SEMF: Supervised Expectation-Maximization Framework for Predicting Intervals

Ilia Azizi ILIA.AZIZI@UNIL.CH
HEC, University of Lausanne, Lausanne, Switzerland
Marc-Olivier Boldi MARC-OLIVIER.BOLDI@UNIL.CH
HEC, University of Lausanne, Lausanne, Switzerland
Valérie Chavez-Demoulin VALERIE.CHAVEZ@UNIL.CH
HEC, University of Lausanne, Expertise Center for Climate Extremes (ECCE), Lausanne, Switzerland

Editor: Khuong An Nguyen, Zhiyuan Luo, Harris Papadopoulos, Tuwe Löfström, Lars Carlsson and Henrik Boström

Abstract

This work introduces the Supervised Expectation-Maximization Framework (SEMF), a versatile and model-agnostic approach for generating prediction intervals with any ML model. SEMF extends the Expectation-Maximization algorithm, traditionally used in unsupervised learning, to a supervised context, leveraging latent variable modeling for uncertainty estimation. Through extensive empirical evaluation of diverse simulated distributions and 11 real-world tabular datasets, SEMF consistently produces narrower prediction intervals while maintaining the desired coverage probability, outperforming traditional quantile regression methods. Furthermore, without using the quantile (pinball) loss, SEMF allows point predictors, including gradient-boosted trees and neural networks, to be calibrated with conformal quantile regression. The results indicate that SEMF enhances uncertainty quantification under diverse data distributions and is particularly effective for models that otherwise struggle with inherent uncertainty representation.[1]

Keywords: Uncertainty estimation, Expectation-Maximization (EM), Latent Representation Learning, conformal prediction

1. Introduction

In the evolving field of machine learning (ML), the quest for models able to predict outcomes while quantifying the uncertainty of their predictions is critical. The ability to estimate prediction uncertainty is particularly vital in high-stakes domains such as healthcare (Dusenberry et al., 2020), finance (Wisniewski and Polanski, 2020), and autonomous systems (Tang et al., 2022), where prediction-based decisions have important consequences. Traditional approaches have primarily focused on point estimates, with little to no insight into prediction reliability. This limitation underscores the need for frameworks that can generate both precise point predictions and robust prediction intervals. Such intervals provide a range within which the true outcome is expected to lie with a fixed probability, offering a finer understanding of prediction uncertainty. This need has spurred research into methodologies that extend beyond point estimation to include uncertainty quantification, thereby enabling more informed decision-making in applications reliant on predictive modeling (Ghahramani, 2015).

1. Code: https://github.com/Unco3892/semf-paper

In this paper, we introduce the Supervised Expectation-Maximization Framework (SEMF) based on the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). Traditionally recognized as a clustering technique, EM is used for supervised learning in SEMF, allowing for both point estimates and prediction intervals using any ML model (model-agnostic), with a particular focus on uncertainty quantification. While EM has been predominantly used in unsupervised settings, its application to supervised learning represents a principled extension that leverages the algorithm's ability to handle latent variables for uncertainty quantification, a crucial need in modern ML applications. This paper details the methodology behind the framework and proposes a training algorithm based on Monte Carlo (MC) sampling, also used in variational inference for Variational Auto-Encoders (VAEs) (David M. Blei and McAuliffe, 2017a; Kingma and Welling, 2014).

Our method, SEMF, differs from prominent supervised EM approaches such as Ghahramani and Jordan (1993), which focus on point prediction using Gaussian Mixture Models (GMMs). Unlike VAE-based approaches that optimize the evidence lower bound (ELBO), SEMF directly maximizes the likelihood through importance-weighted sampling, avoiding the need for variational approximations and their potential limitations, such as poor posterior approximation quality when the true posterior $p(z|x, y)$ is multimodal or highly non-Gaussian, and the inherent gap between the ELBO and true likelihood that can compromise uncertainty estimates. Furthermore, SEMF operates in a frequentist paradigm, directly maximizing the likelihood function through iterative EM steps without integrating over posterior distributions. Although SEMF can be extended to a Bayesian framework as its likelihood component, this extension lies beyond the scope of this paper. Finally, SEMF generates representations for latent modalities through specialized models, holding potential for multi-modal data applications.

The remainder of this paper is organized as follows: Section 2 details the theory and methodology of SEMF. Section 3 reviews related work in latent representation learning and uncertainty estimation. Section 4 describes the experimental setup, including synthetic and benchmark datasets, as well as evaluation metrics. Section 5 discusses the results, demonstrating the efficacy of SEMF. Lastly, Section 6 concludes the paper, and Section 7 outlines the limitations and potential research directions.

2. Method

This section presents the founding principles of SEMF from its parameters, training, and inference procedure with, at its core, the EM algorithm. Invented to maximize the model likelihood, it builds a sequence of parameters that guarantee an increase in the log-likelihood (Wu, 1983) by iterating between the Expectation (E) and the Maximization (M) steps. In the E-step, one computes

$$Q(p|p') = \mathbb{E}_{Z\sim p'(z|x)}\left[\log p(x, Z)\right] = \int \log p(x, z)p'(z|x)dz, \tag{1}$$

where p' stands for the current estimates, $\log p(x, z)$ is the log-likelihood of the complete observation (x, z), and z is a latent variable. The M-step updates the current estimates with the arguments maximizing the Q-function: $p' \leftarrow \arg\max_p Q(p|p')$. The sequence is repeated until convergence.

2.1. Problem Scenario

Let $x = (x_1, x_2, \ldots, x_K)$ denote K inputs and the output be y. For simplicity, we limit y to be continuous numerical, although this assumption could be relaxed to discrete or categorical without loss of generality. Component x_k is a source: a modality, a single or group of variables, or an unstructured input such as an image or text. For clarity, we limit to $K = 2$, where x_1 and x_2 are single variables.

Let $p(y|x)$ be the density function of the outcome given the inputs. A founding assumption, in the spirit of VAE, is that $p(y|x)$ decomposes into $p(y|x) = \int p(y|z)p(z_1|x_1)p(z_2|x_2)dz_1dz_2$, where $z = (z_1, z_2)$ are unobserved latent variables. We assume that $p(y|z, x) = p(y|z)$, that is, z contains all the information of x about y, and that $p(z|x) = p(z_1|x_1)p(z_2|x_2)$, that is, there is one latent variable per source. These are independent, conditionally on their corresponding source. While jointly embedding all features with a single model might seem more intuitive, our separate embedding approach offers several advantages: (1) it naturally handles multi-modal data where different feature groups may require distinct processing (e.g., images vs. tabular data), (2) it allows for modular model selection where each g_{ϕ_k} can be tailored to the characteristics of x_k, and (3) it can potentially lead to to handling missing data by allowing inference on available modalities. This design choice prioritizes flexibility, though the multi-modal setting and handling of missing data are left for future exploration.

The contribution to the log-likelihood of a complete observation (y, z, x) is $\log p(y, z|x) = \log p(y|z) + \log p(z|x)$. In the E-step, we compute

$$\int \log p(y, z|x)p'(z|y, x)dz \quad = \quad \int \log p(y|z)p'(z|y, x)dz + \int \log p(z|x)p'(z|y, x)dz. \quad (2)$$

Eq. 2 can be estimated by MC sampling. Since sampling from $p'(z|y, x)$ can be inefficient, we rather rely on the decomposition $p'(z|y, x) = p'(y|z)p'(z|x)/p'(y|x)$. Thus, we sample z_r from $p'(z|x)$, $r = 1, \ldots, R$, and, setting $w_r = p'(y|z_r)/\sum_t p'(y|z_t)$, approximate the right-hand side term of Eq. 2

$$\int \log p(y, z|x)p'(z|y, x)dz \quad \approx \quad \sum_{r=1}^{R} \{\log p(y|z_r) + \log p(z_r|x)\} w_r. \quad (3)$$

2.2. Objective Function

Adapting Eq. 3 for the observed data $\{(y_i, x_i)\}_{i=1}^N$, the overall loss function, $\mathcal{L}$, is

$$\mathcal{L}(\phi, \theta) = -\sum_{i=1}^{N}\sum_{r=1}^{R} \{\log p_\phi(z_{i,r}|x_i) + \log p_\theta(y_i|z_{i,r})\} w_{i,r}, \quad (4)$$

where the models of $p(y|z)$ and $p(z|x)$ inherit parameters θ and ϕ, respectively. The weights are

$$w_{i,r} = \frac{p_{\theta'}(y_i|z_{i,r})}{\sum_{t=1}^{R} p_{\theta'}(y_i|z_{i,t})} \quad (5)$$

where $z_i, r \overset{ind.}{\sim} p_{\phi'}(z|x_i)$, $r = 1, \ldots, R$.

Eq. 4 shows that $\mathcal{L}$ is a sum of losses associated with the encoder model, p_ϕ, for each source, and the decoder model, p_θ, from the latent variables to the output. p_θ and p_ϕ are

referred to as encoder and decoder in the spirit of auto-encoder models. At each M-step, $\mathcal{L}$ is minimized with respect to θ and ϕ. Then, θ' and ϕ' are updated, as well as the weights and the sampled z. Then, the process is iterated until convergence.

2.2.1. Example: $\mathcal{L}(\phi, \theta)$ under normality

For illustration purposes, we develop below the case where p_ϕ and p_θ are normal distributions, similar to Kingma and Welling (2014). As a reminder, any other distributions could be adopted, including non-continuous or non-numerical outcomes.

Encoder $p_\phi(z|x)$. Let m_k be the length of the latent variable z_k, $k = 1, 2$. We assume a normal model for Z_k given $X_k = x_k$,

$$Z_k | X_k = x_k \sim \mathcal{N}_{m_k}(g_{\phi_k}(x_k), \sigma_k^2 J_{m_k}), \tag{6}$$

where J_{m_k} is the $m_k \times m_k$ identity matrix. In particular,

$$\log p_\phi(z_k|x_k) \quad = \quad -\frac{m_k}{2}\log 2\pi - \frac{m_k}{2}\log\sigma_k^2 - \frac{1}{2\sigma_k^2}\sum_{j=1}^{m_k}\{z_{k,j} - g_{\phi_k,j}(x_k)\}^2, \quad k = 1, 2. \tag{7}$$

The mean $g_{\phi_k}(x_k)$ can be any model, such as a neural network, with output of length m_k, $k = 1, 2$. The scale σ_k can be fixed, computed via the weighted residuals, or learned through a separate set of models. It controls the amount of noise introduced in the latent dimension and is pivotal in determining the width of the prediction interval for $p(y|z)$. The code implementation allows for all three methods. This paper only presents the results for a fixed σ_k, as well as training a separate set of $g_{\phi_k}(x_k)$ models to estimate varying σ_ks.

Decoder $p_\theta(y|z)$. We assume a normal model for Y given $Z = z$,

$$Y | Z = z \sim \mathcal{N}(f_\theta(z), \sigma^2). \tag{8}$$

This results in a log-likelihood contribution,

$$\log p_\theta(y|z) = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\{y - f_\theta(z)\}^2. \tag{9}$$

Again, the mean $f_\theta(z)$ can be any model, such as a neural network.

Summary. Overall, the M-step is

$$\phi_k^* = \arg\min_{\phi_k} \sum_{i,r} w_{i,r} \sum_{j=1}^{m_k}\{z_{k,i,r,j} - g_{\phi_k,j}(x_{k,i,r})\}^2, \quad k = 1, 2, \tag{10}$$

$$\theta^* = \arg\min_{\theta} \sum_{i,r} w_{i,r}\{y_i - f_\theta(z_{i,r})\}^2, \tag{11}$$

$$(\sigma^*)^2 = \frac{1}{N}\sum_{i,r} w_{i,r}\{y_i - f_{\theta^*}(z_{i,r})\}^2, \tag{12}$$

2.3. Training

For efficiency purpose, the training set, $\{1, \ldots, N\}$, is segmented into batches $\{b_1, \ldots, b_L\}$ on which the index i runs (and thus the denominator of Eq. 12 must be adapted accordingly). The process iterates for each batch until the maximum number of steps is reached or an early stopping criterion is satisfied. The full details are given in algorithm 1. The framework requires tuning hyper-parameters such as the number of MC samples R, the number of latent nodes m_k, and the standard deviation σ_k of Z_k. Monitoring the point prediction on a hold-out validation is important to combat overfitting and terminate the training early with a PATIENCE hyper-parameter. Moreover, due to the generative nature of SEMF, the variation resulting from the initial random seed is measured in Subsection 4.1. Additionally, the model-specific hyper-parameters (p_ϕ and p_θ) are also discussed in the same subsection.

Algorithm 1 SEMF Training: two input sources

Require: y, x_1, x_2, R {Training data and number of MC samples}
Ensure: θ, ϕ_1, ϕ_2 {Trained model parameters}
1: Initialize θ, ϕ_1, ϕ_2
2: Initialize D_y, D_{z_1}, D_{z_2} to $\emptyset$ {Data buffers for batch updates}
3: Split $I = \{1, \ldots, N\}$ into L batches $\{b_1, \ldots, b_L\}$
4: **for** $\ell = 1, \ldots, L$ **do**
5: **for all** i in b_ℓ **do**
6: **for** $r = 1, \ldots, R$ **do**
7: Simulate $z_{1,i,r} \sim p_{\phi_1}(\cdot|x_{1,i})$
8: Simulate $z_{2,i,r} \sim p_{\phi_2}(\cdot|x_{2,i})$
9: Set $z_{i,r} = [z_{1,i,r}, z_{2,i,r}]$
10: **end for**
11: **for** $r = 1, \ldots, R$ **do**
12: Compute

$$w_{i,r} = \frac{p_\theta(y_i|z_{i,r})}{\sum_{t=1}^{R} p_\theta(y_i|z_{i,t})}$$

13: Update $D_y \leftarrow D_y \cup [y_i|z_{i,r}|w_{i,r}]$
14: Update $D_{z_1} \leftarrow D_{z_1} \cup [z_{1,i,r}|x_{1,i}|w_{i,r}]$
15: Update $D_{z_2} \leftarrow D_{z_2} \cup [z_{2,i,r}|x_{2,i}|w_{i,r}]$
16: **end for**
17: **end for**
18: Update $\theta \leftarrow \arg\min_\theta \sum_{(y,z,w)\in D_y} w \cdot \mathcal{L}_\theta(y, z)$ {M-step for decoder}
19: Update $\phi_1 \leftarrow \arg\min_{\phi_1} \sum_{(z,x,w)\in D_{z_1}} w \cdot \mathcal{L}_{\phi_1}(z, x)$ {M-step for encoder 1}
20: Update $\phi_2 \leftarrow \arg\min_{\phi_2} \sum_{(z,x,w)\in D_{z_2}} w \cdot \mathcal{L}_{\phi_2}(z, x)$ {M-step for encoder 2}
21: Clear D_y, D_{z_1}, D_{z_2} {Reset buffers for next batch}
22: **end for**
23: Check convergence; Go to step 4 if not

2.4. Inference

The encoder-decoder structure of SEMF entails the simulations of z_r during inference, as depicted in Figure 1. In theory, any inference on y given x can be performed for $\hat{y}$, for instance the mean value $\hat{y} = \frac{1}{R}\sum_{r=1}^{R} f_\theta(z_r)$, where $z_r \sim p_\phi(z|x)$ (see algorithm 2 for the simulation scheme). For prediction intervals, a second simulation step is used,

$$z_r \sim p(z|x), \quad \hat{y}_{r,s} \sim p_\theta(y|z_r), \quad r, s = 1, \dots, R. \tag{13}$$

Prediction interval on y given x at a given level α follows as

$$PI = \text{quantile}\left(\{\hat{y}_{r,s}\}; \frac{\alpha}{2}, 1 - \frac{\alpha}{2}\right). \tag{14}$$

In order to calibrate these intervals, we can use conformal prediction (Vovk et al., 2005; Romano et al., 2019), which we review in Subsection 3.2 and subsequently explain in Subsection 4.1. We primarily use Conformalized Quantile Regression (CQR) from Romano et al. (2019) as our calibration method. While SEMF generates intervals through latent variable sampling, we apply CQR to these raw intervals, ensuring valid marginal coverage while assessing the underlying interval generation quality of SEMF. The CQR procedure is detailed in algorithm 3 below, which extends the general split conformal framework for quantile regression methods. CQR first trains separate quantile regressors to estimate the $\alpha/2$ and $1-\alpha/2$ quantiles, then uses a calibration set to compute conformity scores as $S_i = \max\{\hat{q}_{\alpha/2}(X_i) - Y_i, Y_i - \hat{q}_{1-\alpha/2}(X_i)\}$. The method then adjusts the initial quantile predictions by adding a correction factor derived from the empirical quantile of these conformity scores, ensuring finite-sample coverage guarantees. CQR has proven particularly effective for regression problems where the underlying model can produce reasonable quantile estimates, making it a strong baseline for uncertainty quantification tasks.

Algorithm 2 SEMF Inference: single test example

Require: $\theta^*, \phi_1^*, \phi_2^*, x_1, x_2, R$ {Trained parameters and test input}
Ensure: $\{z_r\}_{r=1}^R$ {R samples from the latent distribution}
1: **for** $r = 1, \dots, R$ **do**
2: Simulate $z_{1,r} \sim p_{\phi_1^*}(\cdot|x_1)$ {Sample from encoder 1}
3: Simulate $z_{2,r} \sim p_{\phi_2^*}(\cdot|x_2)$ {Sample from encoder 2}
4: Set $z_r = [z_{1,r}, z_{2,r}]$ {Concatenate latent representations}
5: **end for**
6: {Applications in Section 2.4}
7: **Point prediction:** $\hat{y} \leftarrow \frac{1}{R}\sum_{r=1}^{R} f_{\theta^*}(z_r)$
8: **Prediction intervals:** $\hat{y}_{r,s} \sim p_{\theta^*}(y|z_r)$ for $s = 1, \dots, R$

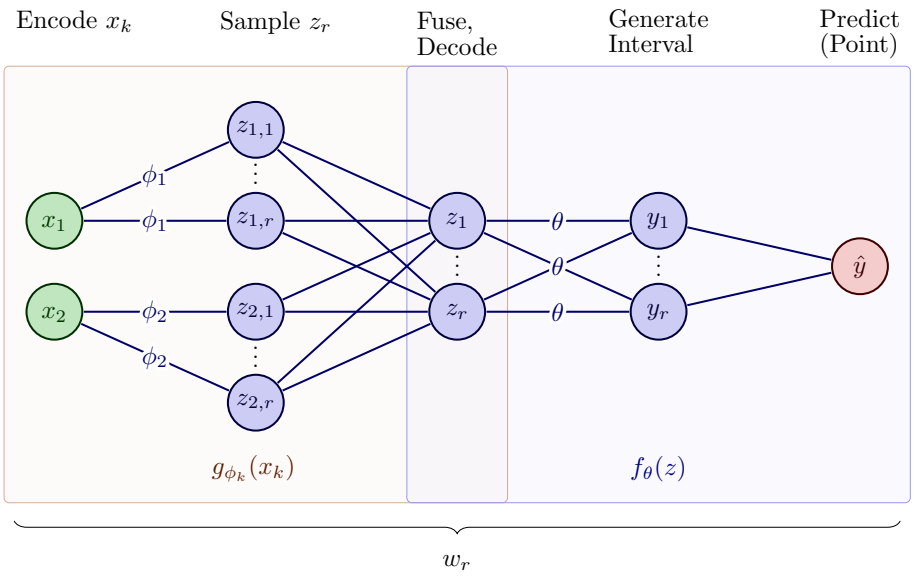

Figure 1: Inference procedure with the SEMF's learnable parameters ϕ_k and θ. Here, we illustrate the number of input sources k as $k = 1, 2$

Algorithm 3 Conformalized Quantile Regression (CQR)

Require: Training data $\mathcal{D}_{train}$, calibration data $\mathcal{D}_{calib} = \{(X_i, Y_i)\}_{i=1}^{n_{calib}}$, test input X_{test}, significance level α

Ensure: Prediction interval $[L_{CQR}, U_{CQR}]$ with coverage guarantee $\mathbb{P}(Y_{test} \in [L_{CQR}, U_{CQR}]) \geq 1 - \alpha$

Train quantile regressors $\hat{f}^{\alpha/2}$ and $\hat{f}^{1-\alpha/2}$ on $\mathcal{D}_{train}$
{Step 1: Compute initial quantile predictions on calibration set}
for $i = 1, \ldots, n_{calib}$ **do**
  $\hat{L}_i \leftarrow \hat{f}^{\alpha/2}(X_i)$ {Lower quantile prediction}
  $\hat{U}_i \leftarrow \hat{f}^{1-\alpha/2}(X_i)$ {Upper quantile prediction}
end for
{Step 2: Compute conformity scores on calibration set}
for $i = 1, \ldots, n_{calib}$ **do**
  $S_i \leftarrow \max\{\hat{L}_i - Y_i, Y_i - \hat{U}_i\}$ {CQR conformity score}
end for
{Step 3: Find quantile correction factor}
Sort $\{S_i\}_{i=1}^{n_{calib}}$ in increasing order
$\hat{q} \leftarrow \lceil (1-\alpha)(n_{calib}+1) \rceil$-th smallest score {If index $> n_{calib}$, use $\max_i S_i$}
{Step 4: Apply correction to test prediction}
$\hat{L}_{test} \leftarrow \hat{f}^{\alpha/2}(X_{test})$, $\hat{U}_{test} \leftarrow \hat{f}^{1-\alpha/2}(X_{test})$
$L_{CQR} \leftarrow \hat{L}_{test} - \hat{q}$, $U_{CQR} \leftarrow \hat{U}_{test} + \hat{q}$
return $[L_{CQR}, U_{CQR}]$

3. Related Work

3.1. Latent Representation Learning

Latent-representation learning typically relies on the encoder-decoder architecture introduced in Section 2. Classic examples are Auto-Encoders (AEs) and VAEs: the encoder g_ϕ maps a sample input x to a latent variable z, and the decoder f_θ reconstructs the input $\hat{x} = f_\theta(z)$. Training jointly optimizes the parameter vectors ϕ and θ by minimizing the reconstruction loss. Unlike AEs, which focus solely on reconstruction, VAEs introduce a variational objective that aims to model the data distribution through the marginal likelihood $p_\theta(x)$ while fitting approximate posterior distributions over the latent variables (Kingma and Welling, 2014; David M. Blei and McAuliffe, 2017b). However, directly maximizing this likelihood is difficult (David M. Blei and McAuliffe, 2017b). Thus, alternative methods exist, such as maximizing the ELBO, which provides guarantees on the log-likelihood (Balakrishnan et al., 2017).

For supervised and semi-supervised tasks, latent representation learning can include task-specific predictions (Kingma et al., 2014). More specifically, models such as AEs follow the classical encoder-decoder objective while training a predictor $h_\psi(z)$ through an additional layer or model to estimate the output y. This dual objective facilitates the learning of more task-relevant embeddings (Zhuang et al., 2015; Le et al., 2018). Semi-supervised VAEs are similar, with the distinction that they couple the reconstruction loss of the unlabeled data with a variational approximation of latent variables. This is effective even with sparse labels (Ji et al., 2020; Zhuang et al., 2023).

The EM algorithm has already been used for supervised learning tasks using specific models (Ghahramani and Jordan, 1993; Williams et al., 2005; Louiset et al., 2021), where the goal has been point prediction with GMMs. Similarly, the EM algorithm adapts well to minimal supervision (Luo et al., 2020), as well as using labeled and unlabeled data in semi-supervised settings for both single and multiple modalities (He and Jiang, 2022; Xu et al., 2024). Our work differs in that we modify the use of MC sampling to generate prediction intervals with any ML model, and in theory, under any distribution.

3.2. Prediction Intervals

Crucial for estimating uncertainty, prediction intervals provide a range for regression outcomes. Common approaches include Bayesian methods (Williams and Rasmussen, 1995; Hensman et al., 2015; Gal and Ghahramani, 2016), ensemble techniques (Breiman, 2001; Lakshminarayanan et al., 2017; Malinin et al., 2021), and quantile regression (Koenker and Bassett, 1978; Koenker and Hallock, 2001). Quantile regression specifically utilizes the pinball loss function to target desired quantiles, proving effective even for non-parametric models (Steinwart and Christmann, 2011) and asymmetric distributions (Koenker and Hallock, 2001). This loss is valuable both for individual models and for refining quantile estimates when used with ensembles (Meinshausen and Ridgeway, 2006)—a loss that can also be learned jointly as demonstrated by Simultaneous Quantile Regression (SQR) (Tagasovska and Lopez-Paz, 2019). SQR represents a powerful neural network approach that jointly estimates multiple quantiles while maintaining their proper ordering, making it particularly effective for constructing prediction intervals and a strong baseline for uncertainty quantification tasks.

Complementary to these approaches, conformal prediction (CP) provides a general framework for constructing and calibrating prediction intervals for any model (Vovk et al., 2022). Assuming data exchangeability, CP adjusts intervals to ensure they meet a pre-specified coverage probability, thereby enhancing reliability in applications demanding rigorous uncertainty quantification. Recent advances in CP include its integration with ensemble models for time series (Xu and Xie, 2021, 2023) and Bayesian approaches, such as Conformal Bayesian model averaging (CBMA), which aggregates conformity scores from several Bayesian models to obtain optimal prediction sets with guarantees (Bhagwat et al., 2025). Furthermore, several CP frameworks that incorporate data-dependent score functions have recently been proposed to improve the efficiency and validity of intervals derived via quantile estimators (Ge et al., 2024), boosting methods (XGBoost) to refine conformity scores (Xie et al., 2024), and locally adaptive techniques to enhance conditional validity (Colombo, 2023).

4. Experimental Setup

4.1. Models

Our baseline consists of quantile regression models based on eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin, 2016), Extremely Randomized Trees (ET) (Geurts et al., 2006), and neural networks using SQR (Tagasovska and Lopez-Paz, 2019), all summarized and depicted in Table 1. To ensure consistency in our experimental setup, we align the families and hyper-parameters of p_ϕ and p_θ with our baseline models. For example, in the case of XGBoost in SEMF, we use K XGBoosts, $g_{\phi_k}(x_k)$, one for each input x_k, $k = 1, \ldots, K$, and one XGBoost for $f_\theta(z)$ with the same hyper-parameters. We refer to the SEMF's adoption of these models as MultiXGBs, MultiETs, and MultiMLPs. When establishing prediction intervals, we conformalize our prediction intervals according to (Romano et al., 2019) at an uncertainty tolerance of 5% for both the baseline and SEMF (Eq. 14).

Regarding notable hyperparameters, we target larger σ_k values that introduce more noise and produce better intervals than point predictions. However, if the primary goal is to produce better point predictions, this hyperparameter should be set to a smaller value. We do not present the point prediction results in this paper, though they can be found in our code implementation. For several datasets, we set σ_k to `train_residual_models`, which trains separate models to predict the scale of the latent variables individually and is more suitable for heteroscedastic noise patterns. This approach often yields substantially improved interval predictions by capturing local uncertainty structures, though at the cost of longer computation times due to the additional model training. In our experience, using `train_residual_models` for σ_k is recommended as a starting point for new applications, with fixed values being a computationally efficient alternative once the behavior of the dataset is better understood. The optimal models are then trained and tested with five different seeds, and the results are averaged, and the variability of the results is presented. The point prediction, $\hat{y}$, uses the mean inferred values. Appendix A contains more details on the hyper-parameters for each SEMF model and dataset introduced below.

SEMF	Base Model	Interval Prediction Baseline
MultiXGBs	XGBoost Trees: 100, Maximum depth: 6, Early stopping steps: 10	Quantile XGBoost Same as point prediction baseline, XGBoost
MultiETs	Extremely Randomized Trees Trees: 100, Maximum depth: 10	Quantile Extremely Randomized Trees Same as point prediction baseline, Extremely Randomized Trees
MultiMLPs	Deep Neural Network Hidden layers: 2, Nodes per layer: 100, Activation functions: ReLU, Epochs: 1000 or 5000, Learning rate: 0.001, Batch training, Early stopping steps: 100	Simultaneous Quantile Regression Same as point prediction baseline, Deep Neural Network

Table 1: SEMF models, baselines, and hyper-parameters.

4.2. Datasets

4.2.1. Simulations

In our experiments, we generate synthetic datasets with 10^3 observations and $k = 2$ predictors from a standard normal distribution according to

$$y = f(x) + \epsilon, \tag{15}$$

where $f(x)$ in a simple setup is defined as

$$f(x) = \sum_{i=1}^{k} \cos(x_i) \tag{16}$$

to isolate noise effects, and in a more complex setup, we instead generate

$$f(x) = \sum_{i=1}^{k} \Big(x_i^2 + 0.5 \, \sin(3 \, x_i) \Big) \tag{17}$$

to introduce nonlinearity and heteroscedasticity. ϵ is drawn from one of four distributions: $\mathcal{N}(0, 0.5)$, $\mathcal{U}(-0.5, 0.5)$, a centered log-normal, or Gumbel$(0, 0.5)$. Figure 2 illustrates an example of Eq. 17 with $k = 1$ (for simplicity) and 500 observations, where blue dots indicate predictions with 95% prediction intervals and the red curve denotes $f(x)$.

For our simulation experiments, the underlying data is generated from a fixed seed with variability only in the model seeds (Subsection 4.1). We evaluate all three model variants (MultiXGBs, MultiETs, and MultiMLPs) against standard quantile regression baselines. For MultiXGBs and MultiETs, we use $R = 10$ with 10 nodes per latent dimension, while MultiMLPs use $R = 5$ with 20 nodes per latent dimension. All models employ early stopping with appropriate patience settings to prevent overfitting on the synthetic data. The data scaling and splits are the same as the benchmark data below.

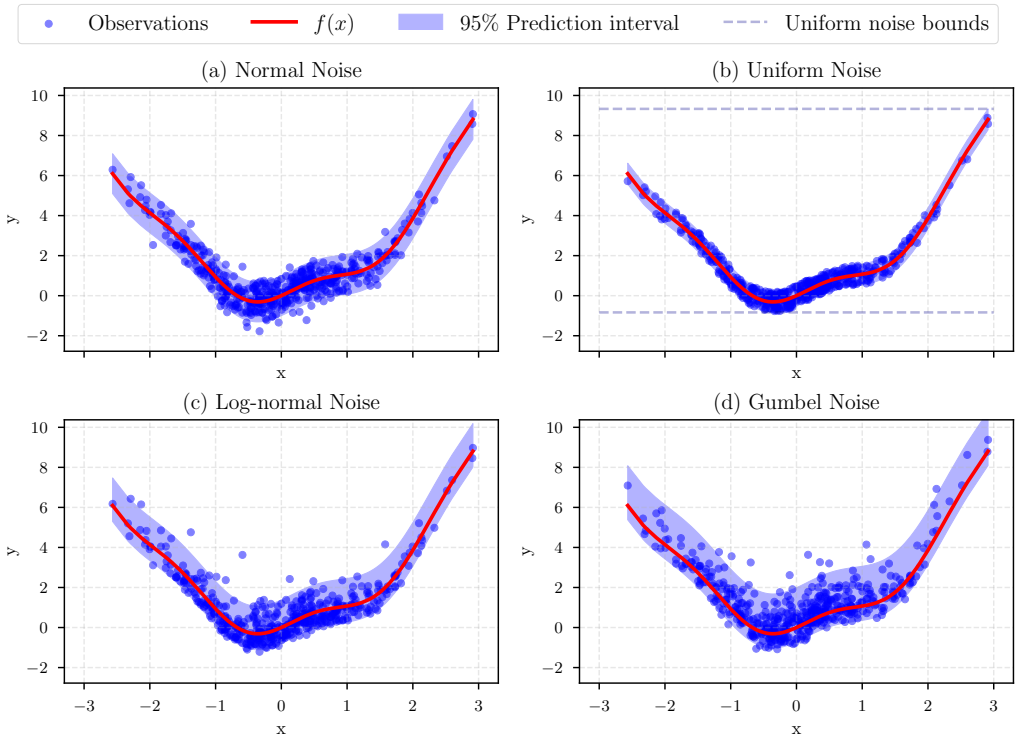

Figure 2: Prediction intervals under different noise distributions for a one-dimensional x generated from 500 observations. Each panel shows predictions (blue dots) with 95% prediction intervals (shaded regions) for a model trained on data with (a) Normal, (b) Uniform, (c) Log-normal, and (d) Gumbel noise. The red curve denotes $f(x)$, the underlying (deterministic) function.

4.2.2. Benchmark data

We systematically curate a subset of datasets from the OpenML-CTR23 (Fischer et al., 2023) benchmark suite to evaluate and carry out our experiments. Initially comprising 35 datasets, we apply an exclusion criteria to refine this collection to 11 datasets. The details and overview are in Appendix B. We remove duplicated rows from all the datasets and carry out the scaling of all predictors, including the outcome. The features of these datasets are then treated as separate inputs to SEMF. In all our datasets, 70% of the data is used to train all models, 15% as a hold-out validation set to monitor SEMF's performance, and 15% to evaluate the models. To combat overfitting, baseline models that benefit from early stopping are allocated another 15% from the training data. Lastly, it is essential to note that all data in SEMF are processed batch-wise, without employing mini-batch training, to ensure consistency and stability in the training process.

4.3. Metrics

The evaluation of prediction intervals employs Prediction Interval Coverage Probability (PICP), Normalized Mean Prediction Interval Width (NMPIW), Continuous Ranked Probability Score (CRPS) and the quantile (pinball) loss. Detailed definitions and closed-form expressions for these metrics (with the CRPS computed under a uniformity assumption on the predictive distribution) are provided in Appendix C. In addition, we introduce the Coverage-Width Ratio (CWR)

$$\text{CWR} = \frac{\text{PICP}}{\text{NMPIW}}, \tag{18}$$

which quantifies the trade-off between reliability and precision.

In our case, measuring the performance of SEMF over the baseline models is far more critical than reviewing absolute metrics in isolation. For any metric above, this is computed as

$$\text{Metric}_{\Delta}(\%) = \left(\frac{\text{Metric}_{\text{SEMF}} - \text{Metric}_{\text{Baseline}}}{\text{Metric}_{\text{Baseline}}} \right) \times 100, \tag{19}$$

on which we base our decisions for selecting the best hyper-parameters as explained in Appendix C.2.

5. Experiments

5.1. Simulated Data

Table 2 and Table 3 summarize the performance of our SEMF variants on synthetic datasets generated using two predictors. The PICP shows that on average, all models across all experiments were able to achieve at least the 95% desired coverage probability.

	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	PICP
MultiXGBs						
normal	**18% [10:35]**	**16% [10:28]**	**6% [3:10]**	**11% [5:17]**	-1% [-3:0]	0.95±0.01
uniform	**25% [-2:58]**	**19% [-5:39]**	**10% [-1:23]**	**18% [0:32]**	-1% [-5:2]	0.96±0.03
lognormal	**28% [8:59]**	**21% [8:39]**	**13% [8:18]**	**7% [2:14]**	-1% [-3:0]	0.96±0.01
gumbel	**10% [-5:20]**	**8% [-9:18]**	**8% [1:14]**	**8% [0:16]**	0% [-3:4]	0.96±0.01
MultiETs						
normal	**15% [1:32]**	**14% [0:27]**	**5% [0:8]**	**8% [0:16]**	-2% [-4:1]	0.95±0.01
uniform	**16% [3:35]**	**13% [-1:28]**	**6% [3:10]**	**13% [8:21]**	0% [-3:4]	0.97±0.01
lognormal	**5% [-14:18]**	**3% [-18:17]**	**9% [5:13]**	-3% [-10:2]	0% [-2:2]	0.97±0.01
gumbel	**3% [-4:14]**	**3% [-7:14]**	**8% [6:11]**	**1% [-2:5]**	0% [-2:2]	0.96±0.01
MultiMLPs						
normal	**3% [-2:9]**	**3% [-2:8]**	**1% [-2:4]**	**2% [-3:5]**	0% [-1:0]	0.96±0.01
uniform	**4% [-2:13]**	**4% [-2:10]**	**2% [-1:8]**	**7% [-5:23]**	0% [-4:3]	0.97±0.02
lognormal	**7% [-4:18]**	**6% [-5:16]**	**3% [-2:7]**	**4% [0:9]**	0% [-1:1]	0.97±0.01
gumbel	**5% [-3:13]**	**5% [-5:14]**	0% [-2:3]	0% [-4:4]	-1% [-2:2]	0.96±0.02

Table 2: Test results (95% prediction intervals) for 1000 observations generated using cosine $f(x)$ with 2 predictors, and an additive noise (ϵ) belonging to one of the four distributions. Relative metrics are shown over 5 seeds as 'mean [min:max]', and absolute metrics as 'mean ± std'. Average performance over the baseline is highlighted in bold.

	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	PICP
MultiXGBs						
normal	**39% [28:65]**	**28% [22:41]**	**29% [26:34]**	**23% [16:35]**	0% [-3:0]	0.95±0.01
uniform	**65% [54:80]**	**39% [34:44]**	**41% [39:47]**	**25% [16:32]**	0% [-1:1]	0.96±0.02
lognormal	0% [-13:7]	-2% [-16:6]	**20% [14:27]**	**4% [-4:16]**	**2% [0:5]**	0.96±0.02
gumbel	**23% [0:49]**	**17% [0:35]**	**20% [9:30]**	**16% [3:21]**	0% [-3:4]	0.96±0.01
MultiETs						
normal	**23% [7:35]**	**19% [7:26]**	**14% [0:21]**	**12% [-4:23]**	-1% [-2:1]	0.95±0.02
uniform	**40% [25:71]**	**28% [20:42]**	**24% [16:33]**	**12% [-4:21]**	0% [-3:2]	0.97±0.02
lognormal	**8% [-9:33]**	**5% [-13:27]**	**11% [0:16]**	**6% [-6:12]**	**1% [-2:3]**	0.97±0.01
gumbel	**29% [-3:68]**	**19% [-4:42]**	**18% [5:30]**	**15% [0:28]**	0% [-2:1]	0.97±0.01
MultiMLPs						
normal	**5% [-14:24]**	**1% [-20:21]**	**7% [2:13]**	**13% [6:20]**	**2% [-2:4]**	0.95±0.01
uniform	**37% [15:66]**	**25% [13:41]**	**19% [13:27]**	**29% [9:40]**	**1% [-2:6]**	0.97±0.02
lognormal	**1% [-12:10]**	**1% [-15:11]**	**4% [2:8]**	-1% [-10:4]	0% [-2:1]	0.96±0.01
gumbel	**15% [-6:38]**	**11% [-8:30]**	**10% [-3:23]**	**11% [-5:23]**	0% [-3:2]	0.96±0.00

Table 3: Test results (95% prediction intervals) for 1000 observations generated using quadratic $f(x)$ with 2 predictors, and an additive noise (ϵ) belonging to one of the four distributions. Relative metrics are shown over 5 seeds as 'mean [min:max]', and absolute metrics as 'mean ± std'. Average performance over the baseline is highlighted in bold.

For the cosine experiments (Table 2), all three methods (MultiXGBs, MultiETs, and MultiMLPs) exhibit positive relative improvements over the baseline in terms of the CWR, NMPIW, CRPS, and pinball loss. For instance, under normal noise, the MultiXGBs variant improves the ΔCWR by an average of 18% (ranging between 10% and 35%), while still preserving a PICP of approximately 0.95. Similar positive trends are seen across the uniform and lognormal noise conditions. Although under Gumbel noise the improvements are less pronounced, SEMF still delivers performance that is at least on par with the baseline.

For the quadratic experiments (Table 3), which represents a more complex heteroscedastic case, the improvements are even more substantial. Under normal noise, MultiXGBs achieve an average improvement of 39% in ΔCWR, and, when the noise is uniform, gains can be as high as 65%. There is some variability—for example, under lognormal noise, the improvement in ΔCWR for MultiXGBs is close to zero—but overall, the results indicate that SEMF produces tighter and more reliable prediction intervals compared to the baseline. Similar to the cosine case, the Gumbel noise setting shows the smallest relative gains; nevertheless, the performance under Gumbel noise remains competitive with the baseline.

5.2. Benchmark Data

We trained and tested 165 models corresponding to the three model types—MultiXGBs, MultiETs, and MultiMLPs—across 11 datasets, using five seeds for each combination. Table 4 presents the means and standard deviations for our metrics aggregated over the five seeds. Appendix E includes the results from each individual run.

The results on OpenML-CTR23 datasets reveal distinct performance patterns across our model variants. MultiXGBs shows improvement over the baseline in most datasets, with notable results on *naval_propulsion_plant* (ΔCWR: 163%, ΔNMPIW: 61%) and *en-*

	Relative Metrics					Absolute Metrics		
Dataset	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
MultiXGBs								
space_ga	**11% [4:21]**	**9% [2:19]**	**11% [10:12]**	**9% [6:13]**	0% [-2:2]	4.51±1.64	0.24±0.08	0.95±0.01
cpu_activity	**5% [-3:11]**	**5% [-4:11]**	**16% [10:26]**	**15% [0:36]**	0% [-1:1]	9.77±0.55	0.10±0.01	0.95±0.01
naval_propulsion_plant	**163% [114:200]**	**61% [53:68]**	**72% [66:74]**	**45% [31:50]**	0% [-3:2]	8.27±0.68	0.12±0.01	0.95±0.01
miami_housing	**5% [-4:12]**	**5% [-4:10]**	**12% [8:17]**	-12% [-21:-5]	0% [-1:0]	8.75±0.36	0.11±0.00	0.95±0.00
kin8nm	**18% [16:19]**	**16% [14:17]**	**23% [20:25]**	**8% [3:11]**	-1% [-1:-1]	2.31±0.04	0.41±0.01	0.94±0.00
concrete_compressive_strength	**40% [24:73]**	**29% [20:45]**	**32% [27:35]**	**12% [4:25]**	-3% [-5:-1]	3.22±0.21	0.29±0.02	0.94±0.02
cars	**32% [10:67]**	**24% [10:42]**	**27% [20:42]**	**17% [6:32]**	-1% [-3:0]	5.33±0.55	0.18±0.02	0.95±0.01
energy_efficiency	**128% [72:230]**	**53% [39:68]**	**67% [59:81]**	**54% [43:73]**	**3% [-2:8]**	13.95±2.46	0.07±0.01	0.96±0.00
california_housing	**1% [-1:4]**	**1% [-1:4]**	**18% [15:23]**	-8% [-10:-3]	0% [-1:0]	2.24±0.08	0.42±0.02	0.95±0.01
airfoil_self_noise	-8% [-32:6]	-12% [-47:6]	**2% [-36:20]**	-17% [-43:4]	0% [-1:0]	2.29±0.38	0.44±0.10	0.97±0.01
QSAR_fish_toxicity	**19% [2:53]**	**14% [1:35]**	**13% [8:24]**	**15% [8:30]**	0% [-1:2]	1.71±0.10	0.57±0.04	0.98±0.01
MultiETs								
space_ga	**5% [-2:15]**	**6% [-2:16]**	**8% [6:11]**	-6% [-9:-3]	-1% [-3:0]	4.38±1.82	0.26±0.09	0.95±0.02
cpu_activity	**2% [-1:5]**	**6% [2:10]**	**3% [-2:7]**	-18% [-22:-11]	-4% [-5:-3]	8.25±0.23	0.11±0.00	0.95±0.01
naval_propulsion_plant	**144% [118:160]**	**60% [56:63]**	**68% [64:72]**	**44% [40:52]**	-2% [-6:0]	3.65±0.25	0.26±0.02	0.95±0.01
miami_housing	-9% [-15:-5]	-10% [-19:-4]	-3% [-7:0]	-73% [-81:-67]	0% [0:1]	6.44±0.27	0.15±0.01	0.95±0.00
kin8nm	**7% [5:10]**	**7% [4:8]**	**14% [12:17]**	-1% [-3:0]	0% [0:1]	2.13±0.05	0.45±0.01	0.95±0.01
concrete_compressive_strength	**6% [-14:26]**	**4% [-20:23]**	**9% [4:12]**	-3% [-12:3]	0% [-3:3]	3.05±0.25	0.31±0.03	0.95±0.01
cars	**15% [-25:54]**	**7% [-40:36]**	**6% [-8:20]**	**8% [-5:28]**	-1% [-3:4]	4.89±0.81	0.20±0.04	0.95±0.02
energy_efficiency	**15% [1:26]**	**14% [1:23]**	**3% [-15:12]**	-11% [-80:25]	-2% [-3:0]	15.93±2.57	0.06±0.01	0.95±0.03
california_housing	**1% [-11:9]**	**2% [-11:11]**	**21% [16:25]**	-22% [-26:-17]	-2% [-3:-1]	1.70±0.09	0.56±0.04	0.95±0.01
airfoil_self_noise	-13% [-39:22]	-20% [-66:18]	-17% [-52:19]	-26% [-61:18]	-1% [-2:1]	2.25±0.46	0.45±0.10	0.97±0.01
QSAR_fish_toxicity	-3% [-20:17]	-6% [-26:18]	-3% [-16:6]	-8% [-19:2]	-1% [-4:0]	1.71±0.22	0.58±0.09	0.97±0.01
MultiMLPs								
space_ga	**5% [-6:25]**	**6% [-4:21]**	**2% [-9:13]**	0% [-13:8]	-2% [-3:0]	5.10±1.78	0.21±0.07	0.94±0.02
cpu_activity	-14% [-16:-9]	-16% [-20:-11]	-9% [-14:-1]	-3% [-11:17]	0% [-1:1]	8.57±0.14	0.11±0.00	0.95±0.01
naval_propulsion_plant	**23% [-2:84]**	**15% [-1:46]**	**17% [-3:46]**	-7% [-25:33]	0% [-1:1]	12.78±1.86	0.08±0.01	0.95±0.01
miami_housing	-21% [-35:-13]	-28% [-56:-15]	-13% [-23:-9]	-26% [-38:-11]	0% [-1:0]	9.06±1.06	0.11±0.02	0.95±0.01
kin8nm	**9% [2:18]**	**8% [2:16]**	**8% [5:14]**	**8% [0:15]**	0% [-2:1]	4.74±0.17	0.20±0.01	0.94±0.01
concrete_compressive_strength	**31% [-5:73]**	**21% [-2:43]**	**16% [-2:35]**	**7% [-19:31]**	-2% [-4:2]	3.43±0.17	0.28±0.02	0.95±0.02
cars	-12% [-44:21]	-21% [-89:22]	-12% [-23:5]	-19% [-29:-13]	0% [-5:7]	5.40±1.25	0.19±0.04	0.96±0.03
energy_efficiency	**87% [46:149]**	**45% [30:63]**	**42% [33:56]**	**23% [0:50]**	-1% [-6:5]	21.91±3.25	0.04±0.01	0.94±0.02
california_housing	-11% [-14:-7]	-12% [-16:-6]	-2% [-5:0]	-9% [-13:-3]	0% [-1:0]	2.13±0.08	0.45±0.02	0.95±0.01
airfoil_self_noise	**42% [16:63]**	**28% [14:39]**	**30% [17:43]**	**23% [5:32]**	-1% [-2:1]	4.92±0.31	0.20±0.01	0.97±0.01
QSAR_fish_toxicity	**8% [-15:36]**	**6% [-19:28]**	**1% [-9:13]**	**3% [-11:15]**	-1% [-3:1]	1.71±0.22	0.58±0.08	0.97±0.01

Table 4: Test results for all models with at 95% quantiles aggregated over five seeds. For relative metrics, values are shown as 'mean% [min:max]', and absolute metrics as 'mean ± std'. Performance over the baseline is highlighted in bold.

ergy_efficiency (ΔCWR: 128%, ΔNMPIW: 53%). For *concrete_compressive_strength* and *cars*, MultiXGBs yields moderate improvements in interval quality (ΔCWR: 40% and 32%, respectively). On *airfoil_self_noise*, however, MultiXGBs performs slightly below the baseline. While maintaining PICP values comparable to the baseline (within ±3%), MultiXGBs generally produces narrower prediction intervals, resulting in higher CWR values.

MultiETs shows more variable performance. It achieves relative improvements on *naval_propulsion_plant* (ΔCWR: 144%, ΔNMPIW: 60%) and modest gains on *energy_efficiency* and *cars*, but underperforms on datasets like *miami_housing*, *airfoil_self_noise*, as well as *QSAR_fish_toxicity*. Interestingly, MultiETs often shows positive ΔCRPS values even when the ΔPinball is negative, suggesting a trade-off between interval sharpness and quantile calibration.

MultiMLPs exhibits mixed results across datasets. It performs well on *energy_efficiency* (ΔCWR: 87%, ΔNMPIW: 45%) and *airfoil_self_noise* (ΔCWR: 42%, ΔNMPIW: 28%), but shows lower performance on *cpu_activity*, *miami_housing*, and *california_housing*. The absolute CWR for MultiMLPs on *energy_efficiency* (21.91) indicates efficient intervals that maintain the target coverage with minimal width.

Across all models, PICP values remain within the target range of approximately 0.95±0.03, indicating that SEMF maintains proper coverage while often reducing interval widths. These results suggest that SEMF's benefits are most pronounced when applied to XGBoost, followed by the randomized trees and neural networks, the order depending on the metric.

5.3. Discussion

Our results indicate that overall, SEMF enhances uncertainty quantification in regression tasks, with varying effectiveness depending on the base model type and dataset characteristics. The framework often produces narrower prediction intervals while maintaining coverage targets, addressing the trade-off between interval width and reliability. The mechanism behind SEMF's effectiveness appears to be model-dependent. For tree-based methods like XGBoost, which have limited capacity to model uncertainty directly, SEMF's latent representation learning introduces a form of distributional modeling that enables more efficient uncertainty quantification. The iterative sampling and weighting procedure in the EM algorithm allows XGBoost to better capture the uncertainty structure in the data. This explains why MultiXGBs show the most consistent improvements across datasets.

Neural networks, which inherently have greater representational capacity, benefit differently from SEMF. The framework's structured approach to uncertainty quantification through latent variables complements the neural network's flexibility, particularly in datasets with complex feature interactions like *energy_efficiency*. However, this advantage is not universal across all datasets, suggesting that the interaction between SEMF's latent structure and the neural network's own representational capabilities may sometimes create redundancies. Even in datasets where SEMF shows more modest improvements (such as *space_ga* and *kin8nm*), it still performs at least on par with SQR, our strong neural network baseline for simultaneous quantile estimation. This performance floor across diverse datasets highlights SEMF's robustness in different modeling scenarios.

The more variability in the performance of MultiETs compared to XGBoost can be attributed to two factors. First, the randomized splitting mechanism in ETs introduces

inherent variability that may interfere with SEMF's iterative refinement process. Second, ETs sometimes had less stable estimates for the weights (Eq. 5), affecting the quality of the latent representations learned during training. This hypothesis is supported by the higher standard deviations observed in MultiETs' performance metrics compared to the other models.

Dataset characteristics also influence SEMF's effectiveness. The framework shows particular strength on datasets like *naval_propulsion_plant* and *energy_efficiency*, which feature complex interactions among input variables that benefit from latent representation learning. Particularly for *naval_propulsion_plant*, the overall y output is uniformly distributed, which the trees baselines fail to capture, but SEMF is more robust to this. Conversely, on simpler datasets or those with highly linear relationships, the additional complexity introduced by SEMF may not provide significant advantages over traditional quantile regression approaches. The relationship between CRPS and pinball Loss also provides insight into SEMF's behavior. For several dataset-model combinations (e.g., MultiXGBs on *miami_housing* and *california_housing*), we observe positive ΔCRPS alongside negative ΔPinball values. This divergence stems from fundamental differences in what these metrics evaluate. CRPS assesses the overall quality of the prediction interval, favoring narrow intervals that are well-centered around the true value. In contrast, pinball Loss measures the calibration of individual quantile predictions, penalizing miscalibrations at the tails more heavily. SEMF typically produces sharper intervals that may sacrifice some calibration at the exact quantile levels, resulting in better CRPS but potentially worse pinball loss compared to baseline models with wider but more conservatively calibrated intervals.

The stability of SEMF across different random seeds, evidenced by the relatively small standard deviations in metrics like PICP and NMPIW, contrasts with the higher variability in baseline models. This suggests that SEMF's sampling-based approach leads to more consistent uncertainty estimates, an important consideration for applications where predictive stability is required. The tight clustering of PICP values around the target level (0.95) across diverse datasets demonstrates SEMF's ability to maintain calibration while improving efficiency. Further, our experimental design focused on the conformalized performance of SEMF, with CP applied consistently across all models. The performance improvements observed with this standardized approach suggest that SEMF could benefit further from specialized calibration techniques designed to leverage its sampling-based uncertainty estimation. Future work might explore adaptive conformalization methods that account for the specific characteristics of SEMF's predictive distributions.

6. Conclusion

This paper introduces the Supervised Expectation-Maximization Framework (SEMF), a novel model-agnostic approach for generating prediction intervals in datasets with any machine learning model. SEMF draws from the EM algorithm for supervised learning to devise latent representations that produce better prediction intervals than quantile regression. Our comprehensive evaluation demonstrates SEMF's effectiveness in two complementary settings. First, controlled simulations across different noise distributions (normal, uniform, log-normal, and Gumbel) and data generating functions (cosine and quadratic with periodic components) show SEMF's robustness to varying data characteristics, with particularly

strong improvements for heteroscedastic data patterns. Second, a set of 165 experimental runs on 11 real-world benchmark datasets with three different model types confirmed that SEMF outperforms quantile regression in practical applications, particularly when using XGBoost, which intrinsically lacks latent representations. The framework's ability to maintain consistent performance across different noise distributions while producing narrower intervals underscores its potential in various application domains and opens new avenues for further exploration of supervised latent representation learning and uncertainty estimation.

7. Limitations & Future Work

The primary limitation of this study was its reliance on the normality assumption, which may not fully capture the potential of SEMF across diverse data distributions. Furthermore, in our simulations, we show that our framework can learn non-normal patterns; however, further investigation examination of how SEMF under broader distributional parameters can be interesting. The computational complexity of the approach presents another significant challenge, as the current implementation can be optimized for large-scale applications. Additionally, while the CWR metric is valid, it implicitly assumes that a 1% drop in PICP equates to a 1% reduction in NMPIW, thus assuming a uniform distribution. Evaluating CWR under various distributional assumptions would provide a more comprehensive assessment of its implications.

Future work presents several intriguing avenues for exploration. A promising direction is the application of SEMF in multi-modal data settings, where the distinct p_ϕ components of the framework could be adapted to process diverse data types—from images and text to tabular datasets—enabling a more nuanced and powerful approach to integrating heterogeneous data sources. This capability positions the framework as a versatile tool for addressing missing data challenges across various domains and can also help expand it to discrete and multiple outputs. Another valuable area for development is the exploration of methods to capture and leverage dependencies among input features, which could improve the model's predictive performance and provide deeper insights into the underlying data structure. These advancements can enhance the broader appeal of end-to-end approaches like SEMF in the ML community.

References

Sivaraman Balakrishnan, Martin J. Wainwright, and Bin Yu. Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1): 77 – 120, 2017. doi: 10.1214/16-AOS1435. URL https://doi.org/10.1214/16-AOS1435.

Pankaj Bhagwat, Linglong Kong, and Bei Jiang. CBMA: Improving conformal prediction through bayesian model averaging. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=BKSeNw2HIr.

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

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

Thomas Brooks, Dennis Pope, and Michael Marcolini. Airfoil self-noise and prediction. NASA Technical Report 1218, NASA, 1989.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Nicolo Colombo. On training locally adaptive cp. In Harris Papadopoulos, Khuong An Nguyen, Henrik Boström, and Lars Carlsson, editors, *Proceedings of the Twelfth Symposium on Conformal and Probabilistic Prediction with Applications*, volume 204 of *Proceedings of Machine Learning Research*, pages 384–398. PMLR, 13–15 Sep 2023. URL https://proceedings.mlr.press/v204/colombo23a.html.

Andrea Coraddu, Luca Oneto, Aessandro Ghio, Stefano Savio, Davide Anguita, and Massimo Figari. Machine learning approaches for improving condition-based maintenance of naval propulsion plants. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1):136–153, 2016. doi: 10.1177/1475090214540874. URL https://doi.org/10.1177/1475090214540874.

Alp Kucukelbir David M. Blei and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017a. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

Alp Kucukelbir David M. Blei and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017b. doi: 10.1080/01621459.2017.1285773. URL https://doi.org/10.1080/01621459.2017.1285773.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. ISSN 00359246. URL http://www.jstor.org/stable/2984875.

Michael W Dusenberry, Dustin Tran, Edward Choi, Jonas Kemp, Jeremy Nixon, Ghassen Jerfel, Katherine Heller, and Andrew M Dai. Analyzing the role of model uncertainty for electronic health records. In *Proceedings of the ACM Conference on Health, Inference, and Learning*, CHIL '20, page 204–213, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370462. doi: 10.1145/3368555.3384457. URL https://doi.org/10.1145/3368555.3384457.

Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. OpenML-CTR23 – a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*, 2023. URL https://openreview.net/forum?id=HebAOoMm94.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger,

editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/gal16.html.

Jiawei Ge, Debarghya Mukherjee, and Jianqing Fan. Optimal aggregation of prediction intervals under unsupervised domain shift. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ldXyNSvXEr.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

Z. Ghahramani. The kin datasets. https://www.cs.toronto.edu/~delve/data/kin/desc.html, 1996. Accessed: 2024-02-02.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521 (7553):452–459, 2015.

Zoubin Ghahramani and Michael Jordan. Supervised learning from incomplete data via an em approach. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann, 1993. URL https://proceedings.neurips.cc/paper_files/paper/1993/file/f2201f5191c4e92cc5af043eebfd0946-Paper.pdf.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E. Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007. doi: https://doi.org/10.1111/j.1467-9868.2007.00587.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2007.00587.x.

Wenchong He and Zhe Jiang. Semi-supervised learning with the em algorithm: A comparative study between unstructured and structured prediction. *IEEE Transactions on Knowledge and Data Engineering*, 34(6):2912–2920, 2022. doi: 10.1109/TKDE.2020.3019038.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

Tianchen Ji, Srikanth Vuppala, Girish V. Chowdhary, and K. Driggs-Campbell. Multi-modal anomaly detection for unstructured and uncertain environments. In *Conference on Robot Learning*, 2020. URL https://api.semanticscholar.org/CorpusID:229220208.

Kaggle. Moneyball dataset. https://www.kaggle.com/datasets/wduckett/moneyball-mlb-stats-19622012, 2017. Accessed: 2024-02-02.

Kaggle. Miami housing dataset. https://www.kaggle.com/datasets/deepcontractor/miami-housing-dataset, 2022. Accessed: 2024-02-02.

R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997. ISSN 0167-7152. doi: https://doi.org/10.1016/S0167-7152(96)00140-X. URL https://www.sciencedirect.com/science/article/pii/S016771529600140X.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

Roger Koenker and Kevin F. Hallock. Quantile regression. *The Journal of Economic Perspectives*, 15(4):143–156, 2001. ISSN 08953309. URL http://www.jstor.org/stable/2696522.

Shonda Kuiper. Introduction to multiple regression: How much is your car worth? *Journal of Statistics Education*, 16(3), 2008. doi: 10.1080/10691898.2008.11889579. URL https://doi.org/10.1080/10691898.2008.11889579.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Lei Le, Andrew Patterson, and Martha White. Supervised autoencoders: Improving generalization performance with unsupervised regularizers. *Advances in neural information processing systems*, 31, 2018.

Robin Louiset, Pietro Gori, Benoit Dufumier, Josselin Houenou, Antoine Grigis, and Edouard Duchesnay. Ucsl: A machine learning expectation-maximization framework for unsupervised clustering driven by supervised learning. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21*, pages 755–771. Springer, 2021.

Zhekun Luo, Devin Guillory, Baifeng Shi, Wei Ke, Fang Wan, Trevor Darrell, and Huijuan Xu. Weakly-supervised action localization with expectation-maximization multi-instance learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 729–745. Springer, 2020.

R. Todeschini M. Cassotti, D. Ballabio and V. Consonni. A similarity-based qsar model for predicting acute toxicity towards the fathead minnow (pimephales promelas). *SAR and QSAR in Environmental Research*, 26(3):217–243, 2015. doi: 10.1080/1062936X.2015.1018938. URL https://doi.org/10.1080/1062936X.2015.1018938. PMID: 25780951.

Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=1Jv6b0Zq3qi.

Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of machine learning research*, 7(6), 2006.

R. Kelley Pace and Ronald Barry. Quick computation of spatial autoregressive estimators. *Geographical Analysis*, 29(3):232–247, 1997. doi: https://doi.org/10.1111/j.1538-4632.1997.tb00959.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.1997.tb00959.x.

Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International conference on machine learning*, pages 4075–4084. PMLR, 2018.

C. Rasmussen, R. Neal, G. Hinton, D. van Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani. Computer activity dataset. http://www.cs.toronto.edu/~delve/data/datasets.html, 1996. Accessed: 2024-02-02.

Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/5103c3584b063c431bd1268e9b5e76fb-Paper.pdf.

Ingo Steinwart and Andreas Christmann. Estimating conditional quantiles with the help of the pinball loss. *Bernoulli*, 17(1), February 2011. ISSN 1350-7265. doi: 10.3150/10-bej267. URL http://dx.doi.org/10.3150/10-BEJ267.

Natasa Tagasovska and David Lopez-Paz. Single-model uncertainties for deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Xiaolin Tang, Kai Yang, Hong Wang, Jiahang Wu, Yechen Qin, Wenhao Yu, and Dongpu Cao. Prediction-uncertainty-aware decision-making for autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 7(4):849–862, 2022.

Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings*, 49:560–567, 2012. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2012.03.003. URL https://www.sciencedirect.com/science/article/pii/S037877881200151X.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. Conformal prediction: General case and regression. In *Algorithmic Learning in a Random World*, pages 19–69. Springer, 2022.

Vladimir V. V'yugin and Vladimir G. Trunov. Online learning with continuous ranked probability score. In Alex Gammerman, Vladimir Vovk, Zhiyuan Luo, and Evgueni Smirnov, editors, *Proceedings of the Eighth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 105 of *Proceedings of Machine Learning Research*, pages 163–177. PMLR, 09–11 Sep 2019. URL https://proceedings.mlr.press/v105/v-yugin19a.html.

Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

David Williams, Xuejun Liao, Ya Xue, and Lawrence Carin. Incomplete-data classification using logistic regression. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 972–979, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102474. URL https://doi.org/10.1145/1102351.1102474.

Tomasz Wisniewski and Arnold Polanski. Conformal prediction in financial risk assessment. In Alexander Gammerman, Vladimir Vovk, Zhiyuan Luo, Evgueni Smirnov, and Giovanni Cherubin, editors, *Proceedings of the Ninth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 128 of *Proceedings of Machine Learning Research*, pages 285–301. PMLR, 09–11 Sep 2020. URL https://proceedings.mlr.press/v128/wisniewski20a.html.

CF Jeff Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

Ran Xie, Rina Foygel Barber, and Emmanuel Candes. Boosted conformal prediction intervals. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Tw032H2onS.

Chen Xu and Yao Xie. Conformal prediction interval for dynamic time-series. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11559–11569. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/xu21h.html.

Chen Xu and Yao Xie. Sequential predictive conformal inference for time series. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Moucheng Xu, Yukun Zhou, Chen Jin, Marius de Groot, Daniel C Alexander, Neil P Oxtoby, Yipeng Hu, and Joseph Jacob. Expectation maximization pseudo labels. *Medical Image Analysis*, page 103125, 2024.

I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998. ISSN 0008-8846. doi: https://doi.org/10.1016/S0008-8846(98)00165-3. URL https://www.sciencedirect.com/science/article/pii/S0008884698001653.

Ting Zhou, Yuxin Jie, Yingjie Wei, Yanyi Zhang, and Hui Chen. A real-time prediction interval correction method with an unscented kalman filter for settlement monitoring of a power station dam. *Scientific Reports*, 13(1):4055, 2023.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: transfer learning with deep autoencoders. In *Twenty-fourth international joint conference on artificial intelligence*, IJCAI'15, page 4119–4125. AAAI Press, 2015. ISBN 9781577357384.

Yilin Zhuang, Zhuobin Zhou, Burak Alakent, and Mehmet Mercangöz. Semi-supervised variational autoencoders for regression: Application to soft sensors. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–8, 2023. doi: 10.1109/INDIN51400.2023.10218227.

Appendix A. Optimal set of hyper-parameters

Remark. The datasets mentioned here have been explained in Table 6 and Appendix B.

The hyper-parameter tuning for SEMF is implemented and monitored using Weights & Biases (Biewald, 2020). A random search is done in the hyper-parameter space for a maximum of 500 iterations on all 11 datasets, focusing on tuning the models only on the the non-conformalized performance. Key hyper-parameters are varied across a predefined set to balance accuracy and computational efficiency. The following grid is used for hyper-parameter tuning: the number of importance sampling operations $R \in \{5, 10, 25, 50, 100\}$ (100 is omitted for MultiMLPS), nodes per latent dimension $m_k \in \{1, 5, 10, 20, 30\}$, and standard deviations $\sigma_k \in \{0.001, 0.01, 0.1, 1.0\}$. For some datasets, the σ_k parameter is set to `train_residual_models`. This indicates that a separate residual model is trained to predict the latent-variable scale, which can be particularly suitable in heteroscedastic scenarios where noise levels vary across instances. Early stopping steps (PATIENCE) are set to five or ten, and R_{infer}, the R for inference as R_{infer}, is explored at [30, 50, 70]. For *california_housing* and *cpu_activity*, the R_{infer} value is set to 30, while for all the other datasets, it is set to 50. We do this to ensure efficient computation, speed, and memory usage (especially for the GPU). Finally, the option to run the models in parallel must be consistently enabled. Table 5 shows the optimal set of the selected hyper-parameters.

MultiXGBs and MultiMLPs benefit from early stopping to reduce computation time. Similarly, the baseline models for these instances use the same hyper-parameters for early stopping. Further, the number of epochs in the case of MultiMLPs is set as 1000, except for *energy_efficiency* and *QSAR_fish_toxicity*, where this is changed to 5000. Any model-specific hyperparameter we did not specify in this paper remains at the implementation's default value (e.g., the number of leaves in XGBoost from (Chen and Guestrin, 2016)). Along with the supplementary code, we provide three additional CSV files: one for the results and hyperparameters of all 165 runs and the other two for the optimal hyperparameters of SEMF models, both raw (directly from SEMF) and conformalized.

For training MultiXGBs and MultiETs, the computations are performed in parallel using CPU cores (Intel® Core™ i9-13900KF). Due to this, MultiETs are not fully deterministic and may exhibit slight variations between runs. For MultiMLPs, they are done on a GPU (NVIDIA® GeForce RTX™ 4090) which should give deterministic results across runs. All the computations are done on a machine with 32 GB of memory. The code provides further details on hardware and reproducibility.

Dataset	R	m_k	σ_k	Patience	R_{infer}
MultiXGBs					
space_ga	10	30	train_residual_models	5	50
cpu_activity	5	30	1	5	30
naval_propulsion_plant	5	30	0.01	5	50
miami_housing	5	10	train_residual_models	5	50
kin8nm	5	30	train_residual_models	10	50
concrete_compressive_strength	25	30	train_residual_models	5	50
cars	50	10	train_residual_models	10	50
energy_efficiency	5	1	0.01	10	50
california_housing	5	10	0.1	5	30
airfoil_self_noise	25	1	train_residual_models	10	50
QSAR_fish_toxicity	50	30	1	5	50
MultiETs					
space_ga	10	30	train_residual_models	10	50
cpu_activity	5	30	1	5	30
naval_propulsion_plant	5	30	0.01	5	50
miami_housing	10	10	train_residual_models	5	50
kin8nm	5	30	train_residual_models	10	50
concrete_compressive_strength	25	30	train_residual_models	10	50
cars	100	5	train_residual_models	10	50
energy_efficiency	5	1	0.01	10	50
california_housing	5	10	0.1	5	30
airfoil_self_noise	25	1	train_residual_models	10	50
QSAR_fish_toxicity	50	30	train_residual_models	10	50
MultiMLPs					
space_ga	25	10	train_residual_models	10	50
cpu_activity	5	20	0.001	5	30
naval_propulsion_plant	5	20	0.001	5	50
miami_housing	5	20	train_residual_models	5	50
kin8nm	5	20	train_residual_models	5	50
concrete_compressive_strength	5	30	train_residual_models	10	50
cars	5	30	train_residual_models	5	50
energy_efficiency	50	30	0.1	10	50
california_housing	5	20	0.01	5	30
airfoil_self_noise	25	10	train_residual_models	10	50
QSAR_fish_toxicity	50	30	train_residual_models	5	50

Table 5: Hyper-parameters for MultiXGBs, MultiETs, and MultiMLPs.

Appendix B. Datasets for tabular benchmark

OpenML-CTR23 (Fischer et al., 2023) datasets are selected in the following manner. The first criterion is to exclude datasets exceeding 30,000 instances or 30 features to maintain computational tractability. Moreover, we exclude the *moneyball* data (Kaggle, 2017) to control for missing values and any datasets with non-numeric features, such as those with

temporal or ordinal data not encoded numerically. We then categorize the datasets based on size: small for those with less than ten features, medium for 10 to 19 features, and large for 20 to 29 features. We apply a similar size classification based on the number of instances, considering datasets with more than 10,000 instances as large. To avoid computational constraints, we exclude datasets that were large in both features and instances, ensuring a varied yet manageable set for our experiments. This leads us to the final list of 11 datasets listed in Table 6.

Dataset Name	N Samples	N Features	OpenML Data ID	Y [Min:Max]	Source
space_ga	3,107	7	45402	[-3.06:0.1]	(Pace and Barry, 1997)
cpu_activity	8,192	22	44978	[0:99]	(Rasmussen et al., 1996)
naval_propulsion_plant	11,934	15	44969	[0.95:1.0]	(Coraddu et al., 2016)
miami_housing	13,932	16	44983	[72,000:2,650,000]	(Kaggle, 2022)
kin8nm	8,192	9	44980	[0.04:1.46]	(Ghahramani, 1996)
concrete_compressive_strength	1,030	9	44959	[2.33:82.6]	(Yeh, 1998)
cars	804	18	44994	[8,639:70,756]	(Kuiper, 2008)
energy_efficiency	768	9	44960	[6.01:43.1]	(Tsanas and Xifara, 2012)
california_housing	20,640	9	44977	[14,999:500,001]	(Kelley Pace and Barry, 1997)
airfoil_self_noise	1,503	6	44957	[103.38:140.98]	(Brooks et al., 1989)
QSAR_fish_toxicity	908	7	44970	[0.053:9.612]	(M. Cassotti and Consonni, 2015)

Table 6: Summary of benchmark tabular datasets retained from (Fischer et al., 2023)

Appendix C. Metrics for prediction intervals

C.1. Metrics' definitions

The most common metrics for evaluating prediction intervals (Pearce et al., 2018; Zhou et al., 2023; Gneiting et al., 2007; V'yugin and Trunov, 2019) are:

- Prediction Interval Coverage Probability (PICP): This metric assesses the proportion of times the true value of the target variable falls within the constructed prediction intervals. For a set of test examples $(x_1, y_1), \ldots, (x_N, y_N)$, a given level of confidence α, and their corresponding prediction intervals $I_1, \ldots, I_N$, the PICP is calculated as:

$$\text{PICP} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}(y_i \in [L_i, U_i]), \tag{20}$$

  where U_i and L_i are the upper and lower bounds of the predicted values for the i-th instance. y_i is the actual value of the i-th test example, and $\mathbb{1}$ is the indicator function, which equals 1 if y_i is in the interval $[L_i, U_i]$ and 0 otherwise. $0 \leq \text{PICP} \leq 1$ where PICP closer to 1 and higher than the confidence level α is favored.

- Mean Prediction Interval Width (MPIW): The average width is computed as

$$\text{MPIW} = \frac{1}{N} \sum_{i=1}^{N} (U_i - L_i), \tag{21}$$

which shows the sharpness or uncertainty, where $0 \leq \text{MPIW} < \infty$ and MPIW close to 0 is preferred.

- Normalized Mean Prediction Interval Width (NMPIW): Since MPIW varies by dataset, it can be normalized by the range of the target variable

$$\text{NMPIW} = \frac{\text{MPIW}}{\max(y) - \min(y)} \tag{22}$$

where $\max(y)$ and $\min(y)$ are the maximum and minimum values of the target variable, respectively. The interpretation remains the same as MPIW.

- Continuous Ranked Probability Score (CRPS): We can approximate a full predictive cumulative distribution function by assuming that the forecast is uniformly distributed between its lower and upper bounds. Let L and U denote the lower and upper bounds (corresponding to the $\alpha/2$ and $1-\alpha/2$ quantiles, respectively). Then, for an observation y, we define

$$\text{CRPS}(y, L, U) = \begin{cases} L - y + \frac{U-L}{3}, & y \leq L, \\ \frac{(y-L)^3 + (U-y)^3}{3\,(U-L)^2}, & y \in [L, U], \\ y - U + \frac{U-L}{3}, & y \geq U. \end{cases} \tag{23}$$

This closed-form expression yields a lower CRPS when the forecast is both sharper and better calibrated.

- Quantile (pinball) Loss: For a given quantile level $\tau \in (0, 1)$ and prediction q, the pinball loss is defined as:

$$\ell_\tau(y, q) = (y - q)\,(\tau - \mathbb{1}\{y \leq q\})\,, \tag{24}$$

where $\mathbb{1}\{y \leq q\}$ equals 1 if $y \leq q$ and 0 otherwise. This loss penalizes underestimation and overestimation asymmetrically and is used to tune the quantile predictions.

C.2. Impact of relative metrics for modeling

As our primary focus is on interval prediction, configurations demonstrating the most significant improvements in ΔCWR and ΔPICP are prioritized when selecting the optimal hyper-parameters. Furthermore, both ΔPICP and ΔCWR must be positive, indicating that we must at least have the same reliability of the baseline (PICP) with better or same interval ratios (CWR). In instances where no configuration meets the initial improvement criteria for both metrics, we relax the requirement for positive ΔPICP to accept values greater than -5% and subsequently -10%, allowing us to consider configurations where SEMF significantly improves CWR, even if the PICP improvement is less marked but remains within an acceptable range for drawing comparisons.

Appendix D. Full synthetic experiments results

	Relative Metrics					Absolute Metrics		
Seed	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
Normal								
0	**17%**	**15%**	**6%**	**11%**	0%	1.78	0.54	0.97
10	**11%**	**11%**	**3%**	**5%**	-1%	1.60	0.60	0.96
20	**19%**	**18%**	**8%**	**14%**	-2%	1.89	0.50	0.95
30	**10%**	**10%**	**5%**	**10%**	-1%	1.99	0.48	0.96
40	**35%**	**28%**	**10%**	**17%**	-3%	2.49	0.37	0.93
Uniform								
0	-2%	-5%	-1%	0%	**2%**	2.36	0.42	0.98
10	**34%**	**26%**	**12%**	**26%**	-1%	2.12	0.47	0.99
20	**9%**	**8%**	**5%**	**12%**	**1%**	2.11	0.47	0.99
30	**28%**	**26%**	**9%**	**19%**	-5%	3.59	0.25	0.91
40	**58%**	**39%**	**23%**	**32%**	-3%	3.61	0.26	0.95
Lognormal								
0	**23%**	**19%**	**15%**	**9%**	0%	2.08	0.46	0.96
10	**59%**	**39%**	**18%**	**7%**	-3%	2.07	0.46	0.95
20	**8%**	**9%**	**8%**	**2%**	-1%	1.96	0.48	0.95
30	**9%**	**8%**	**9%**	**5%**	0%	1.59	0.62	0.98
40	**38%**	**28%**	**15%**	**14%**	0%	2.30	0.42	0.97
Gumbel								
0	**4%**	**1%**	**7%**	**16%**	**4%**	1.61	0.60	0.98
10	**17%**	**18%**	**7%**	**4%**	-3%	1.78	0.53	0.94
20	**20%**	**17%**	**14%**	**16%**	-1%	2.01	0.48	0.96
30	-5%	-9%	**1%**	0%	**4%**	1.75	0.55	0.96
40	**15%**	**14%**	**10%**	**5%**	-1%	2.02	0.47	0.95

Table 7: Test results for MultiXGBs, over 5 seeds, with cosine $f(x)$ using 2 predictors, organized by distribution and seed. Relative metrics are in bold when positive.

	Relative Metrics					Absolute Metrics		
Seed	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
Normal								
0	**8%**	**8%**	**4%**	0%	-1%	1.98	0.48	0.96
10	**17%**	**18%**	**6%**	**12%**	-3%	1.66	0.57	0.94
20	**1%**	0%	0%	**2%**	**1%**	1.81	0.54	0.97
30	**15%**	**14%**	**8%**	**16%**	-1%	1.99	0.49	0.97
40	**32%**	**27%**	**8%**	**9%**	-4%	2.46	0.38	0.94
Uniform								
0	**35%**	**28%**	**10%**	**21%**	-3%	2.80	0.35	0.97
10	**3%**	-1%	**3%**	**8%**	**4%**	2.45	0.41	0.99
20	**7%**	**5%**	**4%**	**11%**	**1%**	2.58	0.37	0.96
30	**9%**	**8%**	**6%**	**14%**	0%	3.42	0.28	0.97
40	**29%**	**25%**	**7%**	**12%**	-3%	3.63	0.26	0.96
Lognormal								
0	-14%	-18%	**8%**	-10%	**1%**	1.85	0.52	0.96
10	**18%**	**17%**	**10%**	-2%	-2%	1.87	0.51	0.95
20	**12%**	**12%**	**13%**	0%	-1%	2.03	0.47	0.96
30	-6%	-8%	**5%**	-4%	**2%**	1.74	0.56	0.98
40	**14%**	**13%**	**11%**	**2%**	-2%	2.10	0.46	0.97
Gumbel								
0	-4%	-7%	**9%**	0%	**2%**	1.66	0.59	0.97
10	**8%**	**7%**	**8%**	-2%	0%	1.72	0.56	0.97
20	-2%	-2%	**7%**	**5%**	0%	1.90	0.50	0.96
30	**2%**	**4%**	**6%**	-2%	-2%	1.85	0.51	0.95
40	**14%**	**14%**	**11%**	**5%**	-2%	1.96	0.50	0.97

Table 8: Test results for MultiETs, over 5 seeds, with cosine $f(x)$ using 2 predictors, organized by distribution and seed. Relative metrics are in bold when positive.

	Relative Metrics					Absolute Metrics		
Seed	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
Normal								
0	0%	**1%**	-2%	0%	-1%	1.98	0.48	0.95
10	**9%**	**8%**	**4%**	**5%**	0%	1.71	0.56	0.96
20	-2%	-1%	-1%	-3%	-1%	2.04	0.46	0.94
30	-1%	-2%	**1%**	**3%**	0%	2.20	0.44	0.97
40	**7%**	**7%**	**1%**	**5%**	0%	2.48	0.39	0.96
Uniform								
0	**2%**	**2%**	0%	**8%**	0%	2.98	0.33	0.98
10	**6%**	**2%**	**5%**	**14%**	**3%**	2.80	0.35	0.98
20	**3%**	**7%**	0%	-5%	-4%	3.02	0.31	0.94
30	-2%	-2%	-1%	-4%	0%	3.85	0.25	0.96
40	**13%**	**10%**	**8%**	**23%**	**1%**	3.67	0.27	0.99
Lognormal								
0	**2%**	**3%**	**1%**	**4%**	0%	1.71	0.57	0.98
10	**18%**	**16%**	**7%**	**8%**	0%	1.99	0.48	0.95
20	-4%	-5%	-2%	0%	**1%**	1.86	0.52	0.98
30	**15%**	**13%**	**7%**	**9%**	-1%	1.74	0.56	0.98
40	**4%**	**5%**	**1%**	**2%**	-1%	2.10	0.46	0.97
Gumbel								
0	-3%	-1%	-2%	-4%	-1%	1.54	0.64	0.98
10	**8%**	**8%**	**1%**	**4%**	-1%	1.88	0.51	0.95
20	**8%**	**8%**	**3%**	**4%**	0%	2.14	0.44	0.94
30	**13%**	**14%**	**2%**	-4%	-2%	1.87	0.51	0.95
40	-3%	-5%	-2%	-2%	**2%**	1.76	0.56	0.98

Table 9: Test results for MultiMLPs, over 5 seeds, with cosine $f(x)$ using 2 predictors, organized by distribution and seed. Relative metrics are in bold when positive.

	Relative Metrics					Absolute Metrics		
Seed	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
Normal								
0	**35%**	**26%**	**28%**	**16%**	0%	5.80	0.17	0.96
10	**65%**	**41%**	**34%**	**35%**	-3%	4.80	0.20	0.96
20	**40%**	**29%**	**31%**	**16%**	0%	4.41	0.21	0.94
30	**30%**	**23%**	**26%**	**23%**	0%	5.32	0.18	0.95
40	**28%**	**22%**	**27%**	**28%**	0%	5.83	0.16	0.95
Uniform								
0	**80%**	**44%**	**42%**	**30%**	0%	10.12	0.09	0.94
10	**54%**	**34%**	**47%**	**32%**	**1%**	5.78	0.17	0.98
20	**62%**	**38%**	**40%**	**31%**	0%	5.62	0.17	0.96
30	**62%**	**39%**	**39%**	**16%**	-1%	7.28	0.13	0.95
40	**68%**	**41%**	**39%**	**18%**	-1%	7.93	0.12	0.97
Lognormal								
0	**2%**	**1%**	**20%**	0%	**1%**	4.37	0.22	0.96
10	-2%	-2%	**14%**	-4%	**1%**	2.92	0.34	0.99
20	**7%**	**2%**	**27%**	**6%**	**5%**	3.81	0.25	0.94
30	-13%	-16%	**15%**	**3%**	**1%**	3.28	0.29	0.96
40	**7%**	**6%**	**23%**	**16%**	0%	3.42	0.29	0.98
Gumbel								
0	**34%**	**27%**	**30%**	**20%**	-2%	5.03	0.19	0.96
10	**49%**	**35%**	**24%**	**21%**	-3%	3.70	0.26	0.95
20	0%	0%	**9%**	**3%**	**1%**	2.81	0.35	0.99
30	**11%**	**7%**	**17%**	**14%**	**4%**	4.49	0.21	0.95
40	**18%**	**15%**	**21%**	**20%**	**1%**	4.37	0.22	0.97

Table 10: Test results for MultiXGBs, over 5 seeds, with quadratic $f(x)$ using 2 predictors, organized by distribution and seed. Relative metrics are in bold when positive.

	Relative Metrics					Absolute Metrics		
Seed	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
Normal								
0	**7%**	**7%**	0%	-4%	-1%	5.77	0.16	0.92
10	**35%**	**26%**	**21%**	**23%**	0%	4.66	0.21	0.98
20	**29%**	**24%**	**15%**	**14%**	-2%	4.69	0.20	0.93
30	**26%**	**21%**	**15%**	**13%**	-1%	4.91	0.19	0.95
40	**21%**	**16%**	**18%**	**12%**	**1%**	4.79	0.20	0.98
Uniform								
0	**28%**	**23%**	**16%**	-4%	-1%	8.81	0.11	0.94
10	**25%**	**20%**	**24%**	**21%**	**1%**	4.95	0.20	0.99
20	**47%**	**34%**	**25%**	**7%**	-3%	4.51	0.21	0.96
30	**30%**	**22%**	**22%**	**15%**	**2%**	5.85	0.17	0.98
40	**71%**	**42%**	**33%**	**21%**	0%	7.34	0.13	0.97
Lognormal								
0	-9%	-13%	0%	-6%	**3%**	4.34	0.22	0.96
10	-1%	-2%	**13%**	**12%**	**1%**	3.36	0.29	0.98
20	**33%**	**27%**	**16%**	**10%**	-2%	4.09	0.23	0.95
30	**8%**	**8%**	**11%**	**9%**	0%	3.29	0.30	0.97
40	**10%**	**8%**	**15%**	**9%**	**1%**	3.78	0.26	0.97
Gumbel								
0	**5%**	**4%**	**10%**	**7%**	**1%**	4.67	0.21	0.98
10	**42%**	**30%**	**27%**	**28%**	-1%	4.55	0.21	0.94
20	**33%**	**25%**	**16%**	**15%**	-1%	3.44	0.28	0.97
30	-3%	-4%	**5%**	0%	**1%**	3.86	0.25	0.97
40	**68%**	**42%**	**30%**	**24%**	-2%	3.97	0.24	0.97

Table 11: Test results for MultiETs, over 5 seeds, with quadratic $f(x)$ using 2 predictors, organized by distribution and seed. Relative metrics are in bold when positive.

	Relative Metrics					Absolute Metrics		
Seed	ΔCWR	ΔNMPIW	ΔCRPS	ΔPinball	ΔPICP	CWR	NMPIW	PICP
Normal								
0	-14%	-20%	**3%**	**6%**	**3%**	6.63	0.15	0.97
10	**24%**	**21%**	**13%**	**19%**	-2%	6.07	0.16	0.96
20	0%	-4%	**2%**	**6%**	**3%**	5.81	0.16	0.93
30	-1%	-5%	**7%**	**17%**	**4%**	6.13	0.16	0.95
40	**15%**	**14%**	**9%**	**20%**	0%	7.47	0.13	0.95
Uniform								
0	**42%**	**26%**	**25%**	**36%**	**6%**	15.86	0.06	0.97
10	**27%**	**20%**	**14%**	**30%**	**2%**	9.42	0.11	1.00
20	**66%**	**41%**	**27%**	**29%**	-2%	10.02	0.10	0.95
30	**34%**	**26%**	**13%**	**9%**	0%	12.50	0.08	0.96
40	**15%**	**13%**	**16%**	**40%**	-1%	13.66	0.07	0.97
Lognormal								
0	**1%**	**2%**	**4%**	0%	-1%	5.41	0.18	0.97
10	**3%**	**2%**	**8%**	0%	**1%**	5.23	0.18	0.96
20	**10%**	**11%**	**3%**	-10%	-2%	4.79	0.20	0.95
30	-12%	-15%	**4%**	0%	**1%**	3.86	0.25	0.97
40	**4%**	**3%**	**2%**	**4%**	0%	4.88	0.20	0.97
Gumbel								
0	**38%**	**30%**	**19%**	**18%**	-3%	6.90	0.14	0.96
10	**32%**	**25%**	**23%**	**23%**	-1%	4.56	0.21	0.96
20	-6%	-8%	-3%	-5%	**2%**	4.10	0.23	0.96
30	**13%**	**12%**	**3%**	**8%**	-1%	5.40	0.18	0.96
40	-3%	-5%	**10%**	**10%**	**1%**	5.14	0.19	0.96

Table 12: Test results for MultiMLPs, over 5 seeds, with quadratic $f(x)$ using 2 predictors, organized by distribution and seed. Relative metrics are in bold when positive.

Appendix E. Full conformalized results on benchmark datasets

	Relative Metrics					Absolute Metrics		
Dataset	ΔCWR	ΔPICP	ΔNMPIW	ΔCRPS	ΔPinball	CWR	NMPIW	PICP
space_ga	**9%**	0%	**9%**	**12%**	**10%**	6.28	0.15	0.95
space_ga	**5%**	**2%**	**2%**	**11%**	**13%**	2.95	0.32	0.95
space_ga	**4%**	**1%**	**2%**	**10%**	**9%**	3.23	0.30	0.97
space_ga	**21%**	-2%	**19%**	**11%**	**9%**	3.37	0.28	0.96
space_ga	**16%**	-2%	**15%**	**11%**	**6%**	6.74	0.14	0.94
cpu_activity	**7%**	0%	**7%**	**17%**	**10%**	9.84	0.10	0.96
cpu_activity	-3%	**1%**	-4%	**10%**	0%	8.69	0.11	0.96
cpu_activity	**11%**	-1%	**11%**	**14%**	**10%**	10.20	0.09	0.94
cpu_activity	**11%**	**1%**	**10%**	**26%**	**36%**	10.14	0.09	0.95
cpu_activity	**1%**	**1%**	0%	**13%**	**18%**	9.99	0.10	0.95
naval_propulsion_plant	**200%**	-3%	**68%**	**73%**	**43%**	9.19	0.10	0.92
naval_propulsion_plant	**114%**	0%	**53%**	**66%**	**31%**	7.21	0.13	0.95
naval_propulsion_plant	**159%**	**1%**	**61%**	**74%**	**50%**	7.97	0.12	0.96
naval_propulsion_plant	**187%**	0%	**65%**	**74%**	**50%**	8.73	0.11	0.95
naval_propulsion_plant	**156%**	**2%**	**60%**	**71%**	**50%**	8.24	0.12	0.96
miami_housing	**9%**	-1%	**9%**	**14%**	-5%	9.08	0.10	0.95
miami_housing	-4%	0%	-4%	**8%**	-12%	8.28	0.12	0.96
miami_housing	**2%**	0%	**3%**	**9%**	-19%	8.49	0.11	0.94
miami_housing	**12%**	0%	**10%**	**17%**	-5%	9.23	0.10	0.95
miami_housing	**8%**	-1%	**9%**	**11%**	-21%	8.70	0.11	0.95
kin8nm	**16%**	-1%	**15%**	**22%**	**3%**	2.29	0.41	0.94
kin8nm	**19%**	-1%	**16%**	**25%**	**11%**	2.31	0.41	0.94
kin8nm	**16%**	-1%	**14%**	**20%**	**6%**	2.27	0.42	0.95
kin8nm	**19%**	-1%	**17%**	**22%**	**8%**	2.29	0.41	0.95
kin8nm	**19%**	-1%	**17%**	**24%**	**11%**	2.40	0.40	0.95
concrete_compressive_strength	**31%**	-1%	**24%**	**35%**	**25%**	2.95	0.33	0.96
concrete_compressive_strength	**73%**	-5%	**45%**	**34%**	**14%**	3.60	0.26	0.92
concrete_compressive_strength	**27%**	-3%	**24%**	**29%**	**6%**	3.14	0.29	0.92
concrete_compressive_strength	**47%**	-2%	**34%**	**33%**	**9%**	3.14	0.30	0.94
concrete_compressive_strength	**24%**	-1%	**20%**	**27%**	**4%**	3.25	0.29	0.93
cars	**32%**	-2%	**25%**	**25%**	**6%**	5.78	0.17	0.95
cars	**10%**	-1%	**10%**	**20%**	**18%**	5.02	0.19	0.95
cars	**27%**	0%	**22%**	**26%**	**17%**	5.08	0.19	0.95
cars	**67%**	-3%	**42%**	**42%**	**32%**	6.15	0.15	0.94
cars	**22%**	-2%	**20%**	**23%**	**15%**	4.63	0.21	0.97
energy_efficiency	**114%**	**1%**	**52%**	**66%**	**43%**	15.29	0.06	0.96
energy_efficiency	**112%**	0%	**53%**	**67%**	**50%**	14.18	0.07	0.95
energy_efficiency	**109%**	-2%	**53%**	**64%**	**44%**	10.78	0.09	0.96
energy_efficiency	**230%**	**8%**	**68%**	**81%**	**73%**	17.66	0.05	0.96
energy_efficiency	**72%**	**5%**	**39%**	**59%**	**58%**	11.84	0.08	0.95
california_housing	**3%**	0%	**3%**	**23%**	-3%	2.09	0.46	0.96
california_housing	**4%**	-1%	**4%**	**18%**	-10%	2.27	0.42	0.94
california_housing	**2%**	0%	**1%**	**21%**	-7%	2.29	0.41	0.95
california_housing	0%	0%	0%	**15%**	-9%	2.27	0.41	0.94
california_housing	-1%	0%	-1%	**15%**	-10%	2.28	0.42	0.95
airfoil_self_noise	-5%	0%	-5%	**9%**	-19%	2.60	0.37	0.96
airfoil_self_noise	**6%**	0%	**6%**	**20%**	**4%**	2.56	0.38	0.98
airfoil_self_noise	-32%	0%	-47%	-36%	-43%	1.57	0.63	0.99
airfoil_self_noise	-4%	0%	-5%	**8%**	-10%	2.26	0.43	0.97
airfoil_self_noise	-8%	-1%	-7%	**9%**	-15%	2.43	0.39	0.95
QSAR_fish_toxicity	**2%**	0%	**1%**	**8%**	**10%**	1.80	0.54	0.97
QSAR_fish_toxicity	**11%**	-1%	**10%**	**8%**	**12%**	1.63	0.59	0.96
QSAR_fish_toxicity	**4%**	**2%**	**3%**	**9%**	**8%**	1.80	0.54	0.97
QSAR_fish_toxicity	**26%**	0%	**20%**	**17%**	**18%**	1.55	0.64	0.99
QSAR_fish_toxicity	**53%**	-1%	**35%**	**24%**	**30%**	1.73	0.57	0.98

Table 13: Test results for MultiXGBs with complete data at 95% quantiles for seeds 0, 10, 20, 30, 40, with rows ordered by seed (ascending). Performance over the baseline is highlighted in bold.

	Relative Metrics					Absolute Metrics		
Dataset	ΔCWR	ΔPICP	ΔNMPIW	ΔCRPS	ΔPinball	CWR	NMPIW	PICP
space_ga	**4%**	-2%	**6%**	**6%**	-9%	5.86	0.16	0.95
space_ga	-2%	0%	-2%	**7%**	-3%	2.66	0.36	0.96
space_ga	**10%**	-1%	**10%**	**7%**	-5%	3.19	0.30	0.97
space_ga	-2%	0%	-2%	**7%**	-5%	2.99	0.33	0.97
space_ga	**15%**	-3%	**16%**	**11%**	-9%	7.22	0.13	0.91
cpu_activity	**5%**	-5%	**10%**	**2%**	-22%	8.69	0.11	0.94
cpu_activity	**1%**	-3%	**4%**	**7%**	-11%	8.05	0.12	0.96
cpu_activity	**3%**	-5%	**8%**	**5%**	-11%	8.26	0.11	0.94
cpu_activity	-1%	-4%	**2%**	-2%	-22%	8.14	0.12	0.95
cpu_activity	**3%**	-4%	**7%**	**2%**	-22%	8.11	0.12	0.95
naval_propulsion_plant	**154%**	-6%	**63%**	**69%**	**40%**	3.55	0.27	0.94
naval_propulsion_plant	**153%**	0%	**60%**	**68%**	**46%**	3.95	0.24	0.95
naval_propulsion_plant	**160%**	0%	**61%**	**72%**	**52%**	3.86	0.25	0.96
naval_propulsion_plant	**136%**	0%	**58%**	**66%**	**41%**	3.67	0.26	0.95
naval_propulsion_plant	**118%**	-4%	**56%**	**64%**	**41%**	3.23	0.29	0.95
miami_housing	-13%	**1%**	-16%	-4%	-75%	6.25	0.15	0.96
miami_housing	-6%	0%	-6%	0%	-67%	6.71	0.14	0.95
miami_housing	-5%	**1%**	-6%	0%	-67%	6.47	0.15	0.95
miami_housing	-5%	0%	-4%	-1%	-75%	6.75	0.14	0.95
miami_housing	-15%	**1%**	-19%	-7%	-81%	6.03	0.16	0.95
kin8nm	**8%**	0%	**7%**	**15%**	0%	2.14	0.44	0.95
kin8nm	**7%**	0%	**7%**	**13%**	-3%	2.11	0.45	0.95
kin8nm	**7%**	0%	**7%**	**14%**	0%	2.10	0.46	0.96
kin8nm	**5%**	**1%**	**4%**	**12%**	0%	2.08	0.46	0.95
kin8nm	**10%**	**1%**	**8%**	**17%**	0%	2.22	0.43	0.95
concrete_compressive_strength	**26%**	-3%	**23%**	**12%**	0%	3.27	0.29	0.94
concrete_compressive_strength	**5%**	0%	**5%**	**8%**	0%	2.97	0.33	0.97
concrete_compressive_strength	**13%**	**3%**	**9%**	**12%**	**3%**	3.39	0.28	0.93
concrete_compressive_strength	-14%	**2%**	-20%	**4%**	-12%	2.73	0.35	0.95
concrete_compressive_strength	**3%**	0%	**3%**	**9%**	-4%	2.87	0.34	0.97
cars	**12%**	-1%	**12%**	**3%**	0%	4.95	0.19	0.96
cars	**2%**	-2%	**4%**	-1%	0%	4.97	0.19	0.96
cars	**29%**	-3%	**24%**	**15%**	**15%**	3.64	0.27	0.97
cars	-25%	**4%**	-40%	-8%	-5%	6.19	0.15	0.92
cars	**54%**	-2%	**36%**	**20%**	**28%**	4.70	0.21	0.97
energy_efficiency	**26%**	-3%	**23%**	**12%**	0%	15.20	0.06	0.96
energy_efficiency	**16%**	-3%	**17%**	**6%**	0%	16.14	0.06	0.94
energy_efficiency	**24%**	0%	**20%**	**9%**	**25%**	14.43	0.07	0.99
energy_efficiency	**8%**	0%	**8%**	**4%**	0%	20.70	0.04	0.91
energy_efficiency	**1%**	-1%	**1%**	-15%	-80%	13.19	0.07	0.96
california_housing	-11%	-1%	-11%	**16%**	-26%	1.55	0.62	0.96
california_housing	-1%	-2%	**1%**	**20%**	-24%	1.68	0.57	0.95
california_housing	**2%**	-2%	**4%**	**24%**	-17%	1.70	0.56	0.95
california_housing	**4%**	-3%	**7%**	**20%**	-25%	1.76	0.53	0.93
california_housing	**9%**	-3%	**11%**	**25%**	-17%	1.83	0.52	0.94
airfoil_self_noise	-21%	-1%	-24%	-26%	-40%	2.15	0.45	0.97
airfoil_self_noise	**22%**	0%	**18%**	**19%**	**18%**	2.91	0.34	0.98
airfoil_self_noise	-20%	-2%	-22%	-21%	-36%	2.17	0.44	0.96
airfoil_self_noise	-39%	**1%**	-66%	-52%	-61%	1.52	0.64	0.98
airfoil_self_noise	-6%	-1%	-5%	-3%	-12%	2.49	0.39	0.97
QSAR_fish_toxicity	-13%	0%	-14%	-4%	-10%	1.69	0.57	0.97
QSAR_fish_toxicity	-20%	0%	-26%	-16%	-19%	1.36	0.72	0.98
QSAR_fish_toxicity	**17%**	-4%	**18%**	**6%**	-2%	2.00	0.47	0.94
QSAR_fish_toxicity	**17%**	-2%	**16%**	**4%**	**2%**	1.87	0.52	0.97
QSAR_fish_toxicity	-18%	0%	-22%	-7%	-14%	1.60	0.61	0.97

Table 14: Test results for MultiETs with complete data at 95% quantiles for seeds 0, 10, 20, 30, 40, with rows ordered by seed (ascending). Performance over the baseline is highlighted in bold.

	Relative Metrics					Absolute Metrics		
Dataset	ΔCWR	ΔPICP	ΔNMPIW	ΔCRPS	ΔPinball	CWR	NMPIW	PICP
space_ga	**5%**	-3%	**8%**	0%	-8%	7.12	0.13	0.92
space_ga	-6%	-2%	-4%	-9%	-13%	3.01	0.32	0.95
space_ga	**3%**	0%	**3%**	**2%**	**3%**	3.60	0.27	0.97
space_ga	**25%**	-2%	**21%**	**13%**	**8%**	4.46	0.21	0.93
space_ga	0%	0%	-1%	**3%**	**8%**	7.29	0.13	0.94
cpu_activity	-16%	-1%	-18%	-14%	-10%	8.51	0.11	0.95
cpu_activity	-9%	0%	-11%	-9%	-11%	8.44	0.11	0.96
cpu_activity	-14%	0%	-16%	-11%	-11%	8.43	0.11	0.96
cpu_activity	-16%	**1%**	-20%	-1%	**17%**	8.73	0.11	0.95
cpu_activity	-14%	0%	-16%	-8%	0%	8.75	0.11	0.95
naval_propulsion_plant	**84%**	-1%	**46%**	**46%**	**33%**	15.95	0.06	0.94
naval_propulsion_plant	**26%**	0%	**20%**	**27%**	0%	12.02	0.08	0.95
naval_propulsion_plant	-2%	-1%	-1%	-3%	-25%	13.52	0.07	0.94
naval_propulsion_plant	**8%**	**1%**	**7%**	**14%**	-25%	11.94	0.08	0.96
naval_propulsion_plant	**1%**	0%	**1%**	**2%**	-20%	10.46	0.09	0.95
miami_housing	-19%	-1%	-23%	-13%	-35%	9.00	0.11	0.95
miami_housing	-35%	0%	-56%	-23%	-38%	7.03	0.14	0.96
miami_housing	-13%	0%	-15%	-9%	-33%	9.98	0.09	0.94
miami_housing	-20%	0%	-25%	-9%	-11%	9.56	0.10	0.94
miami_housing	-17%	0%	-21%	-9%	-12%	9.71	0.10	0.95
kin8nm	**17%**	0%	**14%**	**14%**	**15%**	4.50	0.21	0.96
kin8nm	**4%**	**1%**	**3%**	**5%**	**6%**	4.86	0.20	0.95
kin8nm	**18%**	-2%	**16%**	**11%**	**11%**	4.97	0.19	0.93
kin8nm	**2%**	0%	**2%**	**5%**	**6%**	4.59	0.21	0.95
kin8nm	**3%**	0%	**3%**	**5%**	0%	4.80	0.20	0.93
concrete_compressive_strength	**54%**	-4%	**37%**	**26%**	**21%**	3.33	0.29	0.96
concrete_compressive_strength	-5%	-2%	-2%	-2%	-19%	3.15	0.30	0.94
concrete_compressive_strength	**13%**	-3%	**14%**	**8%**	-12%	3.51	0.26	0.92
concrete_compressive_strength	**73%**	-2%	**43%**	**35%**	**31%**	3.47	0.28	0.97
concrete_compressive_strength	**19%**	**2%**	**14%**	**17%**	**15%**	3.67	0.26	0.95
cars	-11%	-2%	-11%	-13%	-29%	5.13	0.18	0.94
cars	-7%	-1%	-7%	-9%	-13%	5.21	0.19	0.97
cars	-18%	0%	-22%	-17%	-19%	4.00	0.24	0.98
cars	**21%**	-5%	**22%**	**5%**	-21%	7.75	0.12	0.92
cars	-44%	**7%**	-89%	-23%	-14%	4.89	0.20	0.99
energy_efficiency	**102%**	**5%**	**48%**	**56%**	**50%**	19.46	0.05	0.96
energy_efficiency	**74%**	-1%	**42%**	**40%**	**40%**	19.62	0.05	0.95
energy_efficiency	**65%**	-3%	**41%**	**36%**	**25%**	18.91	0.05	0.96
energy_efficiency	**46%**	**3%**	**30%**	**33%**	0%	26.91	0.03	0.90
energy_efficiency	**149%**	-6%	**63%**	**42%**	0%	24.65	0.04	0.92
california_housing	-10%	0%	-11%	0%	-3%	1.99	0.48	0.96
california_housing	-14%	0%	-16%	-5%	-13%	2.12	0.45	0.94
california_housing	-11%	0%	-12%	-1%	-9%	2.12	0.45	0.95
california_housing	-7%	-1%	-6%	0%	-9%	2.22	0.42	0.94
california_housing	-12%	0%	-14%	-2%	-9%	2.20	0.43	0.95
airfoil_self_noise	**60%**	-1%	**38%**	**33%**	**30%**	5.45	0.18	0.97
airfoil_self_noise	**55%**	-1%	**36%**	**37%**	**27%**	4.83	0.20	0.97
airfoil_self_noise	**63%**	0%	**39%**	**43%**	**32%**	4.98	0.20	0.98
airfoil_self_noise	**17%**	**1%**	**14%**	**20%**	**23%**	4.84	0.20	0.95
airfoil_self_noise	**16%**	-2%	**15%**	**17%**	**5%**	4.48	0.21	0.95
QSAR_fish_toxicity	**21%**	-3%	**19%**	**4%**	**6%**	1.85	0.52	0.96
QSAR_fish_toxicity	**2%**	**1%**	**1%**	**1%**	**2%**	1.57	0.62	0.98
QSAR_fish_toxicity	-2%	0%	-2%	-3%	**2%**	1.67	0.58	0.97
QSAR_fish_toxicity	**36%**	-3%	**28%**	**13%**	**15%**	2.04	0.47	0.96
QSAR_fish_toxicity	-15%	**1%**	-19%	-9%	-11%	1.43	0.69	0.98

Table 15: Test results for MultiMLPs with complete data at 95% quantiles for seeds 0, 10, 20, 30, 40, with rows ordered by seed (ascending). Performance over the baseline is highlighted in bold.