variable importance by Breiman (2001) and Fisher *et al.* (2018), or SHAP importance of Lundberg and Lee (2017) and Mayer and Stando (2025) are two popular examples. When it comes to interactions, things become even more difficult. There is Friedman and Popescu (2008)'s H^2-statistics which is a post hoc explainability tool that measures the impact of bivariate components on the regression function (by partial dependence functions). Apart from that, there is little guidance on how to measure the interaction strengths at a reasonable computational effort, i.e., testing pairwise interactions can be very costly.

Coming back to the PIN (2.9), we modify this function in a way so that we do not aim at getting the maximal predictive power, but we try to evaluate which shared interaction unit $h_{j,k}(x)$ contributes most to a loss decrease. For this, we consider the multi-output mapping

$$x \mapsto (f_{\text{PIN},j,k})_{1\le j<k\le q} \in \mathbb{R}^{q(q-1)/2}, \tag{4.1}$$

where for $1 \le j < k \le q$ we set

$$f_{\text{PIN},j,k} = g\left(b + \sum_{l=1}^{q} w_{l,l}\, h_{l,l}(x) + b_{j,k} + w_{j,k}\, h_{j,k}(x)\right),$$

for biases $b_{j,k} \in \mathbb{R}$ and the remaining terms have the same structure as in (2.9). The first part of this multi-output regression function captures the main effects of the covariate components, this corresponds to the diagonal of the tensor (2.7). That is, the main effects are given by

$$b + \sum_{l=1}^{q} f_l(X_l) = b + \sum_{l=1}^{q} w_{l,l}\, h_{l,l}(x), \tag{4.2}$$

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

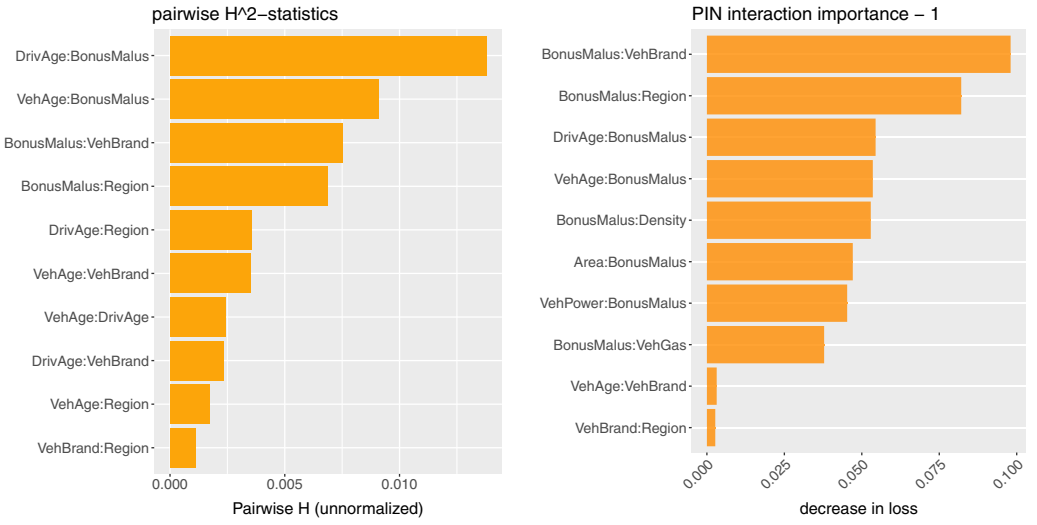


Figure 7. (lhs) Friedman and Popescu (2008)'s H^2-statistics, and (rhs) PIN interaction importance.

where we make the link to the GAM (2.10). The remaining terms $h_{j,k}(x)$ in (4.1), $j < k$, correspond to the off-diagonal terms of the pairwise interaction layer (2.7). Each component in (4.1) adds another interaction term to the diagonal (non-interacting) terms (4.2), and we can determine the term $h_{j,k}(x)$, $j < k$, that leads to the biggest decrease in out-of-sample loss when added to (4.2). The great thing is that we can fit all $q(q-1)/2$ interaction terms in the *same* gradient descent run, i.e., we apply the objective function for model fitting (simultaneously) to all components of (4.1), and jointly fit all components by just aggregating these component-wise losses

$$L_{\text{AGGR}}(\mathcal{L}, (f_{\text{PIN},j,k})_{j<k}) = \frac{1}{q(q-1)/2} \sum_{1 \le j < k \le q} L_{\text{PD}}(\mathcal{L}, f_{\text{PIN},j,k}),$$

the latter is the average PD loss (3.1). This is possible here because we have isolated all interaction pairs in the pairwise interaction layer (2.7).

In our example, we consider one further simplification, namely, we replace the diagonal (non-interacting) terms (4.2) by the fitted GAM terms that are considered to be optimal under the exclusion of interactions, see Table 2. This fitted GAM is kept fixed (frozen) and only the off-diagonal interaction terms in (4.1) are trained with SGD. The training of this multi-output PIN takes approximately 15 minutes on an ordinary laptop.[2] We then compute the decrease in out-of-sample loss compared to the GAM for each individual interaction term, $j < k$. Figure 7 (rhs) shows the results (ranked by size). The biggest decrease in loss compared to the GAM can be achieved by adding an interaction term for (BonusMalus:VehBrand), and the second biggest decrease is obtained by adding (BonusMalus:Region).

Figure 7 (lhs) illustrates the Friedman and Popescu (2008) H^2-statistics computed for the full PIN model. In our experiments, this procedure requires around 90 minutes (typically one chooses a small background dataset to speed up computations). Not surprisingly, the same feature interactions seem to be the most impactful ones, but the order of the most important interactions is different. Friedman–Popescu's H^2-statistics studies the partial dependence function in a fully fitted model (which can be seen as a top-down view – backward selection), whereas our PIN approach considers which interaction should be added to have the biggest impact on the prediction accuracy (which is more of a bottom-up consideration – forward selection). Thus, we can

[2]Intel Core i7-1355U Processor (12M Cache, up to 5.00 GHz).

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

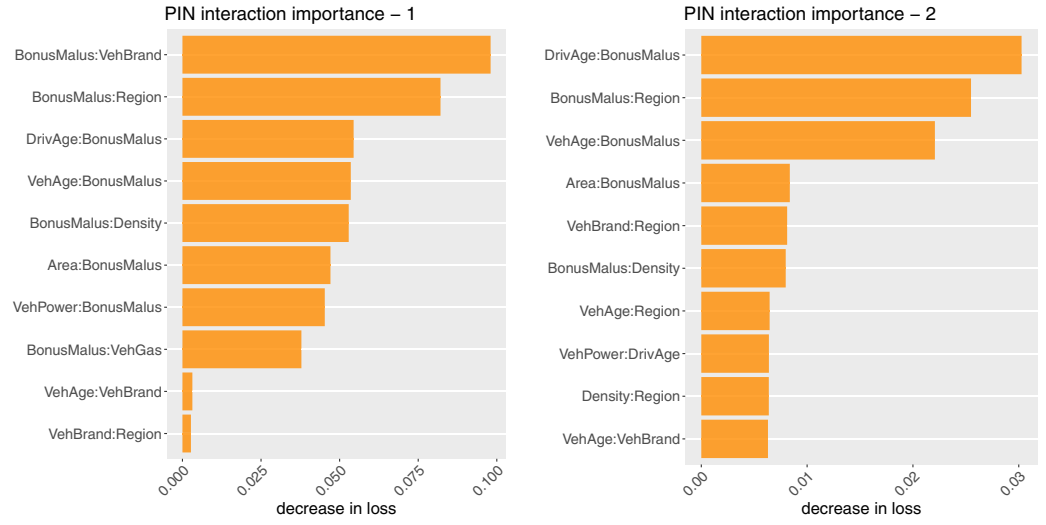


Figure 8. (lhs) PIN interaction importance – 1st round (this plot is identical to Figure 7, rhs), and (rhs) PIN interaction importance – 2nd round.

interpret this as a forward selection of interaction terms, similar to the forward selection model building in GLMs. Moreover, Friedman–Popescu's H^2-statistics is based on partial dependence plots (PDPs), which have the weakness that they are not necessarily reliable in cases where the feature components are highly correlated, because PDPs extrapolate the regression function to rare or impossible covariate combinations (e.g., car drivers aged 20 with 10 years of driving experience), where the regression function is typically poorly estimated or undefined, respectively. Not surprisingly, this weakness also translates to the H^2-statistics. Our method does not suffer from this weakness, since the contribution of each pairwise interaction is evaluated on the empirically observed portfolio – on the empirical population distribution – and assessing each interaction independently of all others. As a consequence, the importance measure of each pairwise interaction remains meaningful even in the presence of feature correlations. Naturally, as is also common in the GLM forward selection process, if there are two highly correlated feature components that interact with a third component, after including the first strong interaction, the second one may no longer be needed. We discuss this next.

Similar to the method of forward selection in GLMs, we add the most significant interaction term (BonusMalus:VehBrand) to the diagonal regression model (4.2). We also freeze this interaction term, and we repeat the same fitting strategy to find the second most important interaction term, after (BonusMalus:VehBrand) has been added to the model. The result of this second round of forward selection is shown in Figure 8 (rhs), and it is compared to the first round results on the left-hand side. The second interaction that should be added is (DrivAge:BonusMalus). This gives a systematic way of model building that may also be useful in improving GLMs and GAMs; this is similar to the forward selection algorithm of GLM model building. Note that this forward selected model should be refitted simultaneously on all included terms, because in this forward inclusion process, we always freeze the previously selected terms, and an additional re-fitting (fine-tuning) step may find a more optimal jointly learned model on the selected pairwise interaction components.

4.2 SHapley's Additive exPlanation

There is another very remarkable property of the tree-like PIN architecture, namely, it allows for a very efficient computation of SHapley's Additive exPlanation (SHAP) as introduced by Lundberg

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

and Lee (2017). Shapley values originate from cooperative game theory. They are used to share a total payoff of a cooperative game in a fair way among q players; see Shapley (1953). Denote by $\mathcal{C} \subseteq \mathcal{Q} = \{1, \ldots, q\}$ any coalition of q players. Based on Shapley's fairness axioms, there is precisely one way to allocate the total payoff to the q players of the cooperative game, resulting in the Shapley values ψ_j assigned to players $j \in \mathcal{Q}$, given by

$$\psi_j = \frac{1}{q} \sum_{\mathcal{C} \subseteq \mathcal{Q} \setminus \{j\}} \binom{q-1}{|\mathcal{C}|}^{-1} (\nu(\mathcal{C} \cup \{j\}) - \nu(\mathcal{C})), \qquad \text{for all } j \in \mathcal{Q}, \tag{4.3}$$

for a given value function $\mathcal{C} \mapsto \nu(\mathcal{C})$. In fact, starting from the given value function ν, the Shapley values $(\psi_j)_{j=1}^q$ are the unique solution to a set of four axioms called Shapley's fairness axioms; see Shapley (1953) and Lundberg and Lee (2017).

There are two equivalent reformulations of the Shapley values (4.3), namely, there is the kernel SHAP formulation of Lundberg and Lee (2017, Theorem 2), and there is the permutation SHAP version of Štrumbelj and Kononenko (2010, 2014). In this work, we focus on the permutation SHAP formulation. Denote by $\pi = (\pi_1, \ldots, \pi_q)$ a permutation of the ordered set $(1, \ldots, q)$. Let $\kappa(j) \in \mathcal{Q}$ be the index defined by $\pi_{\kappa(j)} = j$ and set

$$\mathcal{C}_{\pi,j} = \{\pi_1, \ldots, \pi_{\kappa(j)-1}\} \subset \mathcal{Q}. \tag{4.4}$$

These are all components of π preceding $\pi_{\kappa(j)} = j$; for $\kappa(j) = 1$, this is the empty set $\emptyset$. The permutation SHAP version of Štrumbelj and Kononenko (2010, 2014) proves that the Shapley values (4.3) can equivalently be computed by

$$\psi_j = \frac{1}{q!} \sum_{\pi} \nu(\mathcal{C}_{\pi,j} \cup \{j\}) - \nu(\mathcal{C}_{\pi,j}), \tag{4.5}$$

where the summation runs over all permutations π. These are $q!$ permutations π, which can be an extremely large number making Shapley values computationally unattainable for large q. However, there is a nice result which says that if the value function ν only includes interactions of maximal order two, then an arbitrary permutation $\pi = (\pi_1, \ldots, \pi_q)$, and its reverted version $\rho(\pi) = (\pi_q, \ldots, \pi_1)$ is sufficient to compute the Shapley values

$$\psi_j = \frac{1}{2}(\nu(\mathcal{C}_{\pi,j} \cup \{j\}) - \nu(\mathcal{C}_{\pi,j}) + \nu(\mathcal{C}_{\rho(\pi),j} \cup \{j\}) - \nu(\mathcal{C}_{\rho(\pi),j})), \tag{4.6}$$

if the value function ν is a bilinear form; see Mayer and Wüthrich (2025, Proposition 3.1). For example, the permutation $\pi = (1, \ldots, q)$ and its reverted version $\rho(\pi) = (q, \ldots, 1)$ do the job.

For explaining machine learning predictions, one typically works on the link scale, i.e., one drops the link function g in (2.9). For a given feature x, this motivates the total value of the grand coalition $\mathcal{Q}$

$$\nu(\mathcal{Q}) := \nu_x(\mathcal{Q}) := g^{-1}(f_{\text{PIN}}(x)) = \sum_{1 \le j \le k \le q} w_{j,k}\, h_{j,k}(x) + b.$$

This is the prediction made by $x = (x_1, \ldots, x_q)^\top$, and we want to explain how the feature components x_j contribute to this prediction. This requires the choice of the value function $\nu(\mathcal{C}) = \nu_x(\mathcal{C})$ for any coalition $\mathcal{C} \subset \mathcal{Q}$ if the elements in $\mathcal{Q} \setminus \mathcal{C}$ are masked (not available for prediction). The most popular choice for describing the masking is the so-called interventional SHAP version, in our case given by

$$\nu(\mathcal{C}) := \nu_x(\mathcal{C}) := \mathbb{E}\left[\sum_{1 \le j \le k \le q} w_{j,k}\, h_{j,k}(x_{\mathcal{C}}, X_{\mathcal{Q} \setminus \mathcal{C}}) + b \right], \tag{4.7}$$

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

where the components $x_{\mathcal{C}} = (x_j)_{j\in\mathcal{C}}$ are kept fixed and the components $X_{\mathcal{Q}\setminus\mathcal{C}} = (X_j)_{j\in\mathcal{Q}\setminus\mathcal{C}}$ are resampled from the population distribution $X \sim \mathbb{P}$. An important point in the subsequent computation of the Shapley values is that each term $h_{j,k}$ in (4.7) only depends on the components (j, k), i.e., we only have pairwise interactions, see (2.6). That is, by a slight abuse of notation,

$$h_{j,k}(x_{\mathcal{C}}, X_{\mathcal{Q}\setminus\mathcal{C}}) = \begin{cases} h_{j,k}(x_j, x_k) & \text{if } j,k \in \mathcal{C}, \\ h_{j,k}(X_j, x_k) & \text{if } j \in \mathcal{Q}\setminus\mathcal{C} \text{ and } k \in \mathcal{C}, \\ h_{j,k}(x_j, X_k) & \text{if } j \in \mathcal{C} \text{ and } k \in \mathcal{Q}\setminus\mathcal{C}, \\ h_{j,k}(X_j, X_k) & \text{if } j,k \in \mathcal{Q}\setminus\mathcal{C}. \end{cases}$$

This interventional SHAP formula (4.7) is usually estimated empirically. For this, we select a background set $\mathcal{B}$ from the learning sample $\mathcal{L}$. This allows one to determine the value function empirically by

$$\widehat{v}(\mathcal{C}) = \widehat{v}_x(\mathcal{C}) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\left[\sum_{1\le j\le k\le q} w_{j,k}\, h_{j,k}\left(x_{\mathcal{C}}, X^{(i)}_{\mathcal{Q}\setminus\mathcal{C}}\right) + b\right], \tag{4.8}$$

where $X^{(i)}$ is the i-th instance of the background dataset.

The crucial point now is that the value function $\mathcal{C} \mapsto \widehat{v}(\mathcal{C}) = \widehat{v}_x(\mathcal{C})$ only involves interactions of maximal order two. Therefore, the paired permutation SHAP formula (4.6) applies (and it is exact for the selected value function $\widehat{v} = \widehat{v}_x$). This requires $2(q+1)$ evaluations of the value function $\widehat{v}$, and each value function evaluation requires $|\mathcal{B}|$ computations of PIN predictions, see (4.8). Thus, altogether, the SHAP decomposition of the prediction $g^{-1}(f_{\text{PIN}}(x))$ requires $2(q+1)|\mathcal{B}|$ PIN network computations for computing the *exact* Shapley values $(\psi_j)_{j=1}^q$ of the value function $\widehat{v}$. Any other network architecture that involves higher-order interactions would need $q!\,|\mathcal{B}|$ network computations for receiving the exact Shapley values of the corresponding estimated interventional SHAP value function; thus, the order of computations grows as $q!$ instead of $2(q+1)$ as in our tree-like PIN case.

We revisit the PIN architecture depicted in Table 1 having the best predictive performance of all models presented in Table 2. We emphasize that this network architecture only considers interactions of order two on the link scale; thus, the paired-sampling permutation SHAP formula (4.6) applies, and it gives exact results, for instance, we can use the permutation pair $\pi = (1, \ldots, q)$ and $\rho(\pi) = (q, \ldots, 1)$. As background dataset, we select at random $|\mathcal{B}| = 2,000$ instances from the learning set. This allows us to decompose the predictions $g^{-1}(f_{\text{PIN}}(x))$ of selected instances x. Decomposing 100 predictions takes 23 s on an ordinary laptop (see footnote on page 36).

The left-hand side of Figure 9 shows the Shapley values $(\psi_j)_{j=1}^q$ of a single instance x. We have global mean of the background dataset of $\widehat{v}_x(\emptyset) = -2,776$, and the prediction of feature x is given by $\widehat{v}_x(\mathcal{Q}) = g^{-1}(f_{\text{PIN}}(x)) = -3.239$. Thus, feature x has a prediction being -0.463 below average. The Shapley values $(\psi_j)_{j=1}^q$ illustrated in Figure 9 (lhs) show how this lower prediction can be explained. The biggest decrease is explained by the bonus-malus level and the density features, whereas the vehicle gas feature leads to the biggest increase in the explanation of this difference from $\widehat{v}_x(\emptyset)$ to $\widehat{v}_x(\mathcal{Q})$.

Going from local to global SHAP explanations, we can decompose the predictions $g^{-1}(f_{\text{PIN}}(x_i))$ of many instances $(x_i)_{i\in\mathcal{I}}$, for a randomly selected subset $\mathcal{I}$ of the entire data. This provides us with SHAP decompositions $(\psi_j^{(i)})_{j=1}^q$, which allow us to consider different statistics. The following analysis is based on $|\mathcal{I}| = 1,000$ instances x_i that have been decomposed by the paired-sampling permutation SHAP method. SHAP variable importance is then obtained by considering

$$\overline{\psi}_j = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} |\psi_j^{(i)}|.$$

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

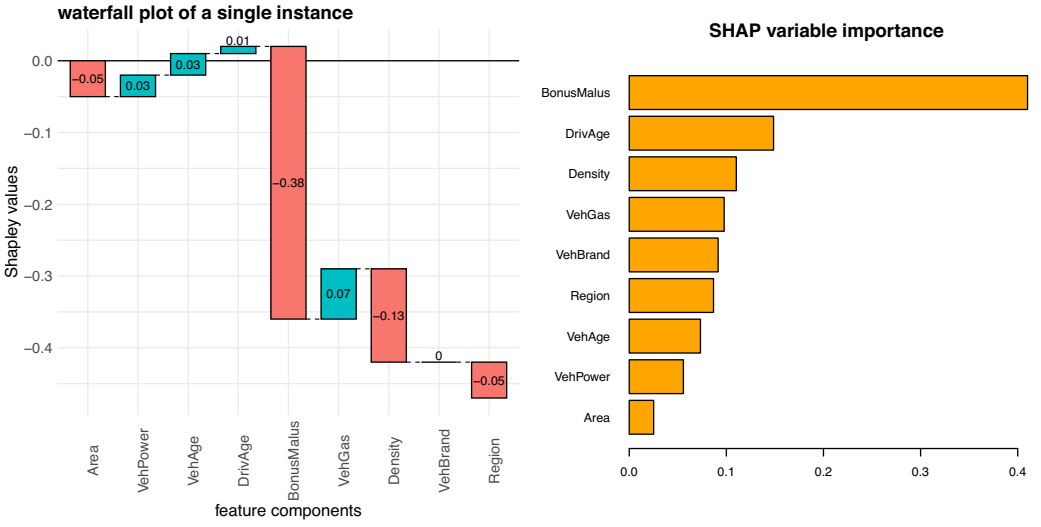


Figure 9. (lhs) SHAP waterfall graph $(\psi_j)_{j=1}^q$ of a single instance x and (rhs) SHAP variable importance.

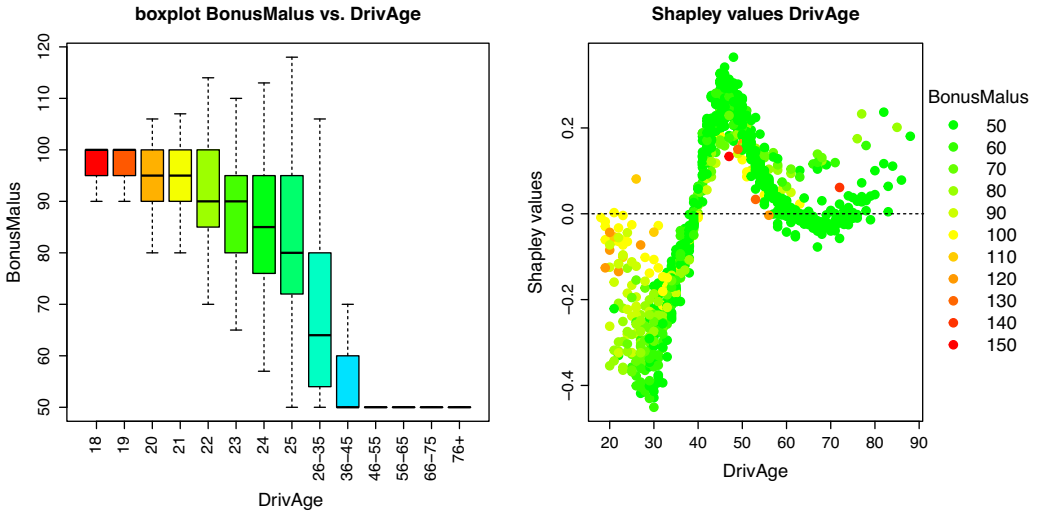


Figure 10. (lhs) Dependence between driver's age and the bonus-malus variable, this graph is taken from Wüthrich and Merz (2023, Figure 13.12), and (rhs) Shapley values $\psi_j^{(i)}$ of the selected 100 instances $i \in \mathcal{I}$ for the covariate $j = \texttt{DrivAge}$; the coloring is the bonus-malus level of these instances.

Figure 9 (rhs) shows these importance measures $\overline{\psi}_j$ for all feature components $j \in \mathcal{Q}$. The most significant role is played by the bonus-malus level (past claims history) followed by the driver's age variable and at the other end is the area code that seems to be the least important variable.

In Figure 10 (rhs), we analyze the 1,000 decomposed instances x_i, $i \in \mathcal{I}$. The plot shows the Shapley values $\psi_j^{(i)}$ of these 1,000 instances for the variable $j = \texttt{DrivAge}$. This driver age variable has its strongest interaction with the bonus-malus level, see Figure 8, and the coloring of the Shapley values in Figure 10 (rhs) is selected according to these bonus-malus levels. We observe that the driver's age variable has a decreasing effect on the prediction for drivers' ages below 40 (negative Shapley values) and an increasing effect for ages between 40 and 60 (positive Shapley values). To fully understand this picture, one needs to understand the interaction with

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

Table 3. Policy-level covariates and their descriptions

Variable	Description
`fuel`	Type of fuel of the vehicle
`coverage`	Insurance coverage level (TPL, TPL+, TPL++)
`fleet`	Indicator for fleet vehicle (yes/no)
`use`	Vehicle use (private or work)
`sex`	Policyholder gender
`bm`	Bonus–malus level
`ageph`	Policyholder age
`long`	Longitude of policyholder municipality
`lat`	Latitude of policyholder municipality
`power`	Vehicle power in kilowatts
`agec`	Vehicle age

the bonus-malus level, and we observe for drivers' ages below 40, that this positive age effect especially holds for low bonus-malus levels. In fact, there is a strong dependence between the driver's age variable and the bonus-malus level for drivers' ages below 40, see boxplot of Figure 10 (lhs), because new drivers start at a bonus-malus level of 100 (yellow color in Figure 10, rhs). This level decreases in every year that the driver is free of accidents, and otherwise it increases. Therefore, young drivers typically have a higher bonus-malus level than mature drivers (in a low-frequency environment), and the Shapley values in Figure 10 (rhs) tell us that young drivers with a lower bonus-malus level (green colors) get a decrease compared to their older peers in the same bonus-malus class. Of course, this interaction makes perfect sense. This completes the French MTPL example.

5. A second real-world data example

To further assess the performance of the proposed tree-like PIN architecture, we consider a second dataset. We select the Belgian MTPL dataset available from Dutang *et al.* (2024), comprising 163,212 individual insurance policies. Also in this case, the aim consists of forecasting claims frequencies conditional on a set of observed risk characteristics. The dataset contains 11 features, and these are detailed in Table 3.

We adopt the same splitting scheme as before, using a 90–10% learning-test split, and we standardize all continuous input features. The same PIN architecture used for the French dataset is employed. The 10 continuous and binary input variables are processed through a separate embedding network, see (2.2), with layer sizes $d' = 20$ and $d = 10$. The single categorical variable (`coverage`) is processed via an entity embedding, see (2.1), with dimension $d = 10$. The application of this tokenization procedure yields a tensor representation $\boldsymbol{\phi} = \boldsymbol{\phi}(x) \in \mathbb{R}^{10\times 11}$, from which pairwise interaction terms are constructed. In total, this results in $10 \times 11/2 = 66$ interaction tokens $e_{j,k} \in \mathbb{R}^{10}$, for $1 \leq j < k \leq 11$. Also in this case, the shared interaction network defined in (2.5) is selected as a two-layer architecture with hidden layer dimensions $d_1 = 30$ and $d_2 = 20$. The comparison is conducted in terms of number of parameters, in-sample and out-of-sample PD losses w.r.t. a selection of benchmark models: Poisson null, Poisson GLM, plain vanilla FNN, and the credibility transformer. For the neural network-based models, performance is evaluated both in terms of average results over 10 SGD runs and ensemble predictions. Table 4 gives the results.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

Table 4. Number of parameters, in-sample and out-of-sample Poisson deviance losses (units are in 10^{-2}) on the Belgian MTPL dataset; for the ensemble models, we state the number of parameters of a single model because each one is trained individually. The best out-of-sample performance is highlighted in bold

Model	# Param.	In-sample Poisson loss	Out-of-sample Poisson loss
Null model (intercept-only)	1	55.071	55.062
Poisson GLM	13	53.494	53.338
Plain-vanilla FNN	1,166	53.357 ($\pm$0.052)	53.168 ($\pm$0.067)
Ensemble plain-vanilla FNN	1,166	53.123	52.976
Credibility transformer	1,646	53.153 ($\pm$0.104)	53.045 ($\pm$0.122)
Ensemble credibility transformer	1,646	53.043	52.934
Tree-like PIN	4,828	53.107 ($\pm$0.100)	**52.927** ($\pm$0.098)
Ensemble tree-like PIN	4,828	53.008	**52.826**

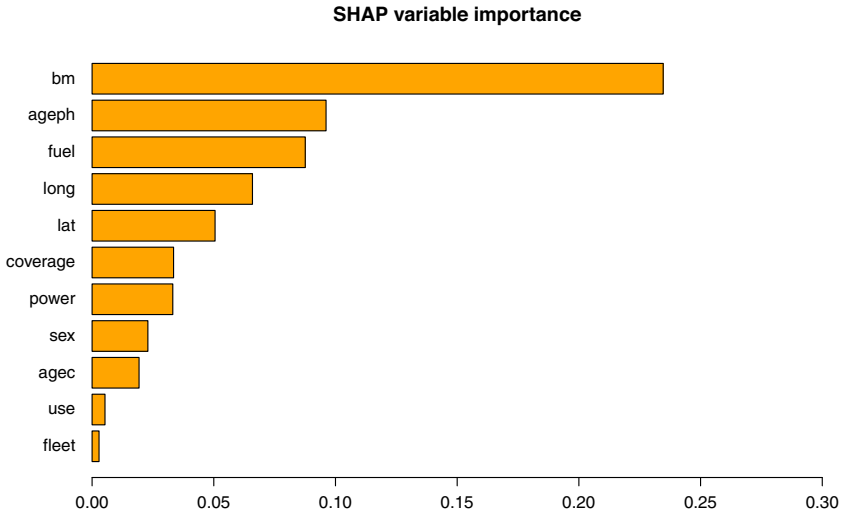


Figure 11. SHAP variable importance on Belgian MTPL dataset.

The tree-like PIN achieves the best out-of-sample predictive performance also on this second dataset, outperforming all other considered benchmark models in terms of PD loss. Moreover, the ensemble version of the tree-like PIN further enhances the predictive accuracy, which appears superior w.r.t. the plain-vanilla FNN and the credibility transformer.

Next, we perform an explainability analysis to identify the features and pairwise feature interactions that most strongly impact the model's predictions. We begin by discussing the SHAP variable importance results. Figure 11 reports a bar plot of SHAP feature importance, showing that the model is mainly driven by the bonus-malus variable (`bm`), similarly to what is observed on the French dataset. The next most influential predictors are `ageph` and `fuel`. Geographic coordinates (`long` and `lat`) and `coverage` produce an intermediate contribution.

Beyond the individual contribution of each covariate, we also investigate the most relevant interactions among covariates. Following the procedure described above, we train the multi-output PIN model defined in (4.1). The objective is to identify the pairwise interactions that lead to the largest reductions in the loss function and, consequently, provide the strongest contribution to the predictive performance. Figure 12 summarizes the outcomes of the two-step selection procedure. The left panel reports the ranking obtained in the first round, where the interaction-free GAM model is frozen and all pairwise interactions are evaluated. The right panel shows the ranking from the second round, where the interaction selected at the first step is frozen as well. In the

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

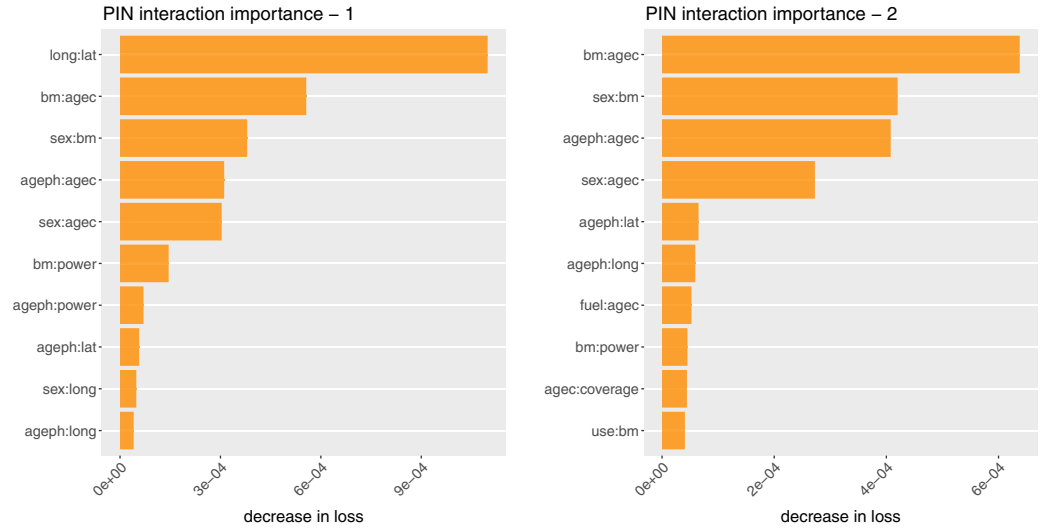


Figure 12. (lhs) PIN interaction importance on Belgian MTPL dataset – 1st round, and (rhs) PIN interaction importance – 2nd round.

first round (lhs), the strongest interaction involves the geographical coordinates (`long` and `lat`). This highlights that, although their marginal effects are not among the most significant ones, their interaction, which captures more informative spatial patterns, has the most significant impact. In the second round, once the main spatial interaction has been fixed, the most important interaction emerges between the bonus-malus indicator and the age of the vehicle. Interestingly, this result is consistent with what we previously observed for the French MTPL dataset, where the same combination appeared among the most influential interaction effects.

6. Conclusion

This paper introduces the tree-like PIN, a novel neural network architecture designed to explicitly capture pairwise covariate interactions in tabular data. Modeling such interactions is essential in predictive tasks, as they reveal how covariates jointly affect the response beyond their individual contributions. The tree-like PIN embeds each input covariate into a latent space, and it models pairwise interactions through a shared FNN equipped with a centered hard sigmoid activation, which replicates the partitioning behavior of decision trees. The use of a shared network ensures parameter efficiency, while the inclusion of interaction-specific learnable parameters preserves the flexibility required to capture heterogeneous dependencies across covariate pairs.

Numerical analysis is performed on two commonly used real-world insurance datasets. The results show that the tree-like PIN achieves a strong predictive performance, and it consistently outperforms both traditional benchmarks and recently introduced neural network architectures, including the credibility transformer of Richman *et al.* (2025). In addition, the tree-like PIN enhances model interpretability by enabling the analysis of how pairs of covariates jointly contribute to the response. Since this architecture only involves pairwise interactions, it allows for a very efficient way of computing SHAP by the paired-sampling permutation SHAP method. The paper also highlights connections between the functioning of the tree-like PIN architecture and existing methods. Despite its novel architecture, the tree-like PIN shares similarities with GA^{2}Ms, gradient boosting, and GNNs.

Possible future research directions include the incorporation of regularization techniques to promote sparsity, thereby enabling an automated selection of the most relevant interaction pairs

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

and improving model generalization. Such regularization strategies could also be employed to penalize terms involving protected attributes, supporting the development of fairer predictive models. Another promising avenue is the extension of the tree-like PIN architecture to time-series data, enabling the joint modeling of temporal dependencies and feature interactions.

Acknowledgements. The authors thank the two anonymous referees whose comments helped to significantly improve the manuscript.

Code and data availability statement. Code and data are available from the GitHub repository: https://github.com/wueth/Tree-Like-PIN.

Competing interests. The authors are not aware of any conflict of interest.

References

Apley, **D. W.**, & **Zhu**, **J.** (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society, Series B: Statistical Methodology*, **82**(4), 1059–1086.

Brauer, **A.** (2024). Enhancing actuarial non-life pricing models via transformers. *European Actuarial Journal*, **14**(3), 991–1012.

Breiman, **L.** (2001). Random forests. *Machine Learning*, **45**(1), 5–32.

Chang, **L.**, **Gao**, **G.**, & **Shi**, **Y.** (2024). Claims reserving with a robust generalized additive model. *North American Actuarial Journal*, **28**(4), 840–860.

Devlin, **J.**, **Chang**, **M.-W.**, **Lee**, **K.**, & **Toutanova**, **K.** (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Dutang, **C.**, **Charpentier**, **A.**, & **Gallic**, **E.** (2024). Insurance dataset. Recherche Data Gouv. https://github.com/dutangc/CASdatasets.

Enouen, **J.**, & **Liu**, **Y.** (2022). Sparse interaction additive networks via feature interaction detection and sparse selection. In *Advances in neural information processing systems (NeurIPS) 35* (pp. 23410–23424).

Fisher, **A.**, **Rudin**, **C.**, & **Dominici**, **F.** (2018). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. arXiv:1801.01489.

Friedman, **J. H.**, & **Popescu**, **B. E.** (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics*, **2**(3), 916–954.

Gneiting, **T.**, & **Raftery**, **A. E.** (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, **102**(477), 359–378.

Gorishniy, **Y.**, **Rubachev**, **I.**, **Khrulkov**, **V.**, & **Babenko**, **A.** (2021). Revisiting deep learning models for tabular data. In *Advances in neural information processing systems 34* (pp. 18932–18943). Curran Associates, Inc.

Hastie, **T.**, & **Tibshirani**, **R.** (1986). Generalized additive models (with discussion). *Statistical Science*, **1**, 297–318.

Havrylenko, **Y.**, & **Heger**, **J.** (2024). Detection of interacting variables for generalized linear models via neural networks. *European Actuarial Journal*, **14**(2), 551–580.

Li, **Z.**, **Cui**, **Z.**, **Wu**, **S.**, **Zhang**, **X.**, & **Wang**, **L.** (2019). Fi-GNN: Modeling feature interactions via graph neural networks for CTR prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)* (pp. 539–548). Association for Computing Machinery.

Lou, **Y.**, **Caruana**, **R.**, **Gehrke**, **J.**, & **Hooker**, **G.**2013). Accurate intelligible models with pairwise interactions . In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 623–631). Association for Computing Machinery.

Lundberg, **S. M.** (2018). shap.PermutationExplainer. https://shap.readthedocs.io/en/latest/generated/shap.PermutationExplainer.html.

Lundberg, **S. M.**, & **Lee**, **S.-I.** (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems 30* (pp. 4765–4774).

Maillart, **T.**, & **Robert**, **C.** (2024). Distilling additive tree models into generalized linear models: A three-step GAM approach. *Annals of Actuarial Science*, **18**(3), 692–711.

Mayer, **M.**, & **Stando**, **A.** (2025). shapviz: SHAP visualizations. R package.

Mayer, **M.**, & **Wüthrich**, **M. V.** (2025). Shapley values: Paired-sampling algorithms. arXiv:2508.12947.

McCullagh, **P.**, & **Nelder**, **J. A.** (1983). *Generalized linear models*. Chapman & Hall.

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use, available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402

Nelder, **J. A.**, & **Wedderburn**, **R. W. M.** (1972). Generalized linear models. *Journal of the Royal Statistical Society, Series A*, **135**(3), 370–384.

Richman, **R.** (2021a). AI in actuarial science–A review of recent advances–Part 1. *Annals of Actuarial Science*, **15**(2), 207–229.

Richman, **R.** (2021b). AI in actuarial science–A review of recent advances–Part 2. *Annals of Actuarial Science*, **15**(2), 230–258.

Richman, **R.**, **Scognamiglio**, **S.**, & **Wüthrich**, **M. V.** (2025). The credibility transformer. *European Actuarial Journal*, **15**, 345–379.

Shapley, **L. S.** (1953). A value for n-person games. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the theory of games, AM-28, Volume II* (pp. 307–318). Princeton University Press.

Song, **W.**, **Shi**, **C.**, **Xiao**, **Z.**, **Duan**, **Z.**, **Xu**, **Y.**, **Zhang**, **M.**, & **Tang**, **J.** (2019). AutoInt: Automatic feature interaction learning via self-attentive neural networks . In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)* (pp. 1161–1170).

Štrumbelj, **E.**, & **Kononenko**, **I.** (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, **11**, 1–18.

Štrumbelj, **E.**, & **Kononenko**, **I.** (2014). Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, **41**(3), 647–665.

Tsang, **M.**, **Cheng**, **D.**, & **Liu**, **Y.** (2018). Detecting statistical interactions from neural network weights . In *Proceedings of the 6th International Conference on Learning Representations (ICLR)* (pp. 1–21).

Vaswani, **A.**, **Shazeer**, **N.**, **Parmar**, **N.**, **Uszkoreit**, **J.**, **Jones**, **L.**, **Gomez**, **A. N.**, **Kaiser**, **Ł.**, & **Polosukhin**, **I.** (2017). Attention is all you need. arXiv:1706.03762v5.

Wood, **S. N.** (2017). *Generalized additive models: An introduction with R* (2nd ed.). CRC Press.

Wüthrich, **M. V.**, & **Merz**, **M.** (2019). Editorial: Yes, we CANN! *ASTIN Bulletin – The Journal of the IAA*, **49**(1), 1–3.

Wüthrich, **M. V.**, & **Merz**, **M.** (2023). *Statistical foundations of actuarial learning and its applications*. Springer Actuarial. https://link.springer.com/book/10.1007/978-3-031-12409-9.

Wüthrich, **M. V.**, **Richman**, **R.**, **Avanzi**, **B.**, **Lindholm**, **M.**, **Maggi**, **M.**, **Mayer**, **M.**, **Schelldorfer**, **J.**, & **Scognamiglio**, **S.** (2025). AI Tools for Actuaries. SSRN Manuscript ID 5162304.

Cite this article: Richman R, Scognamiglio S and Wüthrich M (2026). Tree-like pairwise interaction networks, *Annals of Actuarial Science*, 1–24. https://doi.org/10.1017/S1748499526100402

Downloaded from https://www.cambridge.org/core. UNIVERSIDAD AUTONOMA METROPOLITANA, on 10 Nov 2026 at 09:47:37, subject to the Cambridge Core terms of use , available at https://www.cambridge.org/core/terms. https://doi.org/10.1017/S1748499526100402