Instance-Based Uncertainty Estimation for Gradient-Boosted Regression Trees

Anonymous Author(s)
Affiliation
Address
email

Abstract

Gradient-boosted regression trees (GBRTs) are hugely popular for solving tabular regression problems, but provide no estimate of uncertainty. We propose Instance-Based Uncertainty estimation for Gradient-boosted regression trees (IBUG), a simple method for extending any GBRT point predictor to produce probabilistic predictions. IBUG computes a non-parametric distribution around a prediction using the k-nearest training instances, where distance is measured with a tree-ensemble kernel. The runtime of IBUG depends on the number of training examples at each leaf in the ensemble, and can be improved by sampling trees or training instances. Empirically, we find that IBUG achieves similar or better performance than the previous state-of-the-art across 22 benchmark regression datasets. We also find that IBUG can achieve improved probabilistic performance by using different base GBRT models, and can more flexibly model the posterior distribution of a prediction than competing methods. We also find that previous methods suffer from poor probabilistic calibration on some datasets, which can be mitigated using a scalar factor tuned on the validation data.

1 Introduction

Despite the impressive success of deep-learning models on unstructured data (e.g., images, audio, text), gradient-boosted trees [24] remain the preferred choice for *tabular* or *structured* data [50]. In fact, Kaggle CEO Anthony Goldbloom recently described gradient-boosted trees as the most "glaring difference" between what is used on Kaggle and what is "fashionable in academia" [32].

Our focus is on tabular data for regression tasks, which vary widely from financial [1] and retail-product forecasting [41] to weather [27, 26] and clinic-mortality prediction [6]. Gradient-boosted regression trees (GBRTs) are known to make accurate *point predictions* [40] but provide no estimate of the prediction uncertainty, which is desirable for both forecasting practitioners [9, 66] and the explainable AI (XAI) community [23, 67, 2] in general. Recently, Duan et al. [20] and Sprangers et al. [62] introduced NGBoost and PGBM (respectively), new gradient boosting algorithms that provide state-of-the-art probabilistic predictions. However, NGBoost tends to underperform as a point-predictor, and PGBM is limited in the types of distributions it can use to model the output.

We introduce a simple yet effective method for enabling *any* GBRT point-prediction model to produce probabilistic predictions. Our proposed approach, Instance-Based Uncertainty estimation for Gradient-boosted regression trees (*IBUG*), has two key components: 1) We leverage the fact that GBRTs accurately model the conditional mean and use this point prediction as the mean in a probabilistic forecast; and 2) We identify the k training examples with the greatest *affinity* to the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

test instance and use these examples to estimate the uncertainty of the test prediction. We define the affinity between two instances as the number of times both instances appear in the same leaf throughout the ensemble. Thus, our method acts as a wrapper around any given GBRT model.

In experiments on 21 regression benchmark datasets and one synthetic dataset, we demonstrate the effectiveness of IBUG to deliver on par or improved probabilistic performance as compared to existing state-of-the-art methods while maintaining state-of-the-art point-prediction performance. We also show that probabilistic predictions can be improved by applying IBUG to different GBRT models, something that NGBoost and PGBM cannot do. Additionally, IBUG can use the training instances closest to the test example to directly model the output distribution using any parametric *or* non-parametric distribution; again, something NGBoost and PGBM cannot do. Finally, we show that sampling trees dramatically improves runtime efficiency for computing training-example affinities without having a significant detrimental impact on the resulting probabilistic predictions, allowing IBUG to scale to larger datasets.

2 Notation & Background

We assume an instance space $\mathcal{X} \subseteq \mathbb{R}^p$ and target space $\mathcal{Y} \subset \mathbb{R}$. Let $\mathcal{D} \coloneqq \{(x_i, y_i)\}_{i=1}^n$ be a training dataset in which each instance $x_i \in \mathcal{X}$ is a p-dimensional vector $(x_i^j)_{j=1}^p$ and $y_i \in \mathcal{Y}$.

2.1 Gradient-Boosted Regression Trees

Gradient-boosting [24] is a powerful machine-learning algorithm that iteratively adds weak learners to construct a model $f : \mathcal{X} \to \mathbb{R}$ that minimizes some empirical risk $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. The model is defined by a recursive relationship: $f_0(x) = \gamma, \ldots, f_t(x) = f_{t-1}(x) + \eta\, m_t(x)$ in which f_0 is the base learner, γ is an initial estimate, f_t is the model at iteration t, m_t is the weak learner added during iteration t to improve the model, and η is the learning rate.

Gradient-boosted regression trees (GBRTs) typically choose ℓ to be the mean squared error (MSE), γ as $\frac{1}{n}\sum_{i=1}^n y_i$ (mean output of the training instances), and regression trees as weak learners. Each weak learner is typically chosen to approximate the negative gradient [42]: $m_t = \arg\min_m \frac{1}{n}\sum_{i=1}^n (-g_t^i - m(x_i))^2$ in which $g_t^i = \frac{\partial \ell(y_i, \hat{y}_i)}{\partial \hat{y}_i}$ is the functional gradient of the ith training instance at iteration t with respect to $\hat{y}_i = f_{t-1}(x_i)$.

The weak learner at iteration t recursively partitions the instance space into M_t disjoint regions $\{r_t^j\}_{j=1}^{M_t}$. Each region is called a leaf, and the parameter value for leaf j at tree[1] t is typically determined (given a fixed structure) using a one-step Newton-estimation method [34]: $\theta_t^j = -\sum_{i \in I_t^j} w_i g_t^i / (\sum_{i \in I_t^j} w_i h_t^i + \lambda)$ in which $I_t^j = \{(x_i, y_i) \mid x_i \in r_t^j\}_{i=1}^n$ is the instance set of leaf j for tree t, h_t^i is the second derivative of the ith training instance w.r.t. $\hat{y}_i$, and λ is a regularization constant. Thus, the output of m_t can be written as follows: $m_t(x) = \sum_{j=1}^{M_t} \theta_t^j\, \mathbb{1}[x \in r_t^j]$ in which $\mathbb{1}$ is the indicator function. The final GBRT model generates a prediction for a test example x_{te} by summing the values of the leaves x_{te} traverses to across all T iterations: $\hat{y}_{te} = \sum_{t=1}^T m_t(x_{te})$.

2.2 Probabilistic Regression

Our focus is on probabilistic regression—estimating the conditional probability distribution $P(y|x)$ for some target variable y given some input vector $x \in \mathcal{X}$. Unfortunately, traditional GBRT models only output scalar values. Under a squared-error loss function, these scalar values can be interpreted as the conditional mean in a Gaussian distribution with some (unknown) constant variance. However, homoscedasticity is a strong assumption and unknown constant variance has little value in a probabilistic prediction; thus, in order to allow heteroscedasticity, the predicted distribution needs at least two parameters to convey both the magnitude and uncertainty of the prediction [20].[2]

[1] We use the terms *tree* and *iteration* interchangeably.

[2] By uncertainty, we mean aleatoric [31] (a.k.a. data) uncertainty. This work does not attempt to distinguish between aleatoric and epistemic [42] (a.k.a. knowledge) uncertainty [18].

NGBoost is a recent method by Duan et al. [20] that tackles the aforementioned problems by estimating the parameters of a desired distribution using a multi-parameter boosting approach that trains a separate ensemble for each parameter of the distribution. NGBoost employs the natural gradient to be invariant to parameterization, but requires the inversion of many small matrices (each the size of the number of desired parameters) to do so. Empirically, NGBoost generates state-of-the-art probabilistic predictions, but tends to underperform as a point predictor.

Even more recently, Sprangers et al. [62] introduced PGBM, a single model that optimizes for point performance, but is able to generate accurate probabilistic predictions. PGBM treats leaf values as stochastic random variables, using sample statistics to model the mean and variance of each leaf value. PGBM predicts the output mean and variance of a test example using the estimated parameters of each leaf it is assigned to. The predicted mean and variance is then used as parameters in a specified distribution to generate a probabilistic prediction. PGBM has been shown to produce state-of-the-art probabilistic predictions; however, computing the necessary leaf statistics during training can be computationally expensive, especially as the number of leaves in the ensemble increases (see §5.6). Also, since only the mean and variance are predicted for a given test example, PGBM is limited to distributions using only location and scale to model the output.

To address the shortcomings of existing approaches, we introduce a simple method that performs well on both point and probabilistic performance, *and* can more flexibly model the output than either approach. Our method can also be applied to *any* GBRT model, adding additional flexibility.

3 Instance-Based Uncertainty

Instance-based methods such as k-nearest neighbors have been around for decades and have been useful for many different machine learning tasks [48]. However, defining neighbors based on a fixed metric like euclidean distance may lead to suboptimal performance, especially as the dimensionality of the dataset increases. More recently, it has been shown that random forests can be used as an adaptive nearest neighbors method [38] which identifies the most similar examples to a given instance using the learned model structure. This *supervised tree kernel* can more effectively measure the similarity between examples, and has been used for clustering [44] and local linear modeling [8] as well as instance-[11] and feature-based attribution explanations [49], for example.

In this work, we apply the idea of a supervised tree kernel to help model the *uncertainty* of a given GBRT prediction. Our approach, ***I**nstance-**B**ased **U**ncertainty estimation for **G**radient-boosted regression trees* (IBUG), identifies the neighborhood of similar training examples to a test example using the structure of the GBRT, and then uses those instances to generate a probabilistic prediction. IBUG works for *any* GBRT, and can more flexibly model the output than competing methods.

3.1 Estimating Prediction Variance Using High-Affinity Neighbors

Given a GBRT model f and a test example x_{te}, IBUG first models the magnitude of the test prediction using the scalar output of f: $\mu_{\hat{y}_{te}} = f(x_{te})$ since GBRTs already produce accurate point predictions.

To estimate the variance of $\hat{y}_{te}$, IBUG computes the output variance of the k training examples with the largest *affinity* to the test example. We define affinity between two examples simply as the number of times each instance appears in the same leaf across all trees in f. Thus, the affinity of the ith training example x_i to the test example can be written as:

$$A(x_i, x_{te}) = \sum_{i=1}^{T} \mathbb{1}[R_t(x_i) = R_t(x_{te})], \tag{1}$$

in which $R_t(x_i)$ is the leaf x_i is assigned to for tree t. Alg. 1 summarizes the procedure for computing affinity scores for all training examples. This metric clusters similar examples together based on the learned model representation (i.e., the tree structure). Intuitively, if two examples appear in the same leaf in every tree throughout the ensemble, then both examples are predicted in an identical manner. One may also view eq. (1) as an indication of which training examples

most often affect the leaf values x_{te} is assigned to and thus implicitly which examples are likely to have a big effect on the prediction $\hat{y}_{te}$. Once IBUG identifies the k nearest training instances to x_{te} using eq. (1), we then compute the output variance of those instances to obtain $\sigma^2_{\hat{y}_{te}}$. We can then use $\mu_{\hat{y}_{te}}$ and $\sigma^2_{\hat{y}_{te}}$ to generate a probabilistic prediction for x_{te}, which we show in §3.3.

Algorithm 1 IBUG affinity computation

Input: Input instance $x \in \mathcal{X}$, GBRT model f.
procedure COMPUTEAFFINITIES(x, f)
  $A \leftarrow \vec{0}$ ▷ Init. train affinities
  for $t = 1 \dots T$ **do** ▷ Visit each tree
    Get instance set I^l_t for leaf $l = R_t(x)$
    for $i \in I^l_t$ **do** ▷ Increment affinities
      $A_i \leftarrow A_i + 1$
return A

Algorithm 2 IBUG probabilistic prediction

Input: Input $x \in \mathcal{X}$, GBRT model f, k highest-affinity neighbors $A^{(k)}$, min. variance ρ, variance calibration parameters γ and δ, target distribution D^*.
procedure PROBPREDICT(x, f, $A^{(k)}$, ρ, γ, δ, D^*)
  $\mu_{\hat{y}} \leftarrow f(x)$ ▷ GBRT scalar output
  $\sigma^2_{\hat{y}} \leftarrow \max(\sigma^2(A^{(k)}),\ \rho)$ ▷ Ensure $\sigma^2 > 0$
  $\sigma^2_{\hat{y}} \leftarrow \gamma\sigma^2_{\hat{y}} + \delta$ ▷ Var. calibration, eq. (2)
return $D^*(A^{(k)}|\mu_{\hat{y}}, \sigma^2_{\hat{y}})$ ▷ Eq. (3)

3.2 Calibrating Prediction Variance

The k-nearest neighbors generally do a good job of determining the relative uncertainty of different predictions, but on some datasets, the resulting variance is systematically too large or too small. To correct for this, we apply an additional affine transformation before making the prediction:

$$\sigma^2_{\hat{y}_{te}} \leftarrow \gamma\sigma^2_{\hat{y}_{te}} + \delta, \tag{2}$$

where γ and δ are tuned on validation data after k has been selected. Instead of exhaustively searching over all values of γ or δ, we use either the multiplicative factor (tuning γ with $\delta = 0$) or the additive factor (tuning δ with $\gamma = 1$), and choose between them using their performance on validation data.

We find this simple calibration step consistently improves probabilistic performance for not only IBUG, but for PGBM as well, and at a relatively small cost compared to training the model.

3.3 Modeling the Posterior Distribution

Given a continuous distribution D, we can generate a probabilistic prediction using $\mu_{\hat{y}_{te}}$ and $\sigma^2_{\hat{y}_{te}}$ as follows: $\hat{D}_{te} = D(\mu_{\hat{y}_{te}}, \sigma^2_{\hat{y}_{te}})$. Note D is limited to distributions using location and scale (PGBM can only model these types of distributions), and is typically chosen to be the normal distribution [20, 62].

However, IBUG has additional flexibility when modeling the posterior. Once the k training instances with the largest affinity to x_{te} is found—we denote this set $A^{(k)}$—we can use those samples to directly fit D^*, any distribution including those with high-order moments:

$$\hat{D}_{te} = D^*(A^{(k)}|\mu_{\hat{y}_{te}}, \sigma^2_{\hat{y}_{te}}). \tag{3}$$

Eq. (3) is defined to cover the fact that D^* can be fit directly with $A^{(k)}$ using MLE (maximum likelihood estimation) [45], or may also use $\mu_{\hat{y}_{te}}$ or $\sigma^2_{\hat{y}_{te}}$ as fixed parameter values and then use $A^{(k)}$ to fit any other parameters of the distribution. Overall, directly fitting all or some additional parameters in D^*—for example, the shape parameter in a Weibull distribution—is a benefit over PGBM, which must optimize for a suitable shape value using a gridsearch-like approach with extra validation data.

Note that NGBoost can model any parameterized distribution, but must specify this choice before training; in contrast, IBUG can optimize this choice *after* training. Additionally, IBUG may choose D^* to be a non-parametric density estimator such as KDE (kernel density estimation) [60], something neither PGBM or NGBoost can do.

In summary, Alg. 2 provides pseudocode for generating a probabilistic prediction with IBUG. Note Algs. 1 and 2 work for *any* GBRT model, allowing practitioners to employ IBUG to adapt multiple different point predictors into probabilistic estimators and select the model with the best performance. Empirically, we show using different base models for IBUG can result in improved probabilistic performance than using just one (§5.3).

4 Computational Efficiency

Training efficiency. Since IBUG works with standard GBRT models, it inherits the training efficiency of modern GBRT implementations such as XGBoost [12], LightGBM [34], and CatBoost [50].

Prediction efficiency. If there are T trees in the ensemble and each leaf has at most n_l training instances assigned to it, then IBUG's prediction time is $O(Tn_l)$, since it considers each instance in each leaf. Note training instances that do not appear in a leaf with the test instance x_{te} do not increase prediction time; what matters most is thus the number of instances at each leaf. We find LightGBM often induces regression trees with large leaves—in some cases, over half the dataset is assigned to a single leaf. Thus, prediction time still grows with the size of the dataset, as is typical for instance-based methods. This higher prediction time is the price IBUG pays for greater flexibility.

Prediction efficiency can be increased at training time by using deeper GBRTs with fewer instances in each leaf, after training by subsampling the instances considered for predictions, or at prediction time by sampling the trees used to compute affinities. We explore this last option in the next subsection.

4.1 Sampling Trees

The most expensive operation when generating a probabilistic prediction for x_{te} with IBUG is computing the affinity vector (eq. 1). In order to increase prediction efficiency, we can instead approximate the affinity vector by randomly sampling $\tau < T$ trees without replacement:

$$\tilde{A}(x_i, x_{te}) = \mathbb{E}_{t \sim P(T)} \left[\mathbb{1}[R_t(x_i) = R_t(x_{te})] \right] \tag{4}$$

in which $P(T)$ is the uniform distribution over all T trees. The runtime complexity of $\tilde{A}$ is thus $O(\tau n_l)$, and can provide significant speedups when $\tau \ll T$. Selecting $\tau < T$ introduces a tradeoff between computational complexity and predictive accuracy. However, in our empirical evaluation, we find that sampling a relatively small fraction of the total number of trees can significantly increase prediction efficiency while maintaining accurate probabilistic predictions (§5.6).

5 Experiments

In this section, we demonstrate IBUG's ability to produce competitive probabilistic and point predictions as compared to current state-of-the-art methods on a large set of regression datasets (§5.1, §5.2). Then, we show that IBUG can use different base models to improve probabilistic performance (§5.3), flexibly model the posterior distribution (§5.4), and use approximations to speed up predictions (§5.6).

Implementation and Reproducibility. We implement IBUG in Python, using Cython—a Python package allowing the development of C extensions—to store a unified representation of the model structure. IBUG currently supports all modern gradient boosting frameworks including XGBoost [12], LightGBM [34], and CatBoost [50]. Experiments are run on publicly available datasets using an Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.6GHz with 60GB of RAM @ 2.4GHz. Links to all data sources as well as the code for IBUG and all experiments is currently available online anonymously.[3]

5.1 Methodology

We now compare IBUG's probabilistic and point prediction to NGBoost [20] and PGBM [62] on 21 benchmark regression datasets and one synthetic dataset. Additional dataset details are in §B.1.

Metrics. We compute the average negative log likelihood (NLL) and continuous ranked probability score (CRPS) [27, 73] over the test set to evaluate probabilistic performance. To evaluate point performance, we use root mean squared error (RMSE). See the Appendix §B for detailed descriptions.

[3] `https://anonymous.4open.science/r/ibug-6DDF`.

Table 1: Probabilistic (NLL, CRPS) and point (RMSE) performance for each method on each dataset. Lower is better. Normal distributions are used for all probabilistic predictions. Results are averaged over 20 folds. Standard errors are omitted for brevity but are in the Appendix §B.3. The best method for each dataset is bolded, as well as those with standard errors that overlap the best method. *Bottom row*: Head-to-head comparison between IBUG and each method showing IBUG's total significant wins, ties and losses across all datasets. Overall, IBUG performs competitively with NGBoost and PGBM on both probabilistic and point performance.

	Test NLL (↓)			Test CRPS (↓)			Test RMSE (↓)		
Dataset	NGBst	PGBM	IBUG	NGBst	PGBM	IBUG	NGBst	PGBM	IBUG
Ames	11.29	11.29	**11.23**	37638	**10642**	**10449**	24342	**22763**	**22349**
Bike	1.921	1.819	**1.310**	12.481	1.155	**0.829**	4.139	3.741	**2.379**
California	0.529	0.572	**0.437**	0.256	0.219	**0.210**	0.498	0.440	**0.436**
Communities	**-0.715**	**-0.686**	**-0.659**	0.066	0.068	**0.064**	**0.135**	**0.134**	**0.132**
Concrete	3.033	**2.730**	2.866	3.266	**1.895**	**1.948**	5.456	**3.725**	**3.751**
Energy	0.650	**0.481**	0.861	0.528	0.163	**0.155**	0.468	0.329	**0.306**
Facebook	2.099	3.159	**2.066**	4.132	3.626	**3.059**	**20.75**	**20.33**	**20.39**
Kin8nm	-0.411	**-0.729**	**-0.753**	0.095	0.071	**0.060**	0.177	0.108	**0.103**
Life	2.147	1.961	**1.842**	1.408	0.819	**0.782**	2.261	**1.703**	**1.678**
MEPS	**3.775**	3.870	3.962	**5.550**	6.243	6.342	**23.73**	**23.88**	**23.85**
MSD	3.453	3.571	**3.407**	4.523	4.575	**4.369**	9.120	8.806	**8.778**
Naval	-5.405	-4.706	**-6.281**	0.003	0.000	**0.000**	0.002	**0.000**	**0.000**
News	10.82	**10.63**	**10.63**	**2150**	2328	2432	**11159**	**11047**	**11051**
Obesity	-0.706	-0.532	**-0.981**	3e17	0.113	**0.058**	0.174	0.181	**0.160**
Power	2.773	**2.520**	2.556	2.112	**1.534**	**1.528**	3.891	**2.979**	**2.950**
Protein	2.840	2.682	**2.598**	2.672	1.853	**1.692**	4.526	3.480	**3.420**
STAR	6.873	6.871	**6.854**	**131.50**	**131.00**	**130.30**	233.6	**230.7**	**230.3**
Superconductor	1.495	0.080	**-0.624**	2.429	0.119	**0.087**	**0.143**	0.412	0.357
Synthetic	**3.748**	**3.747**	**3.744**	5.838	**5.774**	**5.771**	**10.23**	**10.21**	**10.21**
Wave	10.71	**10.26**	10.68	570789	**3912**	4265	13505	7901	**7493**
Wine	1.032	0.933	**0.904**	0.385	0.324	**0.320**	0.695	0.607	**0.601**
Yacht	0.540	**0.248**	**0.175**	0.967	**0.240**	0.300	**0.699**	**0.695**	0.899
IBUG W-T-L	17-3-2	13-6-3	-	18-2-2	9-10-3	-	16-4-2	10-11-1	-

Protocol. We follow the same protocol as Sprangers et al. [62] and Duan et al. [20]. First, we create 20 random folds for each dataset. For each fold, we randomly sample 10% of the data to use as a test set. The remaining 90% is randomly split into an 80/20 train/validation set to tune any hyperparameters. Once the hyperparameters are tuned, the model is retrained using the entire 90% training set. For probabilistic predictions, a normal distribution is used to model the output.

Significance testing. To determine which of two methods performs better on a given dataset under a given metric, we use a two-tailed paired t-test over the 20 random folds with a significance level of $p < 0.05$. We also report counts of the number of datasets in which a given method performed significantly better ("Win"), worse ("Loss"), or not different ("Tie") relative to a comparator.

Hyperparameters. We tune NGBoost the same way as in Duan et al. [20]. Since PGBM and IBUG both optimize a point prediction metric, we tune their hyperparameters similarly. We also tune variance calibration parameters γ and δ for each method (§3.2). Exact values evaluated and selected are in §B.2. Unless specified otherwise, we use LightGBM [34] as the base model for IBUG.

5.2 Probabilistic and Point Prediction

We first compare IBUG's probabilistic and point predictions to each baseline on each dataset; see Table 1 for detailed results. Our main findings are as follows:

- On probabilistic performance, IBUG performs equally well or better than NGBoost and PGBM. For NLL, IBUG wins against NGBoost and PGBM on 17 and 13 (out of 22) datasets

Table 2: Probabilistic (NLL, CRPS) performance on the test set for IBUG using different base models. Results are averaged over 20 folds, and standard errors are shown in subscripted parentheses; lower is better. On 7 and 10 datasets, respectively, either IBUG-XGBoost or IBUG-CatBoost outperforms IBUG-LightGBM on the validation set and subsequently on the test set, demonstrating the potential for improved probabilistic performance by using IBUG with different base models.

	Test NLL (↓)		
Dataset	XGBoost	CatBoost	LightGBM
Ames	**11.21**$_{(0.02)}$	**11.21**$_{(0.01)}$	11.23$_{(0.02)}$
Concrete	2.838$_{(0.041)}$	**2.739**$_{(0.055)}$	2.866$_{(0.034)}$
Kin8nm	-0.682$_{(0.006)}$	**-0.845**$_{(0.004)}$	-0.753$_{(0.005)}$
MEPS	**3.829**$_{(0.061)}$	**3.928**$_{(0.106)}$	3.962$_{(0.088)}$
MSD	**3.399**$_{(0.001)}$	3.414$_{(0.001)}$	3.407$_{(0.001)}$
Obesity	**-1.110**$_{(0.019)}$	-0.749$_{(0.020)}$	-0.981$_{(0.012)}$
Wave	10.52$_{(0.01)}$	**10.29**$_{(0.02)}$	10.68$_{(0.01)}$

	Test CRPS (↓)		
Dataset	XGBoost	CatBoost	LightGBM
Ames	10260$_{(142)}$	**9954**$_{(124)}$	10449$_{(138)}$
Concrete	2.010$_{(0.066)}$	**1.736**$_{(0.073)}$	1.948$_{(0.063)}$
Energy	0.157$_{(0.005)}$	**0.142**$_{(0.005)}$	0.155$_{(0.005)}$
Kin8nm	0.065$_{(0.000)}$	**0.052**$_{(0.000)}$	0.060$_{(0.000)}$
MEPS	**6.144**$_{(0.100)}$	**6.158**$_{(0.109)}$	6.342$_{(0.152)}$
Naval	0.000$_{(0.000)}$	**0.000**$_{(0.000)}$	0.000$_{(0.000)}$
Obesity	**0.045**$_{(0.001)}$	0.058$_{(0.001)}$	0.058$_{(0.000)}$
Power	**1.513**$_{(0.011)}$	1.535$_{(0.013)}$	1.528$_{(0.013)}$
Supercond.	**0.009**$_{(0.002)}$	0.151$_{(0.003)}$	0.087$_{(0.002)}$
Yacht	0.231$_{(0.015)}$	**0.191**$_{(0.016)}$	0.300$_{(0.023)}$

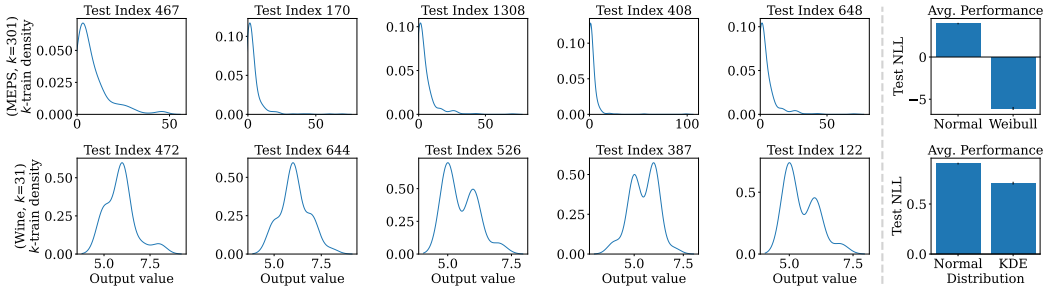


Figure 1: *Left*: Distribution of the k-nearest training instances for 5 randomly-selected test instances from the MEPS and Wine datasets. *Right*: Test NLL when modeling the posterior using two different distributions (lower is better); error bars represent standard error. IBUG can model parametric *and* non-parametric distributions that better fit the underlying data than assuming normality.

respectively, while losing on only 2 and 3. For CRPS, IBUG wins on 18 and 9 datasets, respectively, while again losing on only 2 and 3.

- On point performance, IBUG and PGBM performed significantly better than NGBoost; this is consistent with previous work and is perhaps unsurprising since NGBoost is optimized for probabilistic performance, not point performance. However, IBUG generally performed better than PGBM, winning on 10 datasets and losing on only 1 dataset.

5.3 Different Base Models

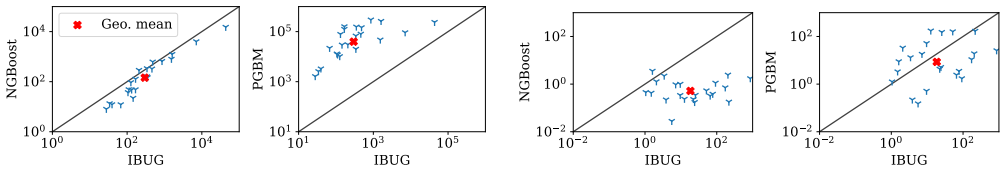
Here we experiment using different base models for IBUG besides LightGBM [34]; specifically, we use XGBoost [12] and CatBoost [50], two popular gradient boosting frameworks. Table 2 shows that using a different base model can result in improved probabilistic performance. This highlights IBUG's agnosticism to GBRT types, enabling practitioners to be apply IBUG to future models with improved point prediction performance. We show IBUG can even improve probabilistic performance for PGBM by using PGBM as the base model for IBUG (see §C.1 for details).

5.4 Posterior Modeling

One of the unique benefits of IBUG is the ability to directly model the output using empirical samples (Figure 1), giving practitioners a better sense of the output distribution for specific predictions. IBUG can optimize a distribution *after* training, and has more flexibility in the types of distributions it can model—from distributions using just location and scale to those with high-order moments

Table 3: Probabilistic performance comparison of each method with vs. without variance calibration. In most cases, calibration maintains or improves performance; it is especially helpful for PGBM.

	NLL			CRPS		
Method	Wins	Ties	Losses	Wins	Ties	Losses
NGBoost	5	15	2	12	10	0
PGBM	14	8	0	17	5	0
IBUG	12	10	0	11	10	1



(a) Total train+tune time (in seconds). (b) Avg. predict time (in seconds) per test example.

Figure 2: Runtime comparison of IBUG against NGBoost and PGBM. *Left*: Total train time (including tuning). *Right*: Average prediction time per test example. Results are shown for each dataset, averaged over 20 folds (exact values are in §B.4, Table 10). Points *above* the diagonal black line means IBUG is faster than the competing method on those datasets, and slower if below. The red "x" represents the geometric mean across all datasets for each method. On average, IBUG has comparable training times to NGBoost but is over two orders of magnitude faster than PGBM. However, IBUG is roughly an order of magnitude slower than NGBoost in terms of prediction time.

as well as non-parametric density estimators. To test this flexibility, we model each probabilistic prediction using the following distributions: normal, skewnormal, lognormal, Laplace, student t, logistic, Gumbel, Weibull, and KDE; we then select the distribution with the best average NLL on the validation set, and evaluate its probabilistic performance on the test set.

Figure 1 demonstrates the selected distributions for the MEPS and Wine datasets achieve better probabilistic performance than assuming a normal distribution. Qualitatively, the empirical densities of $A^{(k)}$ for a randomly sampled set of test instances reaffirms the selected distributions. Additionally, on the MEPS dataset, we note the selected Weibull distribution takes a shape parameter, which IBUG estimates directly on a *per prediction* basis using $A^{(k)}$ and MLE. In contrast, PGBM would have to optimize for a single value over a validation set, which may be suboptimal for individual predictions.

5.5 Variance Calibration

Table 3 shows probabilistic performance comparisons of each method against itself with and without variance calibration. In the majority of cases, variance calibration (§3.2) either maintains or improves performance for all methods, especially PGBM. Overall, these results suggest that variance calibration should be a standard procedure for probabilistic prediction, unless using a method that has particularly well-calibrated predictions to begin with. We therefore use variance calibration in all of our results.

Additionally, §C.2 shows performance results for all methods *without* variance calibration. Overall, we observe similar trends as when applying calibration (Table 1); the W-T-L ratio for IBUG vs. NGBoost and PGBM is 12-5-5 and 15-6-1 (respectively) for NLL and 18-2-2 and 14-4-4 for CRPS.

5.6 Sampling Trees

Table 2 shows the runtime for each method broken down into total training time (including tuning) and prediction time per test example. On average, IBUG has similar training times to NGBoost, but is orders of magnitude faster than PGBM. For predictions, IBUG is significantly slower than NGBoost.

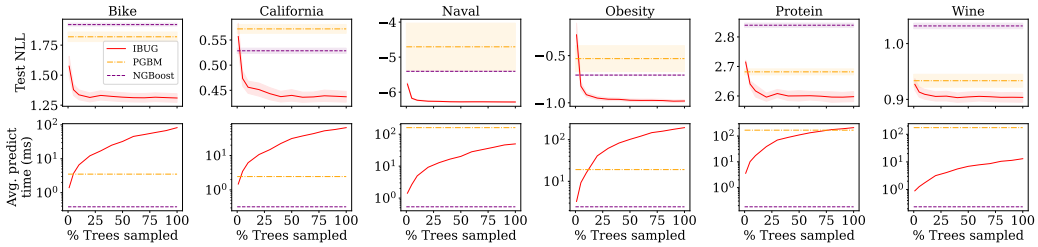


Figure 3: Change in probabilistic (NLL) performance (top row) and average prediction time (in seconds) per test example (bottom row) as τ increases on six datasets; lower is better. PGBM and NGBoost are added for additional context. The shaded regions represent the standard error. In general, average prediction time decreases significantly as τ decreases while test NLL remains relatively stable until roughly 5–20%, at which point probabilistic performance starts to degrade more rapidly; thus, τ introduces a tradeoff between predictive accuracy and efficiency.

However, by sampling $\tau < T$ trees when computing the affinity vector, IBUG can significantly reduce prediction time (Figure 3). We observe as τ decreases, average prediction time decreases significantly while probabilistic performance remains relatively stable until τ/T reaches roughly 5–20%, at which point probabilistic performance starts to decrease more rapidly. Note for the Bike, California, Obesity, and Protein datasets, IBUG can reach the same average prediction time as PGBM while maintaining better probabilistic performance than both NGBoost and PGBM. These results demonstrate that if speed is a concern, IBUG can approximate the affinity computation to speed up prediction times while maintaining competitive probabilistic performance.

6 Related Work

More traditional approaches to probabilistic regression include generalized additive models for location, scale, and shape (GAMLSS), which allow for a flexible choice of distribution for the target variable but are restricted to a pre-specified model form [54]. Prophet [66] also produces probabilistic estimates for generalized additive models, but has been shown to underperform as compared to more recent approaches [58, 3]. Bayesian methods [46, 28] naturally generate uncertainty estimates by integrating over the posterior; but, exact solutions are limited to simple models, and more complex models such as Bayesian Additive Regression Trees (BART) [13, 39] require computationally expensive sampling techniques (e.g., MCMC [4]) to provide approximate solutions.

Other approaches to probabilistic regression tasks include conformal predictions [59, 65, 5] which produce confidence intervals via empirical errors obtained in the past, and quantile regression [30, 35, 43, 55]. Similar to PGBM, distributional forests (DFs) [57] estimate distributional parameters in each leaf, and average these estimates over all trees in the forest. Deep learning approaches for probabilistic regression [52, 71, 3] have increased recently, with notable approaches such as DeepAR [56] and methods based on transformer architectures [36, 37].

7 Conclusion

IBUG uses ideas from instance-based learning to enable probabilistic predictions for *any* GBRT point predictor. IBUG generates probabilistic predictions by using the k-nearest training instances to the test instance found using the structure of the trees in the ensemble. Our results on 22 regression datasets demonstrate this simple wrapper produces competitive probabilistic and point predictions to current state-of-the-art methods, most notably NGBoost [20] and PGBM [62]. We also show that IBUG can more flexibly model the posterior distribution of a prediction using any parametric *or* non-parametric density estimator. IBUG is also fast to train compared to competing methods. IBUG's one limitation is relatively slow prediction time. However, we show that approximations in the search for the k-nearest training instances can significantly speed up prediction time.

References

[1] Yaser S Abu-Mostafa and Amir F Atiya. Introduction to financial forecasting. *Applied Intelligence*, 6(3):205–213, 1996.

[2] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[3] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C Maddix, Syama Sundar Rangapuram, David Salinas, Jasper Schulz, et al. GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[4] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1):5–43, 2003.

[5] Anastasios N Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

[6] Anand Avati, Kenneth Jung, Stephanie Harman, Lance Downing, Andrew Ng, and Nigam H Shah. Improving palliative care with deep learning. *BMC Medical Informatics and Decision Making*, 18(4):55–64, 2018.

[7] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[8] Adam Bloniarz, Ameet Talwalkar, Bin Yu, and Christopher Wu. Supervised neighborhoods for distributed nonparametric regression. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, pages 1450–1459, 2016.

[9] Joos-Hendrik Böse, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Dustin Lange, David Salinas, Sebastian Schelter, Matthias Seeger, and Yuyang Wang. Probabilistic demand forecasting at scale. In *Proceedings of the VLDB Endowment*, volume 10, pages 1694–1705. VLDB Endowment, 2017.

[10] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[11] Jonathan Brophy and Daniel Lowd. Trex: Tree-ensemble representer-point explanations. In *ICML Workshop on Extending Explainable AI*, 2020.

[12] 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*, 2016.

[13] Hugh A Chipman, Edward I George, and Robert E McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[14] Joel W Cohen, Steven B Cohen, and Jessica S Banthin. The medical expenditure panel survey: A national information resource to support healthcare cost research and inform policy and practice. *Medical Care*, pages S44–S50, 2009.

[15] 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. *Journal of Engineering for the Maritime Environment, Part M*, 230(1): 136–153, 2016.

[16] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[17] Dean De Cock. Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3), 2011.

[18] Daniel D'souza, Zach Nussbaum, Chirag Agarwal, and Sara Hooker. A tale of two long tails. *arXiv preprint arXiv:2107.13098*, 2021.

[19] Dheeru Dua and Casey Graff. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2019. [Online; accessed 12-September-2021].

[20] Tony Duan, Avati Anand, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Ng, and Alejandro Schuler. Ngboost: Natural gradient boosting for probabilistic prediction. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2690–2700. PMLR, 2020.

[21] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013.

[22] Kelwin Fernandes, Pedro Vinagre, and Paulo Cortez. A proactive intelligent decision support system for predicting the popularity of online news. In *Proceedings of the 17th Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer, 2015.

[23] Juliana J Ferreira and Mateus S Monteiro. What are people doing about XAI user experience? A survey on AI explainability research and practice. In *Proceedings of the 22nd International Conference on Human-Computer Interaction*, pages 56–73. Springer, 2020.

[24] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[25] Jerome H Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, pages 1–67, 1991.

[26] Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

[27] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[28] Alex Graves. Practical variational inference for neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, volume 24, 2011.

[29] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

[30] Hilaf Hasson, Bernie Wang, Tim Januschowski, and Jan Gasthaus. Probabilistic forecasting: A level-set approach. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, volume 34, 2021.

[31] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

[32] Tim Januschowski, Yuyang Wang, Kari Torkkola, Timo Erkkilä, Hilaf Hasson, and Jan Gasthaus. Forecasting with trees. *International Journal of Forecasting*, 2021.

[33] Heysem Kaya, Pmar Tüfekci, and Fikret S Gürgen. Local and global learning methods for predicting power of a combined gas & steam turbine. In *Proceedings of the 2nd International Conference on Emerging Trends in Computer and Electronics Engineering (ICETCEE)*, pages 13–18, 2012.

[34] Guolin Ke, Qi Meng, et al. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[35] Roger Koenker and Kevin F Hallock. Quantile regression. *Journal of Economic Perspectives*, 15(4):143–156, 2001.

[36] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, pages 5243–5253, 2019.

[37] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 2021.

[38] Yi Lin and Yongho Jeon. Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590, 2006.

[39] Zhao Tang Luo, Huiyan Sang, and Bani Mallick. BAST: Bayesian additive regression spanning trees for complex constrained domain. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, volume 34, 2021.

[40] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Zhi Chen, Anil Gaba, Ilia Tsetlin, and Robert L Winkler. The M5 uncertainty competition: Results, findings and conclusions. *International Journal of Forecasting*, 2021.

[41] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 2022.

[42] Andrey Malinin, Liudmila Prokhorenkova, and Aleksei Ustimenko. Uncertainty in gradient boosting via ensembles. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

[43] Nicolai Meinshausen and Greg Ridgeway. Quantile regression forests. *Journal of Machine Learning Research*, 7(6), 2006.

[44] Frank Moosmann, Bill Triggs, and Frederic Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 985–992, 2006.

[45] In Jae Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90–100, 2003.

[46] Radford M Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996.

[47] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[48] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[49] Gregory Plumb, Denali Molitor, and Ameet S Talwalkar. Model agnostic supervised local explanations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 2515–2524, 2018.

[50] Liudmila Prokhorenkova, Gleb Gusev, et al. CatBoost: Unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

[51] Kumar Rajarshi. Life expectancy (WHO). `https://www.kaggle.com/kumarajarshi/life-expectancy-who?ref=hackernoon.com&select=Life+Expectancy+Data.csv`, 2017. [Online; accessed 12-September-2021].

[52] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, volume 31, pages 7785–7794, 2018.

[53] Michael Redmond and Alok Baveja. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3):660–678, 2002.

[54] Robert A Rigby and D Mikis Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554, 2005.

[55] Yaniv Romano, Evan Patterson, and Emmanuel Candes. Conformalized quantile regression. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, volume 32, pages 3543–3553, 2019.

[56] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020.

[57] Lisa Schlosser, Torsten Hothorn, Reto Stauffer, and Achim Zeileis. Distributional regression forests for probabilistic precipitation forecasting in complex terrain. *The Annals of Applied Statistics*, 13(3):1564–1589, 2019.

[58] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 4837–4846, 2019.

[59] Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3), 2008.

[60] Simon J Sheather and Michael C Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690, 1991.

[61] Kamaljot Singh, Ranjeet Kaur Sandhu, and Dinesh Kumar. Comment volume prediction using neural networks and decision trees. In *IEEE UKSim-AMSS 17th International Conference on Computational Modeling and Simulation, UKSim2015*, March 2015.

[62] Olivier Sprangers, Sebastian Schelter, and Maarten de Rijke. Probabilistic gradient boosting machines for large-scale probabilistic regression. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[63] James H Stock, Mark W Watson, et al. *Introduction to Econometrics*, volume 3. Pearson New York, 2012.

[64] Suzanne. CDC data: Nutrition, physical activity, & obesity. `https://www.kaggle.com/spittman1248/cdc-data-nutrition-physical-activity-obesity`, 2018. [Online; accessed 12-September-2021].

[65] Souhaib Ben Taieb, Raphael Huser, Rob Hyndman, Marc Genton, et al. Probabilistic time series forecasting with boosted additive models: An application to smart meter data. *Department of Economics and Business Statistics, Monash University*, 2015.

[66] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1): 37–45, 2018.

[67] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[68] 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.

[69] Pınar Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140, 2014.

[70] Jan van Rijn. Kin8nm. `https://www.openml.org/d/189`, 2014. [Online; accessed 20-January-2022].

[71] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6607–6617. PMLR, 2019.

[72] I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.

[73] Michaël Zamo and Philippe Naveau. Estimation of the continuous ranked probability score with limited information and applications to ensemble weather forecasts. *Mathematical Geosciences*, 50(2):209–234, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See §7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See §A.1.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See §5.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See §5.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See §5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See §B.1.
   (b) Did you mention the license of the assets? [No] The license of each asset can be found by following its corresponding citation.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See §5.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]