Beyond Black-Box Predictions: Identifying Marginal Feature Effects in Tabular Transformer Networks

Anton Frederik Thielmann* Amazon Music
Arik Reuter* University of Cambridge
Benjamin Säfken Technical University Clausthal
Max-Planck Institute for Intelligent Systems

Abstract

In recent years, deep neural networks have showcased their predictive power across a variety of tasks. Beyond natural language processing, the transformer architecture has proven efficient in addressing tabular data problems and challenges the previously dominant gradient-based decision trees in these areas. However, this predictive power comes at the cost of intelligibility: Marginal feature effects are almost completely lost in the black-box nature of deep tabular transformer networks. Alternative architectures that use the additivity constraints of classical statistical regression models can maintain intelligible marginal feature effects, but often fall short in predictive power compared to their more complex counterparts. To bridge the gap between intelligibility and performance, we propose an adaptation of tabular transformer networks designed to identify marginal feature effects. We provide theoretical justifications that marginal feature effects can be accurately identified, and our ablation study demonstrates that the proposed model efficiently detects these effects, even amidst complex feature interactions. To demonstrate the model's predictive capabilities, we compare it to several interpretable as well as black-box models and find that it can match black-box performances while maintaining intelligibility. The source code is available at `https://github.com/OpenTabular/NAMpy`.

*Work done while at Technical University Clausthal.

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Interpretability has emerged as one of the core concepts of tabular data analysis. Especially in high-risk domains such as healthcare, where understanding the data's underlying effects is of crucial importance, this leads researchers to commonly rely on identifiable and interpretable generalized additive models (GAMs) (Hastie, 2017), instead of powerful neural networks or decision trees (Erfanian et al., 2021; Ravindra et al., 2019; Prata et al., 2020). In applications where predictive power is a central objective, researchers often resort to model-agnostic methods that try to explain model predictions via local approximation and feature importance like Locally Interpretable Model Explanations (LIME) (Ribeiro et al., 2016), or Shapley values (Shapley, 1953) and their extensions (Sundararajan and Najmi, 2020). While these methods are very effective for, e.g., image classification tasks, our ablation study demonstrates that they often fail to produce meaningful or accurate representations of the true feature contributions, as evidenced by near-zero or negative R^2 values. This empirical finding underscores a critical insight: architectural, in-built interpretability is a superior approach for capturing true marginal effects when compared to model-agnostic post-hoc explanations.

To bridge the gap in performance seen with traditional statistical models while preserving interpretability, recent efforts have focused on enhancing visual interpretability by incorporating additivity constraints into neural network architectures (Agarwal et al., 2021; Chang et al., 2021; Enouen and Liu, 2022). Similar to GAMs each feature is fit with a separate shape function. Neural additive models (NAMs) (Agarwal et al., 2021) and their extensions have emerged as a powerful yet interpretable solution for tabular data problems. While these models offer visual interpretability, they come with inherent downsides: **I)** There is a performance gap relative to fully connected black-box models and even to simple MLPs. **II)** The networks can become parameter-dense, depending on the complexity of the marginal

effects, as each feature is modeled with a distinct shape function. **III)** The complexity of the model structure grows rapidly with the number of features, especially when accounting for feature interactions, leading not only to a potentially suboptimal inductive bias, but also a vast hyperparamter space making effective hyperparamter tuning computationally expensive or even impossible. **IV)** Higher-order feature interactions can negatively impact the model's identifiability and are thus often simply excluded (Kim et al., 2022; Siems et al., 2024).

We propose to leverage the existing proven, and highly performant architectures for deep tabular learning and introduce a new architecture to bridge the gap between high-performing tabular models and inherently interpretable models using a flexible tabular transformer architecture. More specifically, we use target-aware embeddings (Gorishniy et al., 2022) and fit shallow one layer neural networks on uncontextualized embeddings while accounting for all higher order interaction effects.

Our contributions can be summarized as follows:

I. We introduce the **NAMformer**, a fully connected tabular deep learning architecture that combines a tabular transformer with interpretable feature networks.

II. We demonstrate that this straightforward approach yields intelligible and identifiable marginal feature effects, while perfectly maintaining the predictive power of its black-box counterparts and adding an almost negligible amount of additional parameters to the model.

III. We show that identifiability can be achieved by employing strategic feature dropout.

2 LITERATURE REVIEW

This manuscript can be categorized into two main areas of literature: **I**): tabular deep learning and **II**): additive interpretable modeling, the latter being largely inspired by classical statistical models.

I) While tree-based bagging and boosting methods have traditionally dominated predictive modeling for tabular data (Breiman, 2001; Chen and Guestrin, 2016; Prokhorenkova et al., 2018), recent advancements in deep learning have demonstrated that neural approaches can be highly competitive and, in some cases, even superior on specific datasets (McElfresh et al., 2024). This resurgence of deep learning for tabular data has begun to challenge the long-standing dominance of boosted decision trees. Transformer-based architectures were among the first to narrow the performance gap (Gorishniy et al., 2021; Somepalli et al., 2021; Arik and Pfister, 2021; Huang et al., 2020), followed by retrieval-based models, which have achieved state-of-the-art results in several benchmarks (Gorishniy et al., 2023; Ye et al., 2024). Additionally, well-tuned yet simpler architectures (Gorishniy et al., 2024; Holzmüller et al., 2024) have proven surprisingly competitive. Despite these advancements, the most performant models - Bayesian prior-fitted transformers (Hollmann et al., 2025) - remain limited to smaller datasets and do not generalize well to truly interpretable modeling (Mueller et al., 2024). This raises fundamental questions about the trade-offs between accuracy and interpretability in deep learning for tabular data, an area that remains an active field of research.

II) An additive model, as the name suggests, learns marginal feature effects and derives its final prediction by summing over these effects (e.g. (Hastie, 2017)). These models originate from simple linear regression, but instead of relying on linear effects, generalized additive models (GAMs) allow for the learning of more complex relationships through shape functions. A key aspect of additive models is that they allow interpretation of marginal feature effects, which quantify the isolated impact of each individual feature on the prediction while holding all other features constant. Marginal effects provide a clear, interpretable mapping of feature-to-outcome relationships, making them especially valuable in domains such as healthcare, finance, and policy-making, where understanding why a model makes a specific prediction is critical (e.g. (Hastie and Tibshirani, 1985; Barrio et al., 2013)). By offering transparency into the model's reasoning, interpretable marginal effects allow practitioners to validate predictions, build trust in the model, and gain actionable insights for decision-making. Classical GAMs (e.g., ((Hastie, 2017; Wood, 2017)) often use splines for basis expansions. For an introduction, see (Fahrmeir et al., 2022). This approach offers significant advantages, particularly in terms of interpretability and intelligibility. However, detecting complex feature effects, especially those involving sharp jumps or interactions, can be challenging for spline-based models. This limitation has motivated the development of Neural Additive Models (NAMs) (Agarwal et al., 2021), which replace splines with shape functions modeled via multi-layer perceptrons (MLPs) optimized through gradient descent. Extensions of NAMs, such as those proposed in (Thielmann et al., 2024a; Kim et al., 2022; Chang et al., 2021; Dammann et al., 2025), build on this simple additive modeling concept. Depending on the model, shape functions vary from Multi-layer Perceptrons (MLPs) (Agarwal et al., 2021; Radenovic et al., 2022; Enouen and Liu, 2022) to neural oblivious decision trees (Chang et al., 2021;

De et al., 2024), splines (Luber et al., 2023; Rügamer et al., 2023; Seifert et al., 2022; Dubey et al., 2022), ensemble decision trees (Nori et al., 2019) or transformer networks (Thielmann et al., 2024b). Although these models outperform classical GAMs, they often lag behind the performance of models like XGBoost or FT-Transformers not imposing the additivity constraint.

3 METHODOLOGY

Let $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ be the training dataset of size n and let y denote the target variable that can be arbitrarily distributed. Each input $\boldsymbol{x} = (x_1, x_2, \ldots, x_J)$ contains J features. Let further $\boldsymbol{x} \equiv (\boldsymbol{x}_{cat}, \boldsymbol{x}_{num})$ denote the partition of the features into categorical and numerical (continuous) features that constitute the whole feature vector $\boldsymbol{x}$. Further, let $x_{j(cat)}^{(i)}$ denote the j-th categorical feature of the i-th observation, and hence $x_{j(num)}^{(i)}$ denote the j-th numerical feature of the i-th observation.

Marginal Feature Effects and Interpretability Given a function $f : \mathbb{R}^d \to \mathbb{R}$ modeling a response variable y based on an input vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_J)$, the *marginal effect* of a single feature x_j is defined as the expected response when averaging over all other covariates:

$$\mathbb{E}[f(\mathbf{x})|x_j] = \int f(\mathbf{x}) p(\mathbf{x}_{-j}|x_j) d\mathbf{x}_{-j}, \tag{1}$$

where $\mathbf{x}_{-j} = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d)$ represents all features except x_j, and $p(\mathbf{x}_{-j}|x_j)$ is the conditional density of $\mathbf{x}_{-j}$ given x_j. This expectation isolates the dependency of $f(\mathbf{x})$ on x_j while marginalizing out the influence of other variables. Given a link function $g(\cdot)$ that connects the linear predictor to the expected value of the response variable, accommodating different types of data distributions, additive models in their fundamental form can be expressed as:

$$g(\mathbb{E}\left[y|\, x_1, x_2, \ldots x_J\right]) = \beta_0 + \sum_{j=1}^{J} f_j(x_j), \tag{2}$$

where β_0 denotes the global intercept or bias term and $f_j : \mathbb{R} \to \mathbb{R}$ denote the univariate shape functions corresponding to input feature x_j and capturing the feature main effects. The marginal effect simplifies to:

$$\mathbb{E}[f(\mathbf{x})|x_j] = f_j(x_j) + \sum_{k \neq j} \mathbb{E}[f_k(x_k)], \tag{3}$$

This form explicitly separates individual feature contributions, making interpretability tractable. In classical GAMs, each feature is often expanded via basis functions to allow fitting smooth curves for each feature (Wood, 2017).

The core of the NAMformer architecture is given by a tabular transformer in combination with shallow MLPs that all take uncontextualized embeddings as their input. In a nutshell, we first encode all numerical features, tokenize all categorical features and feed all encoded features through data type dependent embedding layers. Subsequently, the embeddings are passed through a stack of transformer layers, after which the [CLS] token embedding is processed by a task specific model head. The *uncontextualized* embeddings, before being passed through the transformer layers, are simultaneously fed to shallow, one-layer independent neural networks. The final prediction is gained by summation over all shallow feature networks as well as the task specific head. The model is trained end-to-end. An overview over the models structure is given in figure 2. The model architecture, the reasoning for leveraging the *uncontextualized* embeddings for intelligibility, as well as the identifiability constraints are explained in detail below. In summary, we present a model that achieves identical performance to FT-Transformers (Gorishniy et al., 2021) while maintaining intelligibility with marginally more trainable parameters.

Feature Encoding and Embedding To leverage meaningful shallow networks, we first encode all elements of the numerical features in x_{num} into a target-aware embedding space. More specifically, we use either target-aware one-hot encodings or periodic linear encodings (PLE) (Gorishniy et al., 2022). Thus, for all numerical $x_{j(num)} \in \mathbb{R}$, we encode $x_{j(num)}$ such that it belongs either in $\mathbb{N}^{T_j}$ (one-hot) or in $\mathbb{R}^{T_j}$ (PLE), with a feature specific, target dependent encoding function $h_j(\boldsymbol{x}_{j(num)}, y)$. Decision trees are used for all $h_j(\cdot)$.

Note that the dimension of the encoding T_j depends on the feature, and not all features are necessarily mapped to the same dimension.

We further follow classical tabular transformer architectures where $\mathbf{E}_j(\cdot)$ represents the embedding function for feature j. Depending on the feature type, $\mathbf{E}_j$ embeds into the embedding space as follows: $\mathbf{E}_j : \mathbb{R}^{T_j} \to \mathbb{R}^e$ for numerical PLE encoded features, $\mathbf{E}_j : \mathbb{N}^{T_j} \to \mathbb{R}^e$ for numerical one-hot encoded features, and $\mathbf{E}_j : \mathbb{N} \to \mathbb{R}^e$ for categorical features. Categorical features are encoded using standard embedding layers, and for numerical features are passed through single dense layers as also done by Gorishniy et al. (2021).

It is worth noting that the embedding dimensionality can be chosen arbitrarily and can be smaller than the dimensionality of the encoded features.

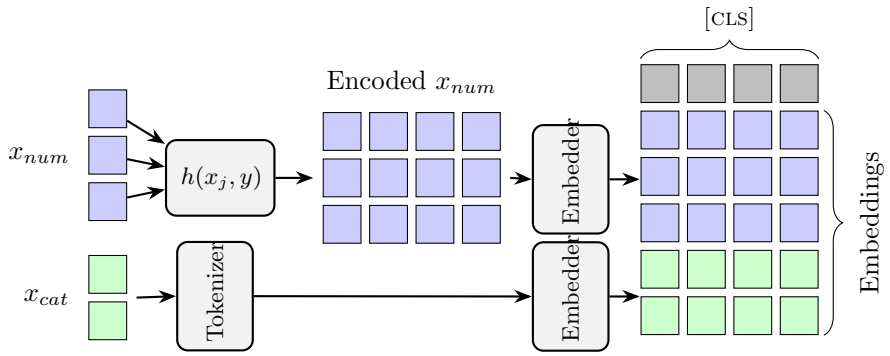
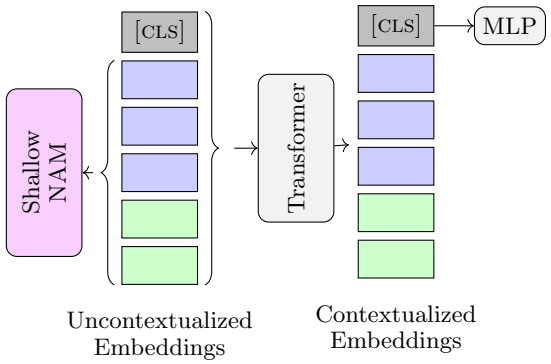
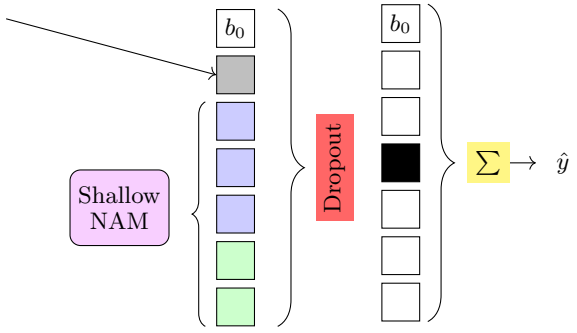


Figure 1: Feature Encoding. The numerical features are independently encoded ($h(x_j, y)$) and afterwards passed through an embedding layer. The categorical features are tokenized and also passed through an embedding layer.



(a) Generation of the embeddings. A [CLS] token is appended to the uncotenxtualized embeddings before being passed through transformer blocks. The uncontextualized embeddings are also inputs to the single-layer shallow feature networks.



(b) The contextualized [CLS] token is passed through a task specific MLP head. The output of the shallow feature networks as well as the output of the MLP is passed through a dropout layer and summed to create the final model prediction.

Figure 2: Training procedure of proposed model structure. The architecture is conceptually very similar to FT-Transformer but allows to identify for marginal feature effects. Note, that feature dropout is only applied during training.

From Uncontextual Embeddings to Marginal Predictions To understand the potential of *uncontextualized* embeddings as direct representations of raw input features in non-textual data settings, our investigation first examines their role within tabular data models.

State-of-the-art language models, leveraging the Transformer architecture (Vaswani et al., 2017) first create context-insensitive input token representations. In Natural Language Processing, these are the raw, *uncontextualized* word embeddings. Subsequently, they compute L layers of context-dependent representations, finally resulting in contextualized embeddings of the raw word representations (Peters et al., 2018). The proposed NAMformer architecture employs these *uncontextualized* embeddings directly to generate identifiable marginal feature predictions in a tabular context, necessitating a thorough analysis of their capability and effectiveness. By leveraging the *uncontextualized* embeddings, we aim to evaluate their utility in the proposed model architecture. This exploration is critical as it may reveal that raw, minimally processed embeddings can sufficiently capture and represent the essential characteristics of the features, potentially simplifying the model architecture while maintaining high predictive accuracy and interpretability.

In the context of tabular transformer networks, which do not employ positional encodings, the nature of *context* differs significantly from that in transformer models trained on textual data. The *context* that is added in the transformer layers, consists of the feature interactions (Huang et al., 2020). The *uncontextualized*, raw embeddings, however, are seldom used and are merely a byproduct of the model architecture. Since the input data for numerical features is not tokenized, *token identifiability* (Brunner et al., 2019) directly applies to the tabular input data. To confirm that the *uncontextualized* embeddings are not compromised by the subsequent layers during training, a straightforward

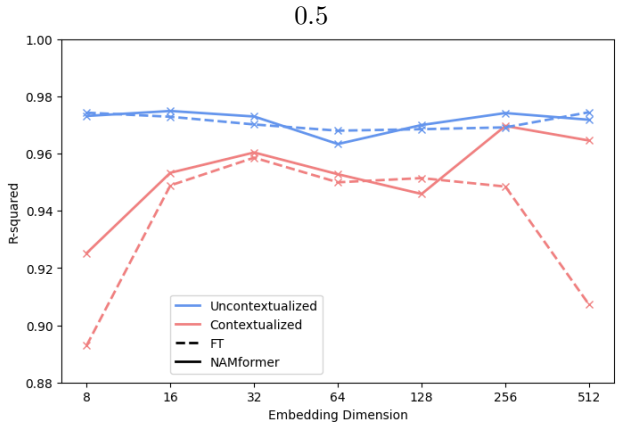


Figure 3: Average R^2 values over all 9 features. The decision trees are fit using either the *uncontextualized* or the contextualized embeddings as training data. The true features are the target variables. We find that the contextualized embeddings are almost perfect representations of the true data.

experiment is conducted: First, we train a tabular transformer model using the California housing dataset. Second, we extract the *uncontextualized* embeddings and analyze how well the true feature data can be recognized.

We follow Brunner et al. (2019) and train a set of J simple decision trees $dt^j : \mathbb{R}^e \to \mathbb{R}$ on the embedding of every feature x_j. The true inputs, x_j serve as the target variables, whereas the *uncontextualized* embeddings $\mathbf{E}_j(\boldsymbol{x}_j)$ are the training features. To put it differently, for each feature, we investigate how well it can be predicted with its contextualized or uncontextualized embedding. We report the values for the coefficient of determination (R^2), and find that *token identifiability* directly transfers to tabular input data and that *uncontextualized* embeddings are nearly perfect representations of the true data. More specifically, the R^2 values are ≥ 0.96 for different embedding sizes. For further details on the experimental setup, please refer to the Appendix.

Additivity Constraint Since we have verified that the *uncontextualized* embeddings almost perfectly preserve the original single feature information, we use them to include marginal feature predictions in the model. To achieve this, we feed the *uncontextualized* embeddings into shallow simple neural networks. Subsequently, we make use of the simple additivity constraint from equation 2.

Let then $f_j^\epsilon : \mathbb{R}^e \to \mathbb{R}$ represent the shape function for the j-th feature's uncontextualized embedding. Furthermore, let $H(\cdot)$ represent a sequence of transformer layers that take as input a sequence of all the uncontextualized embeddings $(\mathbf{E}_j(x_j))_{j=1}^J$ and output a sequence of contextualized embedding, such that $(\boldsymbol{\Xi}_j)_{j=1}^J = H((\mathbf{E}_j(x_j))_{j=1}^J)$. For simplicity, we denote the uncontextualized embeddings as ϵ_j and the contextualized embeddings as $\boldsymbol{\Xi}_j$, where $\boldsymbol{\Xi}_j = H(\epsilon_1, \epsilon_2, \dots \epsilon_J)_j$.

Appending a [CLS] token to the uncontextualized embeddings additionally allows for interpreting attention weights and emulating feature importance (Gorishniy et al., 2021). Let G further represent the MLP for processing the contextualized embeddings (or the [CLS] token embedding). The final model combines the individual transformed uncontextualized embeddings ϵ_j for each feature, along with the contextual embeddings $\boldsymbol{\Xi}_j$. This setup ensures that both individual feature effects (via shape functions) and global contextual interactions, via processing of the contextual embeddings, are accounted for and interpretable in the model's output:

$$g(\mathbb{E}\left[y|x_1, x_2, \dots x_J\right]) = \beta_0 + \sum_{j=1}^{J} f_j^\epsilon(\epsilon_j) + G(\boldsymbol{\Xi}_j). \quad (4)$$

Using target-aware encodings for numerical features allows us to use shallow, single-layer networks for the individual shape function $f_j^\epsilon : \mathbb{R}^e \to \mathbb{R}$. Thereby we can account for interpretable marginal feature effects by only increasing the total number of parameters by $J \times e$.

3.1 Disentanglement via Strategic Regularization

In any model that combines marginal and interaction effects, the issue of identifiability—the ability to uniquely attribute a prediction to its source—is a central challenge. The overparameterized nature of deep neural networks means that different sets of parameters can yield the same function, creating a functional ambiguity. Without specific constraints, the marginal effects can be "absorbed" by the interaction terms, making their individual contributions indistinguishable.

For simplicity, we change notation and assume an additive model, such as the NAMformer, that has the following additive predictor:

$$\hat{\eta} = \beta_0 + \sum_{j=1}^{J} f_j(x_j) + f_{J+1}(x_1, x_2, \dots, x_J), \quad (5)$$

where the marginal effects are modelled in separate networks, $f_j : \mathbb{R} \to \mathbb{R}$ and all interaction effects are jointly modelled in another network, $f_{J+1} : \mathbb{R}^J \to \mathbb{R}$. This simplifies our proposed model architecture but is transferable one to one.

See appendix for details on the used datasets

Further, assume the model is fitted with shape function dropout and a risk of (at most) R. The loss function $\mathcal{L}$ is induced by the choice of the link function g and distributional assumption in 4. Shape function dropout, introduced by (Agarwal et al., 2021), randomly drops out one or several features and their predictions in an additive model and is the main mechanism to ensure identifiability for NAMs. Here, let $\mathbf{w} \in \{0,1\}^{J+1}$ denote the shape function dropout vector leading to the following risk:

$$\mathbb{E}_{\mathbf{x},y\sim P^{\mathcal{D}}}\left[\mathbb{E}_{\mathbf{w}\sim P^{\mathbf{w}}}\left[\mathcal{L}\Bigg(\beta_0 + \sum_{j=1}^{J} w_j f_j(x_j) + w_{J+1} f_{J+1}(x_1, x_2, \ldots, x_J)\Bigg)\right]\right] =: R \tag{6}$$

Now, let $\tilde{\mathbf{w}}_k = (\delta_{jk})_{j=1}^{J+1}$ be the dropout weight vector that drops out everything except for the effect of f_k, i.e. $\tilde{\mathbf{w}}_k$ has a one exactly at the k-th positions and zeros everywhere else. Thus

$$R = \mathbb{E}_{\mathbf{x},y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right] p(\tilde{\mathbf{w}}_k) + R_{\tilde{\mathbf{w}}_{-k}}(1 - p(\tilde{\mathbf{w}}_k)),$$

where $R_{\tilde{\mathbf{w}}_{-k}} = R - \mathbb{E}_{\mathbf{x},y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right]$ is the risk not associated with $\tilde{\mathbf{w}}_{-k}$. Hence,

$$\mathbb{E}_{x_k,y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right] = \frac{R - R_{\tilde{\mathbf{w}}_{-k}}(1 - p(\tilde{\mathbf{w}}_k))}{p(\tilde{\mathbf{w}}_k)}.$$

Assuming a general distance-based loss $\mathcal{L}(y, \hat{y}) = g_{\mathcal{L}}(y - \hat{y})$ for a convex function $g_{\mathcal{L}}$, one obtains with Jensen's inequality:

$$\begin{aligned}
&\mathbb{E}_{x_k,y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right] \\
&\quad = \mathbb{E}_{x_k,y\sim P^{\mathcal{D}}}\left[g_{\mathcal{L}}\left(\beta_0 + f_k(x_k) - y\right)\right] \\
&\quad = \mathbb{E}_{x_k}\left[\mathbb{E}_{y|x_k}\left[g_{\mathcal{L}}\left(\beta_0 + f_k(x_k) - y\right)|x_k\right]\right] \\
&\quad \geq \mathbb{E}_{x_k}\left[g_{\mathcal{L}}\left(\mathbb{E}_{y|x_k}\left[\beta_0 + f_k(x_k) - y|x_k\right]\right)\right] \\
&\quad = \mathbb{E}_{x_k}\left[g_{\mathcal{L}}\left(\beta_0 + f_k(x_k) - \mathbb{E}_{y|x_k}\left[y|x_k\right]\right)\right] \\
&\quad = \mathbb{E}_{x_k}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), \mathbb{E}_{y|x_k}\left[y|x_k\right]\right)\right].
\end{aligned} \tag{7}$$

Note that most common regression loss-functions such as the L^p losses, the Huber loss or the Pinball loss are all distance based loss function of the form assumed above. Furthermore, an analogous argument can be made in the binary classification case with a margin-based binary (classification) loss of the form $\mathcal{L}(y, \hat{s}) = h_{\mathcal{L}}(y \cdot \hat{s})$ with labels $y \in \{-1, 1\}$ and $\hat{s} \in \mathbb{R}$ the output of a scoring classifier (see Appendix A).

In summary, we have shown for broad classes of regression and classification losses $\mathcal{L}$ that we can identify the true marginal effect $\mathbb{E}_{y|x_k}\left[y|x_k\right]$ with the following error, measured in terms of the original loss function $\mathcal{L}$:

$$\begin{aligned}
&\mathbb{E}_{x_k}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), \mathbb{E}_{y|x_k}[y|x_k]\right)\right] \\
&\qquad \leq \frac{R - R_{\tilde{\mathbf{w}}_{-k}}(1 - p(\tilde{\mathbf{w}}_k))}{p(\tilde{\mathbf{w}}_k)}
\end{aligned} \tag{8}$$

When the risk R is uniformly distributed among all values of $\tilde{\mathbf{w}}_k$, such that $R_{\tilde{\mathbf{w}}_{-k}} = R \cdot (1 - p(\tilde{\mathbf{w}}_k))$, one gets the following bound on the expected error with respect to the ground-truth effect:

$$\begin{aligned}
&\mathbb{E}_{x_k}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), \mathbb{E}_{y|x_k}[y|x_k]\right)\right] \\
&\qquad \leq \frac{R \cdot (1 - (1 - p(\tilde{\mathbf{w}}_k))^2)}{p(\tilde{\mathbf{w}}_k)} \\
&\qquad = R \cdot (2 - p(\tilde{\mathbf{w}}_k)) \leq 2R
\end{aligned} \tag{9}$$

4 SIMULATION STUDY: ANALYZING MARGINAL EFFECT IDENTIFIABILITY

To first evaluate the NAMformer's capacity for in-built interpretability, we begin with a controlled simulation study. This provides a direct, ground-truth comparison of how accurately different models can identify marginal feature effects, even in the presence of complex feature interactions.We compare it with other, neural (Agarwal et al., 2021) and decision tree based intelligible models (Nori et al., 2019), as well as a simple linear regression model and a GAM (Hastie, 2017). We simulate multiple datasets with a normally distributed target variable. Each dataset follows a straightforward data generating process where $y = \sum_{j=1}^{J} s_j(x_j) + \prod x_j + \varepsilon$, such that s_j are the marginal feature effects. Note, that via the inclusion of $\prod x_j$ the data generating process includes very strong higher order feature effects. See appendix B for further information on the data generating process. Subsequently, we fit each model on the dataset and analyze the marginal feature predictions. For Explainable Boosting Machines (EBM) (Nori et al., 2019) we use the default hyperparameters. For GAMs we use cubic splines with 25 knots each. For NAMs, we orientate on the architectures from Radenovic et al. (2022) and Dubey et al. (2022). For NAMformer, we use embedding sizes of 32, 4 layers, 2 heads, attention dropout of 0.3 and feed forward dropout of 0.3. For NAMs and NAMformer we use identical feature dropout probability of 0.1, since the shown identifiability is also applicable NAMs. Additionally, we compare, target aware one-hot encodings with 150 bins, PLE encodings with 25 bins and standardization of numerical features for NAMformer.

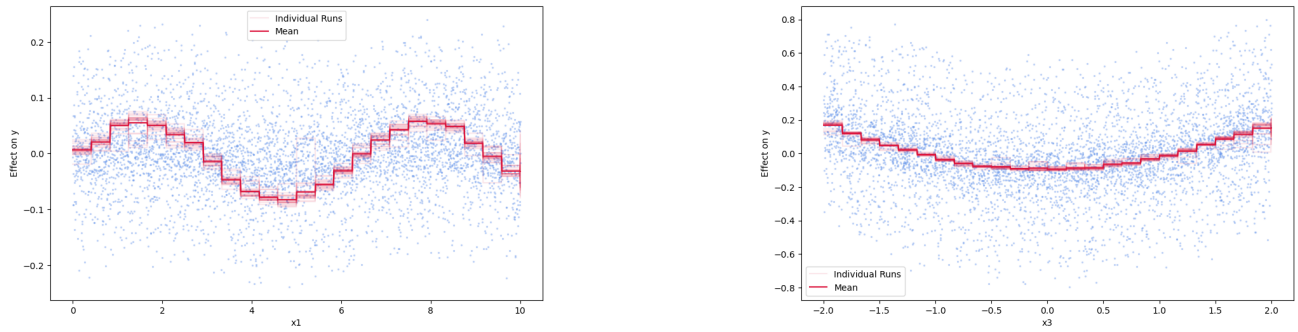


Figure 4: Marginal feature predictions for a simple simulated example with 4 variables and the described data generating process. Over 25 runs, and with only 25 bins, NAMformer accurately identifies the marginal effects.

Table 1: Average R^2 values over marginal feature effects for different datasets. With increasing index, the number of effects as well as the complexity of the data increase. Larger values are better. The gray $\pm$ values denote the standard deviation among the calculated R^2 value for the different marginal effects. *oh* denotes one-hot encoded features, *st* standardized features and *ple* periodic linear encodings.

	Number of marginal effects						
Model	3	4	5	6	7	8	9
Linear	0.467 ±0.41	0.251 ±0.67	0.220 ±0.62	0.238 ±0.59	0.124 ±0.63	0.034 ±0.68	0.092 ±0.68
GAM	0.800 ±0.37	0.534 ±0.76	0.466 ±0.73	0.500 ±0.69	0.356 ±0.76	0.257 ±0.81	0.299 ±0.79
EBM	0.797 ±0.37	0.531 ±0.76	0.464 ±0.73	0.500 ±0.69	0.371 ±0.73	0.266 ±0.79	0.331 ±0.74
EB^2M	0.797 ±0.37	0.531 ±0.76	0.464 ±0.73	0.500 ±0.69	0.371 ±0.73	0.266 ±0.79	0.331 ±0.74
NAM	0.741 ±0.35	0.653 ±0.39	0.611 ±0.39	0.648 ±0.39	0.629 ±0.40	0.477 ±0.61	0.507 ±0.59
Hi-NAM	0.801 ±0.36	0.556 ±0.75	0.577 ±0.63	0.676 ±0.47	0.597 ±0.50	0.526 ±0.64	0.658 ±0.57
FTTransformer (SHAP)	**0.025** ±0.35	0.156 ±0.47	0.121 ±0.41	-0.003 ±0.55	0.074 ±0.55	0.109 ±0.56	0.120 ±0.51
XGBoost (SHAP)	0.531 ±0.61	0.577 ±0.57	0.496 ±0.57	0.324 ±0.73	0.353 ±0.70	0.358 ±0.66	0.335 ±0.63
FTTransformer (IG)	-0.117 ±0.25	-0.120 ±0.22	-0.200 ±0.31	-0.301 ±0.37	-0.332 ±0.41	-0.380 ±0.42	-0.440 ±0.41
$\text{NAMformer}_{\text{st}}$	**0.865** ±0.23	0.662 ±0.57	0.596 ±0.53	0.781 ±0.28	0.605 ±0.43	0.631 ±0.58	0.770 ±0.56
$\text{NAMformer}_{\text{oh}}$	0.826 ±0.11	0.877 ±0.13	0.737 ±0.14	0.837 ±0.09	**0.922** ±0.13	**0.722** ±0.25	**0.826** ±0.23
$\text{NAMformer}_{\text{ple}}$	0.806 ±0.40	**0.918** ±0.15	**0.879** ±0.17	**0.867** ±0.11	0.909 ±0.10	0.617 ±0.59	0.756 ±0.56

Subsequently, we analyze the marginal feature predictions and calculate the R^2 values for each marginal feature prediction with respect to the true data generating function. Table 1 shows the averaged results over all effects.

We find that NAMformer can accurately identify marginal effects, even in the presence of higher-order feature interactions. Interestingly, using target aware encodings is also beneficial for feature identifiability in the NAMformer. Additionally, while NAMs implementing the same identifiability regularizer can also identify the marginal effects, their performance diminishes as more interactions are introduced. Furthermore, NAMs exhibit significantly larger standard deviations among the R^2 values for individual effects compared to NAMformer, which identifies all effects with smaller deviation. Figure 4 shows the detected marginal effects of NAMformer for a simple sinus as well as squared effect. Note, that there strong interaction effects present - as described in the data generating process above - which do not negatively impact NAMformers intelligibility.

See appendix B for the experimental details.

Figure 6 in the appendix shows the detected marginal effects for Latitude and Longitude from the California housing dataset respectively.

The results in Table 1 also provide a powerful empirical critique of post-hoc explanation methods. Post-hoc methods like SHAP (Shapley, 1953) and integrated gradients (IG) (Sundararajan and Najmi, 2020) are demonstrably unable to accurately identify global marginal effects from complex, non-additive deep models, especially in regression. These methods rely on approximations and baselines, often failing to fully disentangle individual feature contributions from interactions, resulting in uniformly low, and in some cases negative, R^2 values. These findings underscore the superiority of our inherently interpretable approach.

5 PERFORMANCE BENCHMARK

The performance-interpretability trade-off has long been a central challenge in tabular machine learning, exacerbated by the lack of a standardized, large-scale benchmark for evaluating interpretable models. To ad-

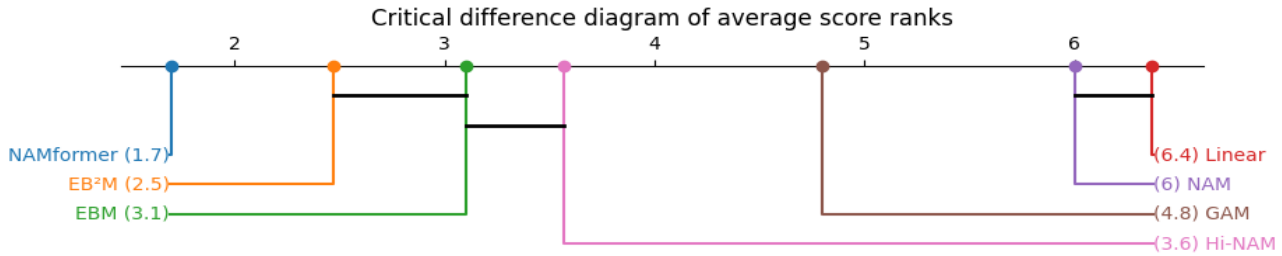


Figure 5: Critical difference diagram for all innherently interpretable models on the benchmark datasets reported in Table 3. The average ranks across tasks are shown in brackets next to each model. Horizontal lines indicate groups of models with no statistically significant differences in performance. *NAMformer* achieves the best average rank, with significant differences at the 5% level to the second-best performing model, EB^2M. The critical differences are computed using the Conover-Friedman test (Pereira et al., 2015), as both average ranks and performance metrics across all datasets are available.

dress this open problem, we evaluate the NAMformer on 15 diverse real-world datasets, an effort that represents the most extensive performance study in the field of interpretable deep learning. By building upon the established benchmark datasets from Fischer et al. (2023), we provide a robust and reproducible comparison resulting in a significantly broader interpretability study than those found in the existing literature (Agarwal et al., 2021; Thielmann et al., 2024b; Dubey et al., 2022; Radenovic et al., 2022). This benchmark aims to provide a clear and definitive assessment of NAMformers ability to exceed the performance of state-of-the-art interpretable models.

We compare NAMformer against a broad set of interpretable models, GAMs, EBMs, and NAMs, as well as a Hi-NAM (Kim et al., 2022) that incorporates a single MLP for feature interactions. For hyperparameter optimization we follow (Gorishniy et al., 2021). All experimental details on the implemented hyperparameter tuning, data splits, and variables can be found in the appendix.

To provide a robust, cross-dataset comparison, we utilize a rank-based evaluation methodology. For each dataset, we rank the models by their predictive performance (RMSE for regression, AUC for classification), with the best-performing model receiving a rank of 1. The average rank across all datasets serves as a reliable measure of overall model superiority. The results are summarized in the critical difference diagram in Figure 5. NAMformer achieves the best average rank and exhibits a statistically significant performance difference from the second-best model, confirming its state-of-the-art status. This provides strong evidence that the NAMformer effectively closes the performance-interpretability gap, offering a powerful, transparent, and robust solution for tabular data analysis.

6 CONCLUSION

We present NAMformer, an intelligible highly performant model. With minimally more parameters compared to the FT-Transformer architecture we achieve identical performance while also generating identifiable marginal feature predictions. The reasoning for including an additivity constraint and fitting shallow feature networks in tabular transformers is thus that without loss of generalizability and without loss of performance, we can get an interpretable model at the cost of -depending on the embedding size and the number of features- marginally more parameters. Our theoretical justification of identifiable marginal feature effects is also seamlessly applicable to models incorporating unstructured data (Rügamer et al., 2023; Reuter et al., 2024).

7 LIMITATIONS

The interpretability of NAMformer, while significantly better than that of black-box models, still does not match the inherent statistical inference capabilities of classical GAMs. True interpretability in the form of significance statistics is still a problem for further research. Additionally, this paper solely focuses on single marginal feature effects. While we account for high-order feature interactions, we do not explicitly account for, e.g., second order feature interactions as models like EB^2M do. Hence, interesting interaction effects as the ones between, e.g., longitude and latitude are not specifically accounted for. However, the introduced identifiability constraint seamlessly enables to account for any amount and order of feature interactions that one wants to account for. Simple feature interaction networks can be easily incorporated and fit on a combination of the *uncontextualized* embeddings.

ACKNOWLEDGEMENTS

Funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within project 450330162 is gratefully acknowledged.

References

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., and Hinton, G. E. (2021). Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34:4699–4711.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Arik, S. Ö. and Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6679–6687.

Barrio, I., Arostegui, I., Quintana, J. M., and Group, I.-C. (2013). Use of generalised additive models to categorise continuous variables in clinical prediction. *BMC medical research methodology*, 13:1–13.

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

Brunner, G., Liu, Y., Pascual, D., Richter, O., Ciaramita, M., and Wattenhofer, R. (2019). On identifiability in transformers. *arXiv preprint arXiv:1908.04211*.

Chang, C.-H., Caruana, R., and Goldenberg, A. (2021). Node-gam: Neural generalized additive model for interpretable deep learning. *arXiv preprint arXiv:2106.01613*.

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

Dammann, L. M., Freitag, M., Thielmann, A., and Säfken, B. (2025). Gradient-based smoothing parameter estimation for neural p-splines. *Computational Statistics*, pages 1–19.

De, A., Thielmann, A. F., and Säfken, B. (2024). Nodegamlss: Interpretable uncertainty modelling via deep distributional regression. In *NeurIPS 2024 Workshop on Bayesian Decision-making and Uncertainty*.

Dubey, A., Radenovic, F., and Mahajan, D. (2022). Scalable interpretability via polynomials. *arXiv preprint arXiv:2205.14108*.

Enouen, J. and Liu, Y. (2022). Sparse interaction additive networks via feature interaction detection and sparse selection. *arXiv preprint arXiv:2209.09326*.

Erfanian, M., Sadeghpour Gildeh, B., and Reza Azarpazhooh, M. (2021). A new approach for monitoring healthcare performance using generalized additive profiles. *Journal of Statistical Computation and Simulation*, 91(1):167–179.

Fahrmeir, L., Kneib, T., Lang, S., and Marx, B. D. (2022). Regression models. In *Regression: Models, methods and applications*, pages 23–84. Springer.

Fischer, S. F., Feurer, L. H. M., and Bischl, B. (2023). OpenML-CTR23 – a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*.

Gorishniy, Y., Kotelnikov, A., and Babenko, A. (2024). Tabm: Advancing tabular deep learning with parameter-efficient ensembling. *arXiv preprint arXiv:2410.24210*.

Gorishniy, Y., Rubachev, I., and Babenko, A. (2022). On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004.

Gorishniy, Y., Rubachev, I., Kartashev, N., Shlenskii, D., Kotelnikov, A., and Babenko, A. (2023). Tabr: Unlocking the power of retrieval-augmented tabular deep learning. *arXiv preprint arXiv:2307.14338*.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

Hastie, T. and Tibshirani, R. (1985). Generalized additive models; some applications. In *Generalized Linear Models: Proceedings of the GLIM 85 Conference held at Lancaster, UK, Sept. 16–19, 1985*, pages 66–81. Springer.

Hastie, T. J. (2017). Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. (2025). Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326.

Holzmüller, D., Grinsztajn, L., and Steinwart, I. (2024). Better by default: Strong pre-tuned mlps and boosted trees on tabular data. *Advances in Neural Information Processing Systems*, 37:26577–26658.

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. (2020). Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

Kim, M., Choi, H.-S., and Kim, J. (2022). Higher-order neural additive models: An interpretable machine learning model with feature interactions. *arXiv preprint arXiv:2209.15409*.

Luber, M., Thielmann, A., and Säfken, B. (2023). Structural neural additive models: Enhanced interpretable machine learning. *arXiv preprint arXiv:2302.09275*.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. (2024). When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36.

Mueller, A., Siems, J., Nori, H., Salinas, D., Zela, A., Caruana, R., and Hutter, F. (2024). Gamformer: In-context learning for generalized additive models. *arXiv preprint arXiv:2410.04560*.

Nori, H., Jenkins, S., Koch, P., and Caruana, R. (2019). Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pereira, D. G., Afonso, A., and Medeiros, F. M. (2015). Overview of friedman's test and post-hoc analysis. *Communications in Statistics-Simulation and Computation*, 44(10):2636–2653.

Peters, M. E., Neumann, M., Zettlemoyer, L., and Yih, W.-t. (2018). Dissecting contextual word embeddings: Architecture and representation. *arXiv preprint arXiv:1808.08949*.

Prata, D. N., Rodrigues, W., and Bermejo, P. H. (2020). Temperature significantly changes covid-19 transmission in (sub) tropical cities of brazil. *Science of the Total Environment*, 729:138862.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., and Gulin, A. (2018). Catboost: unbiased boosting with categorical features. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Radenovic, F., Dubey, A., and Mahajan, D. (2022). Neural basis models for interpretability. *arXiv preprint arXiv:2205.14120*.

Ravindra, K., Rattan, P., Mor, S., and Aggarwal, A. N. (2019). Generalized additive models: Building evidence of air pollution, climate change and human health. *Environment international*, 132:104987.

Reuter, A., Thielmann, A., and Saefken, B. (2024). Neural additive image model: Interpretation through interpolation. *arXiv preprint arXiv:2405.02295*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Rügamer, D., Kolb, C., and Klein, N. (2023). Semi-structured distributional regression. *The American Statistician*, pages 1–12.

Seifert, Q. E., Thielmann, A., Bergherr, E., Säfken, B., Zierk, J., Rauh, M., and Hepp, T. (2022). Penalized regression splines in mixture density networks.

Shapley, L. (1953). Quota solutions op n-person games1. *Edited by Emil Artin and Marston Morse*, page 343.

Siems, J., Ditschuneit, K., Ripken, W., Lindborg, A., Schambach, M., Otterbach, J., and Genzel, M. (2024). Curve your enthusiasm: Concurvity regularization in differentiable generalized additive models. *Advances in Neural Information Processing Systems*, 36.

Somepalli, G., Goldblum, M., Schwarzschild, A., Bruss, C. B., and Goldstein, T. (2021). Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*.

Sundararajan, M. and Najmi, A. (2020). The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR.

Thielmann, A. F., Kruse, R.-M., Kneib, T., and Säfken, B. (2024a). Neural additive models for location scale and shape: A framework for interpretable neural regression beyond the mean. In *International Conference on Artificial Intelligence and Statistics*, pages 1783–1791. PMLR.

Thielmann, A. F., Reuter, A., Kneib, T., Rügamer, D., and Säfken, B. (2024b). Interpretable additive tabular transformer networks. *Transactions on Machine Learning Research*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wood, S. N. (2017). *Generalized additive models: an introduction with R*. chapman and hall/CRC.

Ye, H.-J., Yin, H.-H., and Zhan, D.-C. (2024). Modern neighborhood components analysis: A deep tabular baseline two decades later. *arXiv preprint arXiv:2407.03257*.

CHECKLIST

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A SHAPE FUNCTION DROPOUT FOR THE MSE-LOSS AND CLASSIFICATION LOSSES

In this section, we show how the result in section 3.1 can be made more precise in the case of an MSE loss and how it can be adapted to the case of margin-based classification losses. In general, we want to show

$$\mathbb{E}_{x_k}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), \mathbb{E}_{y|x_k}[y|x_k]\right)\right] \leq \mathbb{E}_{x_k, y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right]. \tag{10}$$

First, one obtains when assuming an MSE-Loss $\mathcal{L}(y, \hat{y}) = (y - \hat{y})^2$:

$$\begin{aligned}\mathbb{E}_{x_k, y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right] &= \mathbb{E}_{x_k, y\sim P^{\mathcal{D}}}\left[\left(\beta_0 + f_k(x_k) - y\right)^2\right] = \\ &\mathbb{E}_{x_k}\left[\mathbb{E}_{y|x_k}\left[\left(\beta_0 + f_k(x_k) - y\right)^2 |x_k\right]\right] \\ &- \left(\mathbb{E}_{y|x_k}\left[\beta_0 + f_k(x_k) - y|x_k\right]\right)^2 + \left(\mathbb{E}_{y|x_k}\left[\beta_0 + f_k(x_k) - y|x_k\right]\right)^2 = \\ &\mathbb{E}_{x_k}\left[\mathbb{V}\left[y|x_k\right]\right] + \mathbb{E}_{x_k}\left[\left(\beta_0 + f_k(x_k) - \mathbb{E}_{y|x_k}[y|x_k]\right)^2\right]\end{aligned} \tag{11}$$

Here, $\mathbb{E}_{x_k}\left[\mathbb{V}\left[y|x_k\right]\right] =: R_{x_k}$ is the irreducible error in predicting y based on x_k. Thus one can obtain the following explicit expression for the risk:

$$\mathbb{E}_{x_k}\left[\left(\beta_0 + f_k(x_k) - \mathbb{E}_{y|x_k}[y|x_k]\right)^2\right] = \frac{R - R_{\tilde{\mathbf{w}}_{-k}}(1 - p(\tilde{\mathbf{w}}_k))}{p(\tilde{\mathbf{w}}_k)} - R_{x_k}$$

For margin-based binary classification losses of the form $\mathcal{L}(y, \hat{s}) = h_{\mathcal{L}}(y \cdot \hat{s})$, such that $y \in \{-1, 1\}$ and $\hat{s} \in \mathbb{R}$ is the output of a scoring classifier and $h_{\mathcal{L}}$ is convex, one obtains:

$$\begin{aligned}\mathbb{E}_{x_k, y\sim P^{\mathcal{D}}}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), y\right)\right] &= \mathbb{E}_{x_k, y\sim P^{\mathcal{D}}}\left[h_{\mathcal{L}}\left(y \cdot (\beta_0 + f_k(x_k))\right)\right] = \\ &\mathbb{E}_{x_k}\left[\mathbb{E}_{y|x_k}\left[h_{\mathcal{L}}\left(y \cdot (\beta_0 + f_k(x_k))\right)|x_k\right]\right] \geq \mathbb{E}_{x_k}\left[h_{\mathcal{L}}\left(\mathbb{E}_{y|x_k}\left[y \cdot (\beta_0 + f_k(x_k))|x_k\right]\right)\right] \\ &= \mathbb{E}_{x_k}\left[h_{\mathcal{L}}\left((\beta_0 + f_k(x_k)) \cdot \mathbb{E}_{y|x_k}\left[y|x_k\right]\right)\right] = \mathbb{E}_{x_k}\left[\mathcal{L}\left(\beta_0 + f_k(x_k), \mathbb{E}_{y|x_k}\left[y|x_k\right]\right)\right].\end{aligned} \tag{12}$$

Note that here $\mathbb{E}_{y|x_k}\left[y|x_k\right] = 2\mathbb{P}(y = 1|x_k) - 1$. Thus for $\tilde{y} = \frac{y+1}{2} \in \{0, 1\}$, which is the 0-1 variant of the label y, and therefore $\tilde{\mathcal{L}}(\tilde{y}, \hat{s}) = h_{\mathcal{L}}((2\tilde{y} - 1) \cdot \hat{s})$, one gets

$$\mathbb{E}_{x_k, y\sim P^{\mathcal{D}}}\left[\tilde{\mathcal{L}}\left(\beta_0 + f_k(x_k), \tilde{y}\right)\right] \geq \mathbb{E}_{x_k}\left[\tilde{\mathcal{L}}\left(\beta_0 + f_k(x_k), \mathbb{E}_{y|x_k}\left[\tilde{y}|x_k\right]\right)\right], \tag{13}$$

where $\mathbb{E}_{y|x_k}\left[\tilde{y}|x_k\right] = \mathbb{P}(\tilde{y} = 1|x_k)$, showing that the difference of the marginal effect of x_k to the Bayes-Optimal classification model is bounded in this case.

Many common classification loss functions, such as the 0-1-loss, the Log-Loss, the Hinge Loss, the Exponential Loss can be expressed as a margin-based loss with a convex function $h_{\mathcal{L}}$.

B ABLATION STUDY

In the ablation study, we simulate a dataset consisting of 25,000 data points. We utilize a train-test split of 70% - 30% to evaluate the impact of various shape functions and categorical feature effects on the model's performance.

Continuous Features We examine the following set of shape functions to model the continuous features. All x variables are uniformly distributed between 0 and 1 and independently sampled. The functions are designed to introduce a variety of nonlinear transformations:

- Linear function: $s_1(x) = 3x$

- Quadratic function: $s_2(x) = (x-1)^2$
- Sinusoidal function: $s_3(x) = \sin(5x)$
- Exponential root function: $f_4(x) = \sqrt{\exp(x)}$
- Absolute deviation: $s_5(x) = |x-1|$
- Sinusoidal deviation: $s_6(x) = |x - \sin(5x)|$
- Signed root function: $s_7(x) = \text{sign}(x) \cdot \sqrt{|x|}$
- Exponential-polynomial function: $s_8(x) = 2^x - x^2$
- Cubic polynomial: $s_9(x) = x^3 - 3x$
- Exponential increment: $s_{10}(x) = \exp(x + 10^{-6})$

Categorical Features The dataset includes three categorical features, each with different levels and associated effects:

- **cat_feature_1**: Levels = {A, B, C}. Effects = {A: 0.5, B: -0.5, C: 0.0}
- **cat_feature_2**: Levels = {D, E}. Effects = {D: 1.0, E: -1.0}
- **cat_feature_3**: Levels = {F, G, H, I}. Effects = {F: 0.2, G: -0.2, H: 0.1, I: -0.1}

Each categorical feature is encoded to reflect its specific impact, which varies depending on the level present in the dataset. These effects are designed to simulate real-world scenarios where categorical features may influence the outcome in both positive and negative ways.

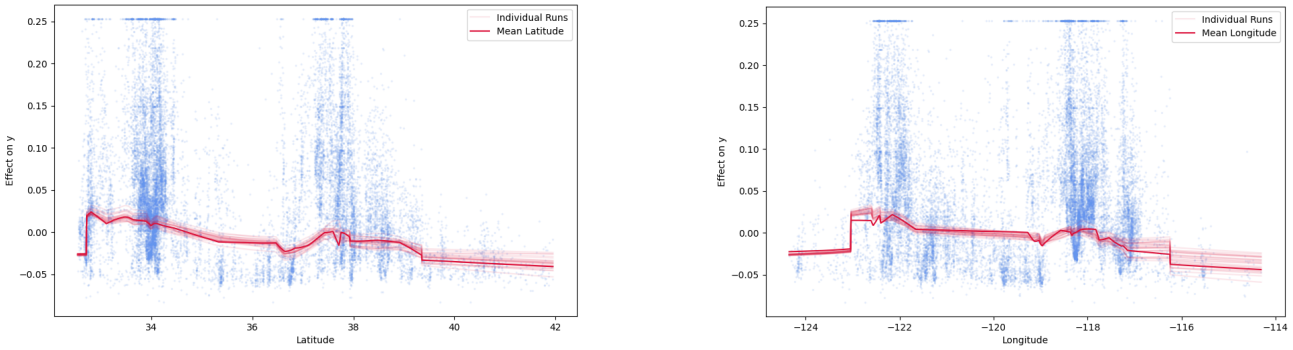
Note that the product structure of the considered interaction effects ensures that the true marginal effects $\mathbb{E}[y|x_k]$ are given by h_k. This is because, first, with independence of the covariates $x_1, x_2, \dots x_J$, and ϵ one has:

$$\mathbb{E}[y|x_k] \quad = \quad \mathbb{E}\left[\sum_{j=1}^{J} s_j(x_j) + \prod x_j + \epsilon \middle| x_k\right] \quad = \quad s_k(x_k) \; + \; \sum_{j=1, j\neq k}^{J} \mathbb{E}[s_j(x_j)] \; + \; x_k \prod_{j\neq k} \mathbb{E}[x_j] \; + \; \mathbb{E}[\epsilon] \quad (14)$$

Second, assuming zero-centered covariates and effects, as well as a zero-centered error term then yields $\mathbb{E}[y|x_k] = s_k(x_k)$.

B.1 Network architectures

We use NAM architecture inspired by Radenovic et al. (2022) and Dubey et al. (2022). Hence, we use simple network with each feature network consisting of [64, 64, 32] hidden neurons respectively each followed by a 0.1 dropout layer and ReLU activation. For Hi-NAMs we implement the same architecture for the feature interaction network. For EBM and EB^{2}M we use the default architecture (Nori et al., 2019). For GAMs we use cubic splines with 25 knots each. For NAMformer we use an embedding size of 32, 4 layers, 2 heads, 150 one hot encoded bins and dropout of 0.3 throughout all dropout layers, except for a feature dropout of 0.1. Learning rates of 1e-03, a patience of 20 epochs and learning rate decay with a patience of 10 epochs regarding the validation loss was used where applicable.



(a) Latitude (b) Longitude

Figure 6: Marginal feature predictions from 25 trained NAMformer models on the California housing datasets for the variables "latitude" and "longitude" using PLE encodings.

Comparison to FT-Transformer Since the proposed architecture closely follows the FT-Transformer architecture from Gorishniy et al. (2021), we first compare whether introducing identifiable marginal feature networks hampers predictive power compared to classical FT-Transformers. We fit both models with identical transformer architectures and use the same feature encodings or preprocessing methods for both models. We perform 5-fold cross-validation and compare mean squared error values on 4 regression datasets and Area under the curve (AUC) on 4 binary classification datasets. See appendix **??** for details on the experimental setup. Notably, we do not find that either model performs better or worse, as no model achieves significantly different performances with respect to the cross validation results. This strengthens our hypothesis, that adding marginally identifiable networks with minimally more parameters ($J \times e < 5000$ for all datasets) does not harm predictive performance at all.

Table 2: Comparison between NAMformer and FT-Transformer with identical hyperparameters on different datasets. 5-fold cross validation was performed. The average performances for both models are not out of the bounds of the standard deviations over the 5-folds. Hence, we find that NAMformer, while producing identifiable marginal effects performs as good as FT-Transformer.

Model	CH ↓	MU ↓	DM ↓	HS ↓	AD ↑	BA ↑	SH ↑	FI ↑
$\text{NAMformer}_{\text{st}}$	0.235	0.798	0.021	0.131	0.908	0.953	0.862	0.788
	±0.013	±0.351	±0.001	±0.021	±0.003	±0.006	±0.007	±0.021
FT-T_{st}	0.227	0.780	0.023	0.127	0.908	0.960	0.861	0.790
	±0.011	±0.323	±0.002	±0.018	±0.002	±0.010	±0.009	±0.010
$\text{NAMformer}_{\text{oh}}$	0.220	0.801	0.022	0.162	0.903	0.901	0.825	0.766
	±0.007	±0.379	±0.001	±0.021	±0.006	±0.022	±0.006	±0.013
FT-T_{oh}	0.225	0.901	0.024	0.158	0.899	0.644	0.820	0.763
	±0.007	±0.417	±0.002	±0.029	±0.007	±0.144	±0.010	±0.010
$\text{NAMformer}_{\text{ple}}$	0.206	0.642	0.020	0.127	0.912	0.945	0.858	0.789
	±0.007	±0.241	±0.001	±0.016	±0.002	±0.007	±0.005	±0.013
FT-T_{ple}	0.197	0.834	0.023	0.129	0.910	0.944	0.858	0.789
	±0.011	±0.420	±0.001	±0.022	±0.005	±0.020	±0.007	±0.009

Table 3: Results for interpretable models. For regression problems (CH, MU, DM, HS) mse values are reported, for binary classification problems (AD, BA, SH, FI) the area under the curve as well as the accuracy in gray are reported.

	Regression Results (MSE ↓)											Classification Reults (AUC)			
Model	CH ↓	MU ↓	DM ↓	HS ↓	AV ↓	GS ↓	K8 ↓	P32 ↓	MH ↓	BH ↓	SG ↓	AD ↑	BA ↑	SH ↑	FI ↑
Linear	0.370	0.726	0.115	0.333	0.700	0.366	0.580	0.843	0.295	0.025	0.445	0.852	0.871	0.764	0.754
GAM	0.288	0.747	0.066	0.267	0.287	0.228	0.557	0.909	0.157	0.023	0.273	0.913	0.911	0.855	0.779
EBM	0.195	0.703	0.023	0.205	0.050	0.079	0.411	0.395	0.096	0.033	0.272	**0.927**	**0.931**	0.868	**0.783**
EB^2M	0.194	0.695	0.023	0.201	0.049	0.079	0.409	**0.388**	0.099	0.026	**0.263**	**0.927**	**0.931**	0.870	**0.783**
NAM	0.306	0.735	0.069	0.451	0.372	0.235	0.927	1.049	0.181	0.025	0.399	0.910	0.901	0.853	0.776
Hi-NAM	0.194	0.718	**0.022**	**0.132**	0.135	**0.034**	**0.076**	0.435	0.102	0.128	0.278	0.910	0.911	0.858	0.779
NAMformer	**0.173**	**0.668**	**0.022**	0.148	**0.023**	0.051	0.108	0.397	**0.095**	**0.022**	0.270	**0.927**	**0.931**	**0.871**	0.780

C RESULTS

For black-box models, we compare NAMformer to classical MLPs, XGBoost and FT-Transformer. We use standardized preprocessed features for the FT-Transformer since we found them to perform best in our initial experiment (Table 2). Note, that we tune the hyperparamters of NAMformer and FT-Transformer separately and thus get different results than those from table 2. Overall, we can confirm the results from Gorishniy et al. (2021) that FT-Transformer can outperform XGBoost on certain datasets and that NAMformer achieves comparable predictive performance to its black-box counterpart while acounting for interpretable marginal feature effects.

Table 4: Results for black-box models. For regression problems (CH, MU, DM, HS) mse values are reported, for binary classification problems (AD, BA, SH, FI) the area under the curve as well as the accuracy (in gray) are reported.

Model	CH ↓	MU ↓	DM ↓	HS ↓	AD ↑	BA ↑	SH ↑	FI ↑
XGB	**0.159**	0.728	0.018	0.163	**0.927** 87.3%	**0.933** 90.5%	0.868 86.3%	**0.781** 70.1%
FT-T	0.184	0.688	**0.017**	**0.111**	0.915 85.9%	0.929 90.2%	0.870 86.3%	0.777 70.1%
MLP	0.195	0.725	0.018	0.164	0.908 89.8%	0.913 85.5%	0.862 86.5%	0.771 70.0%
NAMformer	0.173	**0.668**	0.022	0.148	**0.927** 87.2%	0.931 90.8%	**0.871** 86.5%	0.780 70.7%

D HYPERPARAMETER TUNING

We use Bayesian hyperparameter tuning using the Optuna library (Akiba et al., 2019). We use 50 trials for each method, and report the results for the best trial on either the validation mean squared error or the validation (binary) cross entropy. We use median pruning. For all neural models we implement early stopping with a patience of 15 epochs based on the validation loss and return the best model with respect to the validation loss of that time frame. Additionally we implement learning rate decay with a patience of 10 epochs also based on the validation loss with a factor of 0.1. All hyperparameter spaces for all models are reported below:

XGBoost For the XGBoost model, we tune the following hyperparameters:

- **n_estimators:** Number of trees, varied from 50 to 400.
- **max_depth:** Maximum depth of each tree, with values ranging from 3 to 20.
- **learning_rate:** Learning rate, adjusted between 0.001 and 0.2.
- **subsample:** Subsample ratio of the training instances, from 0.5 to 1.0.
- **colsample_bytree:** Subsample ratio of features for each tree, ranging from 0.5 to 1.0.

Explainable Boosting Machine (EBM and EB^2M) For the EBM model, the following hyperparameters are tuned to optimize performance:

- **Interactions:** Set to `0.0` to disable automatic interaction detection.

- **Learning Rate:** The rate at which the model learns, varied from 0.001 to 0.2.
- **Max Leaves:** The maximum number of leaves per tree, with choices ranging from 2 to 4.
- **Min Samples Leaf:** The minimum number of samples required to be at a leaf node, varying from 2 to 20.
- **Max Bins:** The maximum number of bins used for discretizing continuous features, sampled from 512 to 8192.

For EB^2M we tune the following:

- **Interactions:** Set to `0.95` to strongly favor interaction effects among features.
- **Learning Rate:** Adjusted identically to the EBM, within the range of 0.001 to 0.2.
- **Min Samples Leaf:** Also ranging from 2 to 20 to control overfitting.
- **Max Leaves:** From 2 to 4, to define the complexity of the learned models.
- **Max Bins:** The binning parameter, ranging from 512 to 8192, to optimize the handling of continuous variables.

Neural Additive Models (NAMs) For Neural Additive Models, we configure a range of hyperparameters to optimize model performance. The hyperparameter space explored includes dimensions of hidden layers, dropout rates, learning rates, and more, as specified below:

- **Hidden Dimensions:** A categorical choice among different layer configurations to adapt the model capacity, including configurations such as `[64, 64]`, `[64, 32, 16]`, `[128, 64]`, `[128, 128, 64]`, `[128, 32]`, and `[128, 64]`.
- **Dropout Rate:** Dropout rate for regularization, varied from 0.1 to 0.5.
- **Feature Dropout Probability:** Probability of dropping a feature to prevent overfitting, ranged from 0.1 to 0.5, sampled on a logarithmic scale.
- **Learning Rate (lr):** Learning rate for training, explored on a logarithmic scale between 10^{-5} and 10^{-3}.
- **Weight Decay:** Regularization parameter to minimize overfitting, also explored on a logarithmic scale, with values ranging from 10^{-6} to 10^{-4}.
- **Activation Function:** The type of activation function used in the model, selected from options such as ReLU, Leaky ReLU, GELU, SELU, and Tanh.
- **Batch size:** Fixed batch size from $\{32, 64, 128, 256, 512\}$.

Multi-Layer Perceptron (MLP) For the MLP model, we fine-tune the following hyperparameters:

- **Hidden Layer Sizes:** Sizes of each hidden layer, where each layer's size is individually tuned between 8 and 512 neurons, defined dynamically for each layer during the trials.
- **Learning Rate (lr):** The optimizer's learning rate, sampled logarithmically between 10^{-5} and 10^{-3}.
- **Weight Decay:** Regularization parameter to minimize overfitting, explored on a logarithmic scale from 10^{-6} to 10^{-4}.
- **Batch Normalization:** A binary choice to either use or not use batch normalization in each layer.
- **Skip Connections:** Option to include skip connections between layers.
- **Activation Function:** Determines the type of activation function used, options include ReLU, Leaky ReLU, GELU, SELU, and Tanh.
- **Dropout Rate:** Dropout rate for each layer to prevent overfitting, adjustable between 0.0 and 0.5.
- **Batch size:** Fixed batch size from $\{32, 64, 128, 256, 512\}$.

FT-Transformer For the FT-Transformer model, we fine-tune the following hyperparameters:

- **Embedding Size:** Dimensionality of embeddings for categorical features, selected from {16, 32, 64, 128, 256}.
- **Number of Heads (n_head):** The number of attention heads in the transformer, chosen from {1, 2, 4, 8}.
- **Number of Layers (n_layers):** Configurable number of transformer layers, ranging from 1 to 8.
- **Learning Rate (lr):** Optimized on a logarithmic scale from 10^{-5} to 10^{-3}.
- **Weight Decay:** Regularization parameter explored on a logarithmic scale between 10^{-6} and 10^{-4}.
- **Activation Function:** Options include ReLU, Leaky ReLU, GELU, SELU, and Tanh.
- **Head Dropout:** Dropout rate in the heads, adjustable from 0.0 to 0.5.
- **Attention Dropout:** Dropout rate specifically for the attention mechanisms, also adjustable from 0.0 to 0.5.
- **Head Layer Sizes:** A variety of configurations for layer sizes in the model's head are tested, including:
  - Single layer configurations: {1}, {32}, {64}
  - Dual layer configurations: {64, 64}, {128, 64}, {128, 32}
  - Triple layer configurations: {64, 32, 16}, {128, 128, 64}, {128, 64, 32}
- **Batch Size:** The size of the batches for training, selected from {32, 64, 128, 256, 512}.

NAMformer For the NAMformer model, we fine-tune the following hyperparameters:

- **Embedding Size:** Dimensionality of embeddings for categorical features, selected from {16, 32, 64, 128, 256}.
- **Number of Heads (n_head):** The number of attention heads in the transformer, chosen from {1, 2, 4, 8}.
- **Number of Layers (n_layers):** Configurable number of transformer layers, ranging from 1 to 8.
- **Learning Rate (lr):** Optimized on a logarithmic scale from 10^{-5} to 10^{-3}.
- **Weight Decay:** Regularization parameter explored on a logarithmic scale between 10^{-6} and 10^{-4}.
- **Activation Function:** Options include ReLU, Leaky ReLU, GELU, SELU, and Tanh.
- **Head Dropout:** Dropout rate in the heads, adjustable from 0.0 to 0.5.
- **Attention Dropout:** Dropout rate specifically for the attention mechanisms, also adjustable from 0.0 to 0.5.
- **Feature Dropout:** Dropout rate for features, ranging from 0.1 to 0.5.
  - Single layer configurations: {1}, {32}, {64}
  - Dual layer configurations: {64, 64}, {128, 64}, {128, 32}
  - Triple layer configurations: {64, 32, 16}, {128, 128, 64}, {128, 64, 32}
- **Batch Size:** The size of the batches for training, selected from {32, 64, 128, 256, 512}.

E EMBEDDING IDENTIFIABILITY

To demonstrate that the uncontextualized embeddings almost perfectly represent the raw input data, we conducted the experiment described in the main body of our work. We fitted both a NAMformer and a FT-Transformer (Gorishniy et al., 2021) to the California housing dataset, with preprocessing as outlined previously. Our comparison includes both one-hot encoded numerical features and standardized features.

The models were trained using identical architectures with an embedding size of 64, 4 layers, 4 heads, and a uniform dropout rate of 0.3; for the NAMformer, feature dropout was set at 0.1. We employed a learning rate of 1e-04, weight decay of 1e-05, and a patience setting of 15 epochs for early stopping. The reported results represent the average outcomes from a 5-fold cross-validation.

After the models converged, we fitted simple decision trees—using the default settings from (Pedregosa et al., 2011)—to the uncontextualized embeddings from the training split, treating the true features as labels. We then computed the R^2 on the test data. Subsequently, we performed the same procedure for the contextualized embeddings. The findings are illustrated in Figure 7.

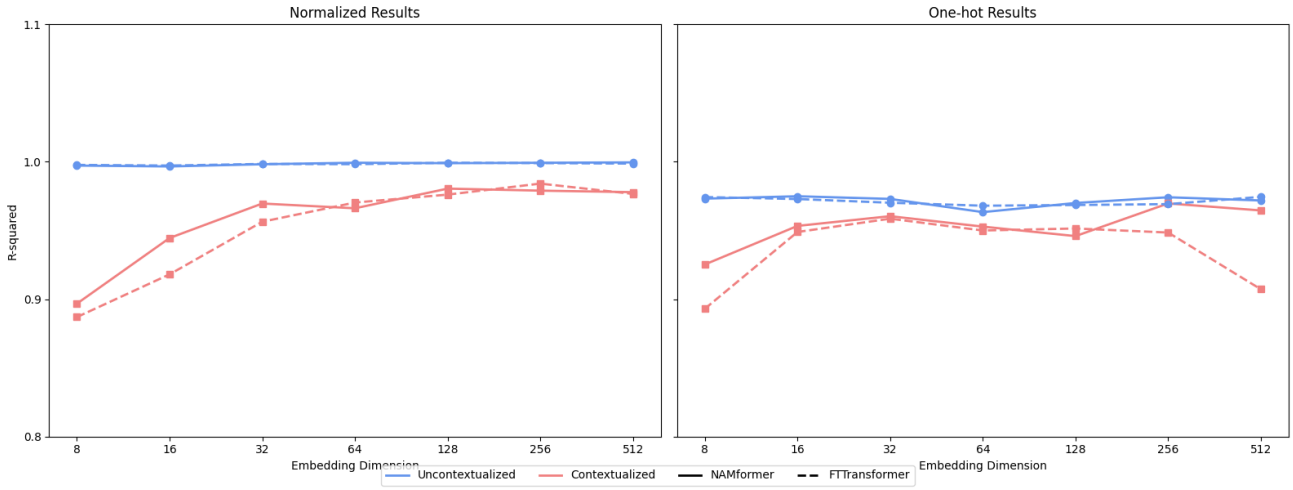


Figure 7: Average R^2 values over all 9 features in the california housing dataset. The decision trees are fit, using either the uncontextualized or the contextualized embeddings as training data and the true features as target variables.

F ENCODING

We orientate on Gorishniy et al. (2022) and denote the encoded feature as $\boldsymbol{z}_{j(\text{num})}$ with entries

One-hot encoding

$$z^t_{j(\text{num})} = \begin{cases} 0 & \text{if } x < b_t, \\ 1 & \text{if } x \geq b_t. \end{cases}$$

PLE

$$z^t_{j(\text{num})} = \begin{cases} 0 & \text{if } x < b_{t-1}, \\ 1 & \text{if } x \geq b_t, \\ \frac{x - b_{t-1}}{b_{t-2} - b_{t-1}} & \text{else.} \end{cases}$$

where b_t denote the decision boundaries from the decision trees.