Multivariate prediction intervals for bagged models

IOP Publishing Mach. Learn.: Sci. Technol. 4 (2023) 015022 https://doi.org/10.1088/2632-2153/acb9d5

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
20 September 2022

REVISED
26 January 2023

ACCEPTED FOR PUBLICATION
7 February 2023

PUBLISHED
17 February 2023

PAPER

Multivariate prediction intervals for bagged models

Brendan Folie[1,*] **and Maxwell Hutchinson**[2,3]

[1] Citrine Informatics, Redwood City, CA, United States of America
[2] Kiva Microfunds, San Francisco, CA, United States of America
[3] Maxwell Hutchinson performed the work while at Citrine Informatics
[*] Author to whom any correspondence should be addressed.

E-mail: bfolie@citrine.io

Keywords: materials informatics, uncertainty quantification, sequential learning, random forest

Abstract

Accurate uncertainty estimates can significantly improve the performance of iterative design of experiments, as in sequential and reinforcement learning. For many such problems in engineering and the physical sciences, the design task depends on multiple correlated model outputs as objectives and/or constraints. To better solve these problems, we propose a recalibrated bootstrap method to generate multivariate prediction intervals for bagged models such as random forest and show that it is well-calibrated. We apply the recalibrated bootstrap to a simulated sequential learning problem with multiple objectives and show that it leads to a marked decrease in the number of iterations required to find a satisfactory candidate. This indicates that the recalibrated bootstrap could be a valuable tool for practitioners using machine learning to optimize systems with multiple competing targets.

1. Introduction

One drawback of many regression algorithms is that they do not naturally admit an uncertainty estimate. It is often important to know not just what a model predicts but how confident it is in that prediction. For situations in which a machine learning model is making a single decision, especially one of great import to humans, such as who gets a mortgage or who is granted parole (Kuchibhotla and Berk 2023), an overly-confident model can be disastrous. Accurate uncertainty estimates are also crucial when applying machine learning to the physical sciences. For example, a practitioner may be using a model to identify a material that achieves outstanding performance in some application (Meredig *et al* 2018). Many models are unable to predict values substantially outside the range of the training data, so a direct prediction of such performance is unlikely. But with a well-calibrated uncertainty estimate we may be able to identify candidates that have a reasonable probability of achieving high performance. For this reason uncertainty estimates are an integral part of machine learning based approaches to experimental design, such as Bayesian Optimization, active and reinforcement learning, and sequential learning (SL) (Ling *et al* 2017). We are especially concerned with multi-objective SL, where the optimization goal is to find a candidate $\vec{x}$ that meets or exceeds several ambitious design targets $\vec{y}(\vec{x}) \geqslant \vec{b}$ for an expensive-to-evaluate function $\vec{y}$. SL has been used to identify novel organic LED materials (Antono *et al* 2020, Abroshan *et al* 2021), battery materials (Dave *et al* 2020, Verduzco *et al* 2021), charging protocols (Attia *et al* 2020), colloidal nanoparticles (Fong *et al* 2021) and perovskite photovoltaics (Liu *et al* 2022). More broadly, machine learning-guided experimental design has been used in an astounding variety of fields, including structural engineering (Mathern *et al* 2021, Zhang *et al* 2021), public health (Chandak *et al* 2020, Awal *et al* 2021), and transportation (Liu *et al* 2021, Fakhrmoosavi *et al* 2022).

Many real-world problems involve multiple objectives that are strongly related to each other. For example, a scientist may seek to create a new thermoelectric material that efficiently converts waste heat into electricity. This requires high electrical conductivity and low thermal conductivity, but increasing one tends to increase the other. Or a rover may seek to plan a trajectory through space that maximizes the information it collects over the trajectory while minimizing the length of the trajectory (Wang *et al* 2018). In order to

reliably identify promising candidates, a model should be able to produce multivariate prediction intervals that quantify output correlations.

Here we propose a *multi-output recalibrated bootstrap* method to generate multivariate prediction intervals for bagged models by rescaling the bootstrap standard deviation based on the out-of-bag (OOB) errors. This method is fast, easy to implement, and works even with small numbers of bags and training rows. We study the accuracy of the recalibrated bootstrap on a variety of synthetic and real-world test problems, showing that it is well-calibrated in both the single- and multi-output cases. Finally we test the recalibrated bootstrap on a multi-output SL scenario, in which the algorithm attempts to identify a candidate that will simultaneously satisfy several competing objectives. We find a marked decrease in the number of iterations required, which has significant implications for real-world SL.

2. Related work

Several early and influential studies of uncertainty in random forests focused on generating a confidence interval for the predicted value. Wager *et al* (2014) studied two generalizations of the jackknife, the infinitesimal jackknife (IJ) and the jackknife after bootstrap (JaB), and introduced a bias-correction term that decreases the number of bags needed for the calculation to converge. Around the same time Mentch and Hooker (2016) proposed a modification to the sub-sampling procedure that allows for the generation of confidence intervals. These procedures estimate model variability: by how much is the prediction likely to vary if the model were retrained on a new set of training data drawn from the underlying distribution? This is not the same as a prediction interval, which is an attempt to bound the true value. To drive this distinction home, figure 7 in appendix D.1 plots the jackknife standard deviation and model error for two one-dimensional test functions and shows that the jackknife can greatly underestimate the true prediction error. Despite this mismatch many works have used the jackknife methods of Wager *et al* (2014) as if they were a prediction interval (Ling *et al* 2017, Ruesch *et al* 2020, Wahab *et al* 2020, Carrella 2021, Lepioufle *et al* 2021, Roman *et al* 2021), sometimes leading to sub-optimal results. For example, when using the IJ to estimate uncertainty as part of a sequential learning study, Rohr *et al* (2020) take note of the '...general overconfidence of [jackknife-based] methods.'

So how does one generate a prediction interval? Quantile regression forests (QRF) (Meinshausen 2006) were an early attempt. A separate model is trained for each desired quantile. QRFs are highly versatile but require large amounts of data to train and can be noisy (Zhang *et al* 2020). The last few years have seen renewed interest in the subject. Lei *et al* (2018) developed a general 'split conformal' method for producing regression prediction intervals. Conformal methods are flexible in that they make no assumptions about the model. They generate prediction intervals by examining the residuals on new test points. The split conformal method reserves some training data for this purpose, hence it is data-inefficient. Dewolf *et al* (2023) conduct a review of methods to generate prediction intervals on regression problems and find that different methods are better at different problems, but all methods can achieve high accuracy if a conformal method is used to recalibrate the prediction intervals.

Zhang *et al* (2020) specialized the conformal approach to random forest and made it more efficient by considering the OOB residuals. The OOB predictions are those made by base learners that were not exposed to a given training point, and hence they provide a reasonable proxy for how the model will perform on new data points. Zhang *et al* (2020) prove that their OOB prediction interval has an asymptotically correct coverage rate given certain assumptions. Kim *et al* (2020), Barber *et al* (2021)expand on this idea, constructing the prediction intervals using both the OOB residuals and predictions. This allows them to prove bounds on the coverage that are valid even without large amounts of training data. But one drawback of these approaches is that the interval is not conditioned on the input, meaning it does not generalize well when the noise is heteroskedastic or the training and test data are not drawn from the same underlying distribution. The lack of conditioning is particularly problematic when designing an experiment (e.g. SL), where the prediction interval should be smaller near previously sampled regions of the domain and larger in unexplored ones.

A conditional prediction interval is proposed by Lu and Hardin (2021), who weight the OOB residuals based on how similar each training point is to the test point. Similarity is determined by the tree structure—a test point is similar to training points with which it shares the leaf node. In this way, the prediction interval is sensitive to the local OOB error. Another conditional prediction interval is suggested by Palmer *et al* (2022), who estimate the local uncertainty by calculating the standard deviation over the base learner predictions. This value is shifted and rescaled by constant factors that are estimated based on the residuals computed by cross-validation.

As far as we can tell, the question of multivariate prediction intervals for bagged learners is entirely unexplored in the literature. Looking at other types of models, one can generate multi-output uncertainty

estimates with Gaussian process regression (GPR). By modeling the joint distribution as a multivariate normal, GPR can in principle estimate the covariance between any two (input point, output variable) pairs. But in practice, choosing a kernel and evaluating these covariances can be complicated. A common technique, linear model of coregionalization, yields a correlation coefficient for each output pair that is not conditioned on the input. Conditional correlation can be introduced, but requires that the practitioner be opinionated as to the underlying structure of the data (Marcotte 2012). One can also use a deep neural network to predict both the mean and the covariance matrix (Russell and Reale 2022), but due to the large amount of data required, deep learning is not always feasible for scientific research problems.

Our contribution is to establish a method that produces conditional multivariate prediction intervals for bagged models. For multivariate problems that are best represented using a bagged model, practitioners no longer need to choose between model accuracy and access to correlated uncertainty estimates.

3. Recalibrated bootstrap prediction intervals

Consider N training examples $Z_i = (\vec{x}_i, \vec{y}_i)$, where $\vec{x}$ is an input vector and $\vec{y}$ is an output vector. The inputs can be of any type but the outputs are real-valued. Assume that the training data are drawn according to some ground-truth function $f(\vec{x})$ with sampling noise parameterized by a distribution $\epsilon(\vec{x})$ that could be heteroskedastic and non-uniform. That is, $\vec{y}_i = f(\vec{x}) + \epsilon(\vec{x})$.

We draw B bootstrap samples of the training data, each of which is generated by sampling N times with replacement. The resulting sample is used to train a base learner that makes predictions $\vec{t}_b(\vec{x})$. The model prediction is the mean of the base learners: $\hat{\vec{\theta}}(\vec{x}) = \frac{1}{B}\sum_{b=1}^{B} \vec{t}_b(\vec{x})$. We use a carat to denote a quantity that is estimated over the bootstrap samples.

We propose a prediction interval that is given by a normal distribution centered on $\hat{\vec{\theta}}(\vec{x})$. Though the true interval is not necessarily normal, this approximation is worth considering because it facilitates the generalization to multiple outputs. The only thing that needs to be computed is the covariance matrix of the normal distribution. We take this to be the covariance matrix of the bootstrap predictions, but we rescale each entry using recalibration factors that are computed using the OOB predictions. We use the notation $(-i)$ to determine a quantity computed over the OOB trees for training point i.

We first consider a single response y and show how to generate a standard deviation that corresponds to a normal prediction interval. We choose a confidence level parameter $p \in (0, 1)$ and calculate the equivalent number of standard deviations of a normal distribution: $\eta(p) = \Phi^{-1}(\frac{1+p}{2})$ where Φ is the CDF of a unit normal distribution. Given p, Algorithm 1 describes how to calculate the recalibration factor α and apply it to new predictions.

Like Palmer *et al* (2022), this method computes factors to recalibrate the bootstrap standard deviation. But there are two key differences. First, using the OOB residuals instead of cross-validation makes the method more data-efficient. Second, as we will show in the subsequent analysis, using the p-percentile instead of the maximum likelihood estimate (MLE) makes the method more robust to outliers as long as the chosen value of p is not too extreme.

The generalization to a multivariate prediction interval is straight-forward. For each pair of outputs we calculate a recalibrated covariance using the individual recalibration factors. Let α_j and α_k be the recalibration factors for outputs j and k. For a given output j let $t_{bj}(\vec{x})$ be the prediction of tree b, $\hat{\theta}_j(\vec{x})$ be the mean prediction, and $\hat{\sigma}_j(\vec{x})$, written explicitly in equation (1), be the recalibrated standard deviation that forms the univariate prediction interval

$$\hat{\sigma}_j(\vec{x}) = \alpha_j \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(t_{bj}(\vec{x}) - \hat{\theta}_j(\vec{x}))^2}. \tag{1}$$

The covariance estimate is then given by equation (2)

$$\begin{aligned}\hat{\sigma}^2_{jk} &= \frac{1}{B-1}\sum_{b=1}^{B}\Big(\alpha_j(t_{bj}(\vec{x}) - \hat{\theta}_j(\vec{x}))\Big)\Big(\alpha_k(t_{bk}(\vec{x}) - \hat{\theta}_k(\vec{x}))\Big) \\ &= \frac{\sum_b \Big(t_{bj}(\vec{x}) - \hat{\theta}_j(\vec{x})\Big)\Big(t_{bk}(\vec{x}) - \hat{\theta}_k(\vec{x})\Big)}{\sqrt{\Big(\sum_b (t_{bj}(\vec{x}) - \hat{\theta}_j(\vec{x}))^2\Big)\Big(\sum_b (t_{bk}(\vec{x}) - \hat{\theta}_k(\vec{x}))^2\Big)}}\hat{\sigma}_j(\vec{x})\hat{\sigma}_k(\vec{x}) \\ &\equiv \hat{\rho}_{jk}(\vec{x})\hat{\sigma}_j(\vec{x})\hat{\sigma}_k(\vec{x}).\end{aligned} \tag{2}$$

Algorithm 1: Recalibrated bootstrap prediction interval in one dimension.

Stage : Determine recalibration factor α;
Given : N training points $(\vec{x}_i, y_i)$;
Given : B bootstrap samples and base learners $t_b(\vec{x})$;
Given : A confidence level parameter, $p \in (0, 1)$;
for all training points $\vec{x}_i$ **do**
  Calculate the OOB mean, $\hat{\theta}_{(-i)}(\vec{x})$;
  Calculate the OOB standard deviation, $\hat{s}_{(-i)}(\vec{x})$;
  Calculate the OOB standard residual, $|\tilde{r}_{\text{oob}}| = \dfrac{|\hat{\theta}_{(-i)}(\vec{x}) - y_i|}{\hat{s}_{(-i)}}$;
end
Let $\tilde{r}_p$ be the p-percentile value of the $|\tilde{r}_{\text{oob}}|$;
Set α equal to $\tilde{r}_p/\eta(p)$, where $\eta(p)$ is the equivalent number of standard deviations of a normal distribution;
Stage : estimate prediction interval for a new input;
Given : new input $\vec{x}$;
Calculate $\hat{s}(\vec{x})$, the standard deviation over the predictions $t(\vec{x})$;
The prediction interval is defined by a normal distribution with mean $\hat{\theta}(\vec{x})$ and standard deviation $\alpha * \hat{s}(\vec{x})$

The bootstrap correlation coefficient $\hat{\rho}_{jk}(\vec{x})$ is shown to be the ordinary Pearson correlation coefficient calculated over the tree-wise predictions. If the resulting covariance matrix is $\hat{\Sigma}(\vec{x})$ then the prediction interval region is defined by a normal distribution $\mathcal{N}(\hat{\vec{\theta}}(\vec{x}), \hat{\Sigma}(\vec{x}))$. We refer to this distribution as the *prediction distribution*.

This is a pleasing result—by forcing the prediction distribution to be normal and by using the tree-wise standard deviation as a proxy for uncertainty, we have arrived at a multivariate distribution that is nearly free to generate and evaluate, since it makes use of the existing bootstrap predictions, and capable of fully describing a multivariate prediction interval without large amounts of training data. Other methods to compute a prediction interval, such as Zhang *et al* (2020) or Lu and Hardin (2021), are parameter-free but do not have a clear multivariate generalization, and any such generalization would likely require the amount of data to grow exponentially with the number of outputs.

The efficacy of this procedure will be established in subsequent section by showing that the prediction distribution is well calibrated and that it leads to more efficient SL. But it is also worth briefly considering the theoretical justification for Algorithm 1, which presupposes that the standard OOB residuals are drawn from the same distribution as the test set residuals and also that the bootstrap variance is proportional to the squared residual. We show in appendices B.1 and B.2 that these are reasonable propositions, given that the training and test sets are drawn independently from identical distributions. In real-world data sets, including those used in this work, the training and test sets are often drawn from non-identical distributions. But as we will see, the recalibrated bootstrap still performs well.

If the standard residuals are normally distributed, then the choice of p has no effect on the rescaling factor. In practice we expect some non-normality. How strongly does the distribution deviate from normality and how does that affect the resulting prediction intervals? We investigate these questions numerically, by training a large number of random forest models on different training data draws and examining the distribution of standard OOB residuals. Results are shown in figure 1(a) for the Friedman–Grosse function (see appendix A.3). Considering both the PDF and the CDF we see that the values are evenly distributed and close to normal, but have slightly fatter tails. The inability to capture these tails is one drawback of restricting the prediction interval to be a normal distribution. The results on other test problems are similar (see appendix D.3), though real-world data sets tend to produce a more skewed distribution. Even in those cases the normality assumption is a useful first-order approximation, as we will see in section 5.

In figure 1(b) we plot the recalibration factor vs. p. Consistent with the distribution having fat tails, we see the recalibration factor increase around $p = 0.9$ because a broader distribution is needed to capture the larger residuals. We also consider the recalibration factor that maximizes the total log likelihood of the OOB residuals, similar to what is proposed in Palmer *et al* (2022), and plot it as a dashed line. Maximizing the log likelihood proposes a larger recalibration factor because it is sensitive to the penalty imposed by underestimating the large residuals.

For simplicity we set $p = 0.683$ for most of the remainder of this manuscript, corresponding to one standard deviation. A larger value of p would produce prediction intervals that are slightly wider. Exactly how this impacts SL depends on the acquisition function, and is a potential topic for further study. In section D.5 we compare the results of simulated SL using both $p = 0.683$ and $p = 0.95$ (corresponding to two standard deviations) and find that there is no statistical difference.

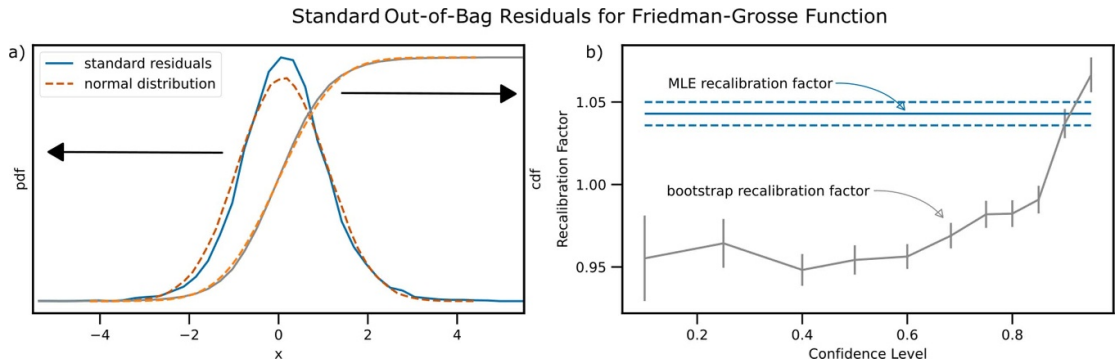

Figure 1. The standard out-of-bag (OOB) residuals approximate a normal distribution, hence the recalibrated bootstrap is expected to produce a well-calibrated prediction interval. This is seen by (a) comparing the PDF and CDF to that of an ideal normal and (b) checking the uniformity of the recalibration factor as a function of confidence level. We also calculate the recalibration factor generated using the MLE method and see that it is strongly influenced by outliers. Data were generated using the Friedman–Grosse function with a noise level of 2.0. There are 128 training points, 64 bags, and 100 trials.

4. Numerical experiments: how well calibrated is the proposed prediction interval?

We perform numerical experiments to investigate the accuracy of the prediction intervals proposed by the recalibrated bootstrap method. We consider modifications of two commonly-used synthetic problems, Friedman–Silverman and Friedman–Grosse, with added Gaussian noise. We also consider two real-world data sets, 'thermoelectrics' (Gaultois *et al* 2013) and 'mechanical properties' (Borg *et al* 2020), both of which have uncharacterized noise that is expected to be heteroskedastic and non-normal. All of the data sets are dense, although the methods described here are amenable to sparse data as well. Details of the data sets are in appendix A.

For each test problem we generate training and test data sets, train a random forest model on the training data, apply the model to the test data, and evaluate the performance of the proposed prediction intervals by comparing them to the 'observed' values. For all models we calculate the recalibration factor using $p = 0.683$, corresponding to $\eta \approx 1$.

All random forest models were trained with the package Lolo (Hutchinson 2016), which is available under the Apache License 2.0. Unless otherwise stated, all forests contained 64 decision trees. All decision trees were grown to full depth and all inputs were considered at each split. The split was chosen to maximize the reduction in total variance summed over all outputs. All outputs were standardized before training to have mean 0 and variance 1.

4.1. Metrics

We consider three metrics of prediction interval quality.

Standard error: the mean of the absolute residual divided by the predicted uncertainty, as given in equation (3) (the sum is over the test data). This value should be around 1.0, and it should be stable as the number of training data are varied. This metric is only used for univariate prediction intervals

$$\texttt{Standard error} = \frac{1}{M}\sum_j \frac{|\hat{\theta}(\vec{x}_j) - y_j|}{\hat{\sigma}(\vec{x}_j)}. \tag{3}$$

Standard confidence: how closely does the number of residuals within some magnitude match the number of residuals that are *expected* to be within that magnitude? The magnitude of the residual is the Mahalanobis distance, $r_M = \sqrt{\vec{r}^T \hat{\Sigma}(\vec{x}_j)^{-1}\vec{r}}$, where $\vec{r} = \hat{\vec{\theta}}(\vec{x}_j) - \vec{y}_j$ is the residual. The squared Mahalanobis distance follows a χ^2 distribution with d degrees of freedom, where d is the number of output dimensions. For a given coverage level $p_c \in (0,1)$ we can therefore use the inverse CDF of χ^2_d to calculate the associated cutoff distance η_c, and count the number of observations for which $r_M < \eta_c$. In one dimension this reduces to equation (4)

$$\texttt{Standard confidence} = \frac{1}{M}\sum_j \mathbb{1}\left[\frac{|\hat{\theta}(\vec{x}_j) - y_j|}{\hat{\sigma}(\vec{x}_j)} \leqslant \Phi^{-1}\left(\frac{1+p_c}{2}\right)\right]. \tag{4}$$

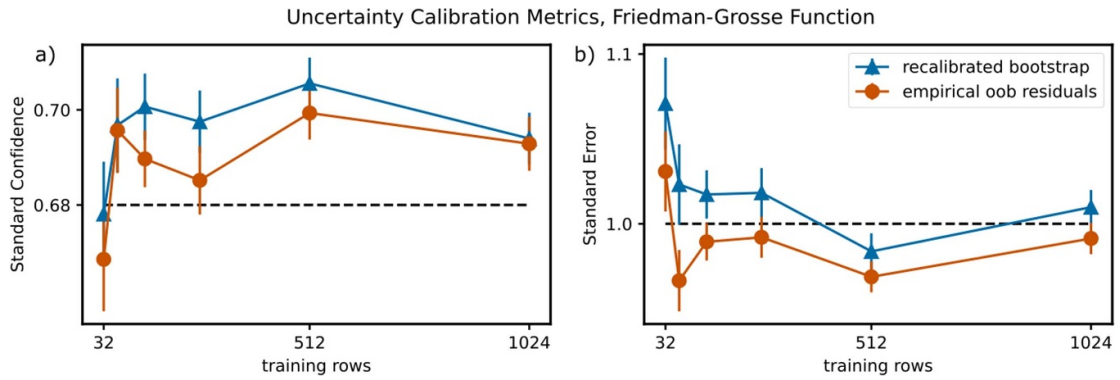

Figure 2. The recalibrated bootstrap produces a well-calibrated univariate prediction interval on synthetic data, as seen by examining (a) standard confidence and (b) standard error. The empirical OOB residuals method (Zhang *et al* 2020) also performs well on this problem. Data are generated using the Friedman–Grosse function with normally distributed noise of magnitude 2.0. There are 128 test points and each random forest has 64 bags. 64 trials were run, each involving a different set of training and test data. Error bars show one standard error.

If the standard confidence is greater than p_c then the model is under-confident, and if it is less than p_c then the model is over-confident. In this work we use 'standard confidence' to refer to the case when $p_c \approx 0.683$.

Median negative log probability density (MNLPD): Let $p(\vec{y}; \hat{\vec{\theta}}(\vec{x}), \hat{\Sigma}(\vec{x}))$ be the probability density function of the prediction distribution at point $\vec{x}$. The NLPD for test point $(\vec{x}_j, \vec{y}_j)$ is given by equation (5). Lower values are better. NLPD penalizes both over- and under-confident prediction intervals. Because NLPD is prone to outliers (especially for the jackknife method), here we consider the median value over all test points

$$\texttt{NLPD}(\vec{x}_j) = -\ln(p(\vec{y}_j; \hat{\vec{\theta}}(\vec{x}_j), \hat{\Sigma}(\vec{x}_j))). \tag{5}$$

4.2. Univariate calibration

We show that the recalibrated bootstrap is well-calibrated by calculating the standard confidence and standard error for data generated using the Friedman–Grosse function. We also calculate these metrics using a non-parametric prediction interval calculated via the empirical OOB residuals, as proposed in Zhang *et al* (2020). As shown in figure 2 both methods achieve standard confidence and standard error close to the expected values. In figure 9 we consider several other test problems. Performance is generally good for both methods, but the skewed real-world data poses more of a challenge. In particular, for the ZT data, the 'empirical OOB residuals' method is highly over-confident. The recalibrated bootstrap proves more robust.

The stability of the recalibrated bootstrap is further explored in the appendices. In appendix D.2 we consider what happens when the test and training data are drawn from different distributions. The recalibrated bootstrap is significantly more robust than the 'empirical OOB residuals' method. We also consider the impact of the number of bags, and find it is largely irrelevant (figure 10). Finally we consider the behavior of the recalibrated bootstrap in the high-noise limit, finding that it correctly identifies the noise as the primary source of uncertainty (figure 11).

While an in-depth comparison of univariate prediction intervals is beyond the scope of this work, we have shown that the recalibrated bootstrap is well-calibrated in a variety of situations, including those with real-world data for which the error is not expected to be Gaussian. We have also shown that it compares favorably to a distribution-free method of construction a prediction interval. In the subsequent sections we show that it also produces well-calibrated multivariate prediction intervals and that this quality results in significantly more efficient SL.

4.3. Multivariate calibration

To investigate the multivariate case we calculate $\hat{\sigma}_j(\vec{x})$ using the recalibrated bootstrap method and consider four different methods of calculating $\hat{\rho}_{jk}(\vec{x})$.

(a) **Trivial**: $\hat{\rho}_{jk} = 0$, the probability of satisfying each objective is considered independently. In the absence of existing methods to compute multivariate prediction intervals, this is the baseline approach.
(b) **Training data**: calculate the Pearson correlation coefficient over the training data and and use that value for all predictions.

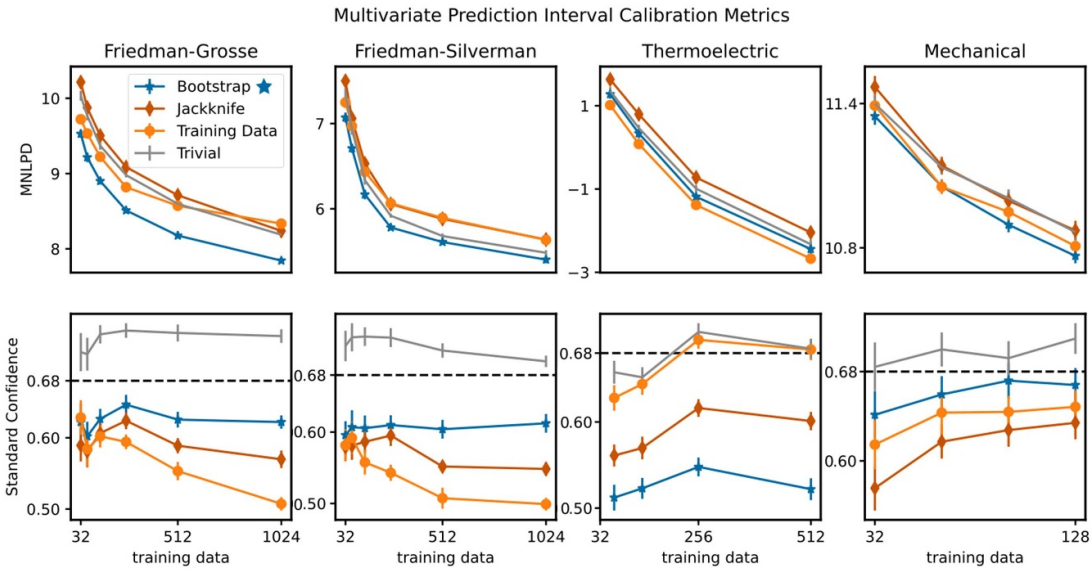

Figure 3. The recalibrated bootstrap produces the best overall multivariate prediction intervals on a suite of test problems, as judged by a low MNLPD (top row) and a standard confidence that is close to 0.68 (bottom row). Error bars show one standard error. The four test problems are Friedman–Grosse with noise 2.0 (128 test points, 16 trials), Friedman–Silverman with noise 2.0 (128 test points, 16 trials), thermoelectrics (64 test points, 32 trials), and mechanical properties (48 test points, 32 trials). See appendix A for details. In all instances the random forest models have 64 bags.

(c) **Jackknife**: calculate the jackknife variance, V_J, and the jackknife covariance, Cov_J, and set $\hat{\rho}_{jk} = \text{Cov}_J[j,k]/\sqrt{V_J[j] * V_J[k]}$. See appendix C for more details.

(d) **Bootstrap**: calculate the Perason correlation coefficient over the tree-wise predictions as in equation (2). This is the approach we are proposing here.

It might seem strange to consider the jackknife method, since it produces a quantity that is known to be more confident than a prediction interval. But it is possible that this bias will cancel out between the variance and covariance terms, leaving us with a decent estimate of correlation.

Figure 3 shows the MNLPD and Standard Confidence values for four test problems using each approach to calculating ρ_{jk}. We see that the bootstrap method, which is highlighted with a star icon, generally has the best MNLPD and a standard confidence that is good but slightly under-confident. The trivial method is slightly over-confident by a similar amount. The thermoelectrics data set is an exception, for which the bootstrap method is significantly under-confident. Yet as we will see in the next section, that does not prevent the recalibrated bootstrap from achieving superior performance on simulated SL.

5. Application to multi-objective SL

A lower MNLPD value is all well and good, but what we really want to know is whether or not this prediction distribution can help us make better decisions. We consider SL, a type of active learning, as a test problem (Ling *et al* 2017). In SL, a user attempts to find an input point $\vec{x}$ that will lead to output $\vec{y}$ that simultaneously satisfies their objectives. Given initial training data, we train a multi-output random forest model. The model makes predictions at all unknown points, and an acquisition function is used to evaluate the suitability of some test point. The highest-scoring test point is chosen for measurement. If it satisfies the objectives then we are done. If not then it is added to the training set and the cycle repeats. This is similar in practice to Bayesian Optimization (Shahriari *et al* 2016), but with a non-Bayesian model for the objectives and an acquisition function defined only on the posterior distribution of the predictions and not on the posterior of the objective model parameters.

Any multi-objective acquisition function may be used. Here we use the predicted probability of the candidate lying in the satisfactory region. This probability is estimated by drawing 10 000 samples from the prediction distribution. In other situations it might be preferable to use other acquisition functions, such as the probability of being Pareto non-dominated (del Rosario *et al* 2020).

We simulate SL on two problems. First, we modify the Friedman–Grosse function to generate two outputs that have a non-trivial relationship to each other. The outputs are positively correlated in one region and negatively correlated in another region. See appendix A.3 for more details. The objectives are that each

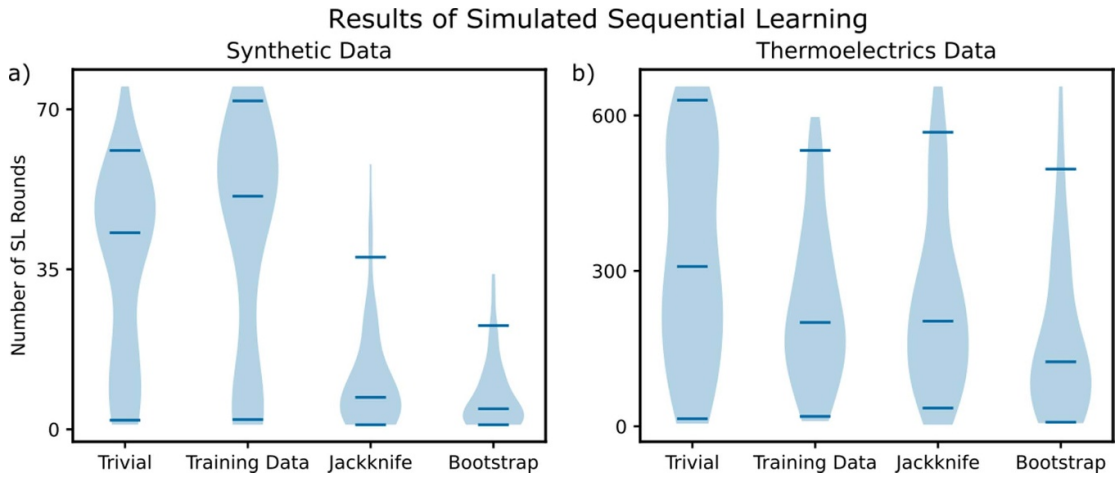

Figure 4. Using the recalibrated bootstrap decreases the number of iterations required to find a configuration that satisfies multiple competing objectives. It is better than the jackknife method and much better than the trivial method, which ignores correlations. The underlying data for the (a) synthetic and (b) thermoelectrics problem are plotted along with dashed lines to indicate the satisfactory regions. Violin plots in (c) and (d) show the distribution of results over 64 trials, along with lines to indicate the 5th percentile, median, and 95th percentile. The synthetic data problem starts with 16 training data and the thermoelectrics problem starts with 32 training data.

Table 1. Distribution of rounds required to find a satisfactory candidate in simulated sequential learning. Mean values are reported along with one standard error. Results are calculated over 64 independent trials.

	Method	Mean	5th percentile	Median	95th percentile
7em synthetic data (2 objectives)	Trivial	36.0 ± 2.6	2	43	61
	Training data	41.8 ± 3.1	2	51	72
	Jackknife	11.2 ± 1.5	**1**	7	37.7
	Bootstrap	$\mathbf{7.5 \pm 1.0}$	**1**	**4.5**	**22.7**
	Random	36	3	33	87
7em thermoelectrics data (4 objectives)	Trivial	330 ± 25	15	309	630
	Training data	230 ± 19	19	200.5	532.5
	Jackknife	240 ± 21	35	203	568
	Bootstrap	$\mathbf{170 \pm 20}$	**8**	**125**	**497**
	Random	328	33	328	623

output should exceed 22, a value chosen to make the problem challenging. As seen in figure 5 of appendix A.3, only two points satisfy both objectives.

Second, we consider the thermoelectrics data set and devise a problem for which there is one solution: ZT > 1.25, Seebeck coefficient $> 175\ \mu\mathrm{V\,K^{-1}}$, power factor $> 5 \times 10^{-3}\ \mathrm{W\,m\,K^{-2}}$, and thermal conductivity $> 1.5\ \mathrm{W\,mK^{-1}}$. The data are plotted in figure 6 of appendix A.5.

Figure 4 shows the results of 64 trials of simulated SL. Each trial involved a different set of initial training data points. The distribution of rounds to find a satisfactory candidate is shown as a violin plot. The 5th percentile, median, and 95th percentile values are indicated with horizontal lines. These values, along with the mean and standard error, are shown in table 1. Table 1 also includes the values that would be expected if test points were drawn randomly. The bootstrap consistently requires the fewest iterations to find a satisfactory candidate. Compared to the trivial method, the median trial requires 90% fewer iterations for the synthetic problem and 60% fewer iterations for the thermoelectrics problem. Perhaps more importantly, the trivial method suffers from a long tail of disastrous trials in which the performance is worse than that of random guessing. The recalibrated bootstrap largely avoids this pathology.

6. Conclusions

We propose a 'multi-output recalibrated bootstrap' method to generate multivariate prediction intervals for random forest or any other bagged ensemble model. The covariance matrix of the bootstrap predictions defines a multivariate normal distribution, but the values are rescaled using the OOB residuals. We show that this prediction interval is well-calibrated by testing it against several data sets.

Our focus is not, however, the prediction interval itself, but its applicability to multi-output SL. We simulate SL on both synthetic and real data and show that using the recalibrated bootstrap significantly decreases the number of iterations required to find a high-performing candidate by between 2× and 8× compared to common alternatives. Although this study is limited in that it only considers a few data sets, we feel that the dramatic improvement seen on real-world, noisy, imbalanced data is promising enough to merit its use and further study.

Developing the next-generation materials crucial to renewable energy generation and storage is an ideal problem for multi-objective SL: there are typically multiple ambitious goals that compete against each other, existing data sets are modest in size, each experiment can be enormously expensive, and there is a positive social impact. But SL is a value-agnostic algorithm for optimizing any complex system using machine learning, and it can just as easily be used to increase the lethality of explosives or the potency of a nicotine cartridge. Our contribution increases the efficacy of SL, and hence its societal impact depends on the problems it is applied to.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/CitrineInformatics/multivariate-prediction-intervals.

Appendix A. Compendium of test problems

The sections below describe how to generate ground-truth data for each test problem. For synthetic problems additional noise may be added in the form of $\epsilon * \mathcal{N}(0,1)$, where ϵ is the noise level.

A.1. Tophat

The tophat method used in figure 7(a) is defined in equation (6)

$$y = \begin{cases} 1.0, & \text{if } |x| < 0.33 \\ 0.5, & \text{if } 0.33 \leqslant |x| < 0.67 \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

A.2. Cubic

The cubic method used in figure 7(b) is simply $y = x^3$.

A.3. Friedman–Grosse

The Friedman–Grosse function (Friedman *et al* 1983), in equation (7) is defined on the unit hypercube in at least five dimensions. In this paper we use eight dimensions, but dimensions 6–8 do not contribute to the output

$$y_0 = 10\sin(\pi x_0 x_1) + 20(x_2 - 0.5)^2 + 10x_3 + 5x_4. \tag{7}$$

A.3.1. Additional outputs for calibration studies

For the purposes of investigating the prediction distribution's calibration (figure 3) we generate two additional outputs, both of which are correlated with Y_0. The first additional output, Y_1, has some fixed linear correlation, ρ, with Y_0. The second additional output, Y_2, varies between being positively and negatively correlated with Y_0.

We now describe the procedure to generate a new variable, Y_1, that has some fixed linear correlation with the existing output variable, Y_0. Assume that we have first drawn N points $\vec{x}_i$ with associated output values y_{0i}. To created a new variable with known correlation we want to mix together those y_{0i} with an uncorrelated signal, $Z^{'}$. We create an uncorrelated signal by drawing N values from a unit normal distribution, z_i, performing linear least squares regression $Z = mY$, and computing an orthogonal signal $Z' = Z - mY$. Let $\sigma_{Z'}$ be the standard deviation of the residuals and let σ_{Y0} be the standard deviation of the Y_0 values. For a desired correlation coefficient ρ, the first additional output is $Y_1 = \rho\sigma_{Z'}Y_0 + \sqrt{1-\rho^2}\sigma_{Y0}Z'$.

Because Y_0 and $Z^{'}$ are orthogonal, $\mathrm{Cov}(Y_1, Y_0) = \rho\sigma_{Z'}\mathrm{Cov}(Y_0, Y_0) = \rho\sigma_{Z'}\sigma^2_{Y_0}$. Also, $\mathrm{Var}(Y_1) = \rho^2\sigma^2_{Z'}\sigma^2_{Y_0} + (1-\rho^2)\sigma^2_{Y_0}\sigma^2_{Z'} = \sigma^2_{Y_0}\sigma^2_{Z'}$. The correlation coefficient between Y_0 and Y_1 is therefore exactly equal to ρ. In this work we set $\rho = 0.9$ unless otherwise specified.

The second additional output is defined with a quadratic equation. Let μ_{y0} be the mean of the values y_{0i}. We then define $y_{2i} = (y_{0i} - \mu_{y0})^2 + f * \mathcal{N}(0,1)$, where f is some adjustable parameter. In this work we set $f = 0.5$ unless otherwise specified, which leads to a strong (but non-uniform) correlation.

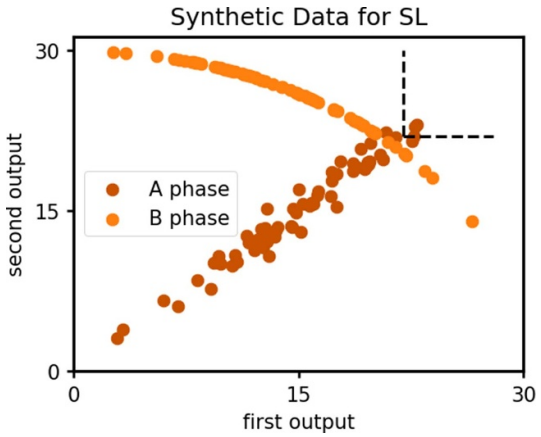

Figure 5. Synthetic data set generated using the Friedman–Grosse function for simulated sequential learning. In one region ('A phase') there is a strong positive correlation between the outputs, while in the other region ('B phase') there is a strong negative correlation between the outputs. The dashed lines correspond to the two objectives used in the simulated sequential learning study.

A.3.2. Additional output for SL

For the purposes of synthetic SL we generate the second output differently, because the intent is to create a problem in which the correlation coefficient varies depending on the region of input space. We first generate 128 inputs by sampling uniformly from the hypercube and calculate the first output y using equation (7). We then create another input, 'phase,' and randomly assign each point to either 'A' or 'B'. For the points in phase A, the second output is generated using the linear procedure in the previous subsection, using $\rho = 0.98$. For the points in phase B, the second output is equal to $\sqrt{30^2 - y^2}$ (30 is the largest possible value of the Friedman–Grosse function).

This data set is plotted in figure 5 along with dashed lines at 22 for both the first and second output, corresponding to the objectives used in the SL simulation. We see that only two points satisfy both objectives.

A.4. Friedman–Silverman

The Friedman–Silverman function (Friedman and Silverman 1989), in equation (8) is defined on the unit hypercube in at least five dimensions. In this paper we use 12 dimensions, but dimensions 6–12 do not contribute to the output

$$y_0 = 0.1\mathrm{e}^{4x_0} + \frac{4}{1 + \mathrm{e}^{-20(x_1 - 0.5)}} + 3x_2 + 2x_3 + x_4. \tag{8}$$

For the purposes of investigating the prediction distribution's calibration (figure 3), two additional outputs are generated in the same way as for the Friedman–Grosse function.

A.5. Thermoelectrics

The data come from Gaultois *et al* (2013). There are 688 thermoelectric materials, each with a chemical formula, crystallinity (either poly-crystalline or single crystal), and temperature at which the measurements were taken. The chemical formulae were featurized using the Matminer package (Ward *et al* 2018) to calculate the Magpie features (Ward *et al* 2016). Along with temperature and crystallinity, these are the inputs to the ML models. There are five measured output properties: ZT, Seebeck coefficient, thermal conductivity, power factor, and log resistivity. One projection of this data set is plotted in figure 6.

A.6. Mechanical properties

The data come from Borg *et al* (2020). There are 630 multi-principal element alloys, some measured under a variety of conditions. The data set contains several inputs and several outputs, many of which are sparse. To yield a dense training table we only consider the following inputs: processing method (cast, wrought, anneal, powder, and other), crystal structure (bcc, fcc, and other), test type (compression or tension), and chemical formula. The chemical formulae are featurized as with the thermoelectrics data. We only consider two output properties: Young's modulus and elongation.

We only keep rows that have values for all inputs and outputs. We only keep rows that were measured at room temperature (between 20°C and 25°C). If there are multiple rows with identical inputs, we average their output properties. This yields 287 rows.

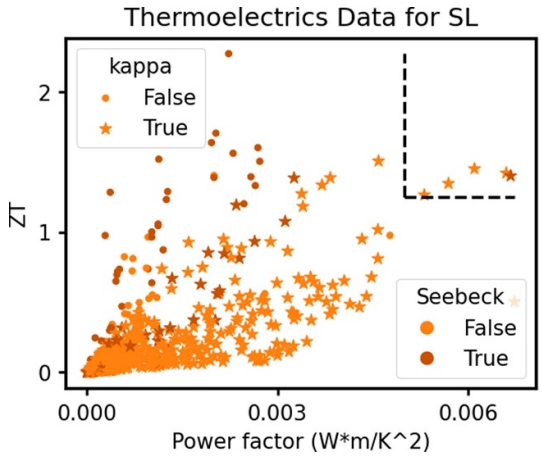

Figure 6. A projection of the thermoelectrics data set, meant to visualize the difficulty of the simulated SL problem. Dashed lines indicate the objectives for ZT (1.25) and power factor (0.005 $\mathrm{W m K^{-2}}$). Points are colored red or orange according to whether or not the seebeck coefficient exceeds the objective of 175 $\mu \mathrm{V K^{-1}}$. The marker is a star or a circle depending on whether or not the thermal conductivity, kappa, exceeds the objective of 1.5 $\mathrm{W mK^{-1}}$.

Appendix B. Analysis of the recalibrated bootstrap

A decision tree partitions the domain into rectangular subspaces, each corresponding to a terminal node of the tree. The prediction in this subspace is the average of the values of the training data that appear in that node. This training data $\rightarrow$ subspace correspondence is a complicated function of the training data, the splitting function, and random variables (such as which input dimensions are being considered), but we can write the tree's predictions as a weighted sum of the training data. The RF prediction is therefore given by equation (9), where $\{Z_n\} \in Z$ is represents the training data draw of size n and $\phi \in \Phi$ represents a random variables that controls the splitting. The weights $w_{bi}(\vec{x})$ are non-zero only when $\vec{x}_i$ is 'close' to $\vec{x}$, and the weights sum to 1 for each tree and value of $\vec{x}$

$$\hat{\theta}(\vec{x}) = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{N} w_{bi}(\vec{x};\{Z_n\},\phi)y_i. \tag{9}$$

The OOB prediction for $\vec{x}_i$ is similar, but the sum is taken only over the trees for which $\vec{x}_i$ is not present in the bootstrap sample. This is, on average, B/e trees

$$\hat{\theta}_{(-i)}(\vec{x}_i) = \frac{1}{B/e}\sum_{b:z_i\notin Z_b}\sum_{j} w_{bj}(\vec{x}_i;\{Z_n\},\phi)y_i. \tag{10}$$

B.1. Asymptotic equivalence of standard residual distributions

We wish to show that the distribution of true standard residuals is equivalent to the distribution of OOB standard residuals:

$$\left\langle \frac{\hat{\theta}_{(-i)}(\vec{x}_i) - y_i}{\hat{s}_{(-i)}(\vec{x}_i)} \right\rangle_{\{Z_n\},\phi,z_j} \stackrel{?}{=} \left\langle \frac{\hat{\theta}(\vec{x}) - (f(\vec{x}) + \epsilon\mathcal{N}(0,1)}{\hat{s}(\vec{x})} \right\rangle_{\{Z_n\},\phi,x,y}. \tag{11}$$

The expectation is taken over all training data draws $\{Z_n\}$ and parameters ϕ, and on the LHS over all training data $(\vec{x}_j, y_j) \in \{Z_n \backslash z_i\}$ while on the RHS it is over all $\vec{x}$ in the domain and all associated observations drawn from $y = f(\vec{s}) + \epsilon\mathcal{N}(0,1)$.

Consider what a fixed point $\vec{x}$ contributes to each side of this equation. Because the labels y_i are generated according to $f(\vec{x}) + \epsilon\mathcal{N}(0,1)$, and drawing the noise at $\vec{x}$ is independent of $\hat{s}(\vec{x})$, we can replace both y_i and $f(\vec{x}) + \epsilon\mathcal{N}(0,1)$ in expectation with $f(\vec{x})$. On the LHS we therefore have the mean standard residual at $\vec{x}$ for an RF model trained with $N-1$ iid training points and B/e trees. On the right we have the mean standard residual at $\vec{x}$ for an RF model trained with N iid training points and B trees. For large values of B and N, these are equivalent.

B.2. Bootstrap variance is sensitive to the true error

The total expected squared error due to a model can be represented as the sum of the bias squared, the model variance, and the noise variance. We wish to show that the bootstrap variance (and hence the recalibrated bootstrap standard deviation) is sensitive to each of these terms.

B.2.1. Noise variance

The bootstrap variance is written in equation (12)

$$\langle \hat{s}^2(\vec{x}) \rangle = \left\langle \frac{1}{B-1} \sum_b (t_b(\vec{x}) - \hat{\theta}(\vec{x}))^2 \right\rangle_{\{Z_n\},\phi} = \frac{B}{B-1} \langle t_b(\vec{x})^2 \rangle_{\{Z_n\},\phi,b} - \frac{B}{B-1} \langle \hat{\theta}(\vec{x})^2 \rangle_{\{Z_n\},\phi}. \tag{12}$$

The simplification is possible because $\hat{\theta}$ is the mean of t_b. First, we consider the expectation of $t_b(\vec{x})^2$:

$$\langle t_b(\vec{x})^2 \rangle_{\{Z_n\},\phi,b} = \left\langle \left(\sum_i w_{bi}(\vec{x}; \{Z_n\}, \phi)(f(\vec{x}_i) + \epsilon \mathcal{N}(0,1)) \right)^2 \right\rangle_{\{Z_n\},\phi,b}. \tag{13}$$

The weights $w_{bi}(\vec{x})$ mostly depend on which training data are 'close' to $\vec{x}$ in input space. There is some dependence on the exact values of y_i, but we can consider the weights w_{bi} to be roughly independent of the noise drawn from $\epsilon\mathcal{N}(0,1)$. In that case the expectation simplifies to equation (14), where L is the typical number of unique training observations on a leaf node

$$\langle t_b(\vec{x})^2 \rangle \approx \left\langle \left(\sum_i w_{bi}(\vec{x}; \{Z_n\}, \phi) f(\vec{x}_i) \right)^2 \right\rangle_{\{Z_n\},\phi,b} + \frac{\epsilon^2}{L}. \tag{14}$$

Similarly, $\langle \hat{\theta}(\vec{x})^2 \rangle_{\{Z_n\},\phi}$ also has a term that goes as ϵ^2. But since there is an additional average over the bootstrap samples and each bootstrap sample masks off some training data, there will be a larger number of independent noise draws being averaged together and hence the effective value of L is larger. Therefore, $\langle \hat{s}(\vec{x})^2 \rangle$ has a term that is proportional to the noise variance ϵ^2. This term is smaller if there are more training examples on each leaf node, but that makes sense since we would expect a model that averages over more training data to be less sensitive to the noise.

In appendix D we show that, as the noise becomes the dominant term, the recalibrated bootstrap correctly identifies each observation as being an independent random variable with uncertainty equal to the noise.

B.2.2. Model variance

Having considered the noise, we set $\epsilon = 0$ for the remainder of this appendix. We next consider the variance due to the finite training data set size and the parameters of the model, $\langle \hat{\theta}(\vec{x})^2 \rangle_{\{Z_n\},\phi} - \langle \hat{\theta}(\vec{x}) \rangle^2_{\{Z_n\},\phi}$. Assume for now that there is no bias, so $\langle \hat{\theta}(\vec{x}) \rangle = f(\vec{x})$, and without loss of generality assume $f(\vec{x}) = 0$. Our task is to show that $\langle \hat{s}(\vec{x})^2 \rangle \propto \langle \hat{\theta}(\vec{x})^2 \rangle_{\{Z_n\},\phi}$, which by equation (12) is equivalent to showing that $\langle t_b(\vec{x})^2 \rangle_{\{Z_n\},\phi,b} \propto \langle \hat{\theta}(\vec{x})^2 \rangle_{\{Z_n\},\phi}$.

But this is clearly true because $\hat{\theta}(\vec{x})$ is the mean of $t_b(\vec{x})$. Consider some fixed $\{Z_n\}$ and ϕ. The $t_b(\vec{x})$ are the predictions made by all trees trained on bootstrap replicates of $\{Z_n\}$. The $\hat{\theta}(\vec{x})$ are the averages of all sets of B draws from the distribution of $t_b(\vec{x})$. By the law of large numbers, the variance of the latter is equal to the variance of the former times $1/B$. Hence, $\langle \hat{s}(\vec{x})^2 \rangle$ is sensitive to model variance.

B.2.3. Bias squared

Assume for simplicity that the training data are independent and identically distributed, that $f(\vec{x})$ is continuous, and that the random forest is full-depth. In this case the model is asymptotically consistent and free of bias *except* for at the boundaries. As a way of illustrating how the bootstrap variance can pick up bias, we consider the boundary of a one-dimensional test problem. A more rigorous investigation would need to consider multivariate and non-identically distributed data, which is the case for the real-world data sets used here.

Let the domain be $[0,1]$ and let the function $f(x)$ have slope m at $x = 0$. Because the trees are grown to full depth the predicted value at $x = 0$ will be equal to the value of the training point with the smallest value of x.

Because the training data are drawn uniformly this is equal in expectation to $1/(N+1)$. Assuming the linear approximation of $f(x)$ is valid over the length-scale of $1/(N+1)$, the predicted value will therefore be equal in expectation to $m/(N+1)$.

However, roughly $1/e$ trees will be missing this data point. Most of these trees will make a prediction that is equal to the value of the training point with the second-smallest value of x, which in expectation is at $2/(N+1)$ and has a value of $2m/(N+1)$. There are other terms as well, but they are all proportional to $m/(N+1)$. Hence the bias goes as $m/(N+1)$ and the bias squared goes as $m^2/(N+1)^2$.

Now consider the bootstrap variance, which is the mean of $(t_b(0)-\hat{\theta}(0))^2$. As discussed above, $t_b(0)$ is always proportional to $m/(N+1)$. Hence $\hat{\theta}(0) \propto m/(N+1)$. Though the sum is complex, every term has a factor of $m/(N+1)$ that can be factored out, and so the bootstrap variance goes as $m^2/(N+1)^2$, just like the bias squared does.

B.2.4. Conclusion

These arguments are not precise, and in fact we do not expect the bootstrap variance to perfectly capture the true residual. We merely show that it has the capability to pick up on the bias, the variance, and the noise. Its efficacy and ability to balance these three terms is shown through the numerical experiments in this manuscript.

Appendix C. Details of Jackknife variance and covariance (including the bias-correction term)

Wager *et al* (2014) introduces bias-corrected versions of two methods to estimate a confidence interval: the IJ and the JaB. Here we generalize those derivations from variance to covariance. Wager *et al* (2014) found that the IJ and JaB had opposite lowest-order biases, and hence suggested averaging them. In this manuscript, whenever 'jackknife methods' are invoked, it is the average of the IJ and the JaB (co)variance. The square root of this term is referred to as the 'jackknife standard deviation.'

Although the derivations below consider covariance between two outputs predicted by one ensemble model at one input point, they also apply to the covariance between predictions made at two distinct input points or to predictions made by two distinct ensemble models. Indeed, Ghosal (2021) derive an analogue of equation (15) but their focus is on the covariance between the predictions made by two models, in order to test if the models are equivalent.

C.1. IJ covariance

We show that the bias-corrected IJ covariance between outputs j and k at point $\vec{x}$ is given by equation (15), where Y_{bi} is the number of times training datum i appears in bag b

$$\mathrm{Cov}_{IJ}[j,k](\vec{x}) \approx \sum_{i=1}^{N}\left(\sum_{b=1}^{B}\frac{(Y_{bi}-1)(t_{bj}(\vec{x})-\hat{\theta}_j(\vec{x}))}{B}\right)\left(\sum_{b=1}^{B}\frac{(Y_{bi}-1)(t_{bk}(\vec{x})-\hat{\theta}_k(\vec{x}))}{B}\right) - \frac{N-1}{B}\sum_{b=1}^{B}(t_{bj}(\vec{x})-\hat{\theta}_j(\vec{x}))(t_{bk}(\vec{x})-\hat{\theta}_k(\vec{x})). \tag{15}$$

C.1.1. Main term

We follow the arguments in Efron (2014), which themselves are similar to those in Efron (1982), but we generalize to covariance. Consider two estimators, $\hat{\theta}_j$ and $\hat{\theta}_k$, that evaluate some functions θ_j and θ_k for a given distribution of training data. For our purposes, θ_j and θ_k correspond to two outputs that the model predicts. The predictions depend on the bootstrap samples, which are generated by drawing N samples from N training data according to some probability vector, $\vec{p}$ of length N. For an ordinary bootstrap, $\vec{p}=\vec{p}_0$, a uniform vector for which each entry is equal to $1/N$. Equivalently, this can be thought of as an isotropic rescaled multinomial distribution, $\mathbb{M}$.

We are therefore interested in the covariance between $\theta_j(\vec{p})$ and $\theta_j(\vec{p})$ over the multinomial distribution. Both θ_j and θ_k are effectively functions of $\vec{p}$, and we linearize them around $\vec{p}_0$ as $\theta_j(\vec{p}) = \theta_j(\vec{p}_0)$ $+ (\vec{p}-\vec{p}_0)\cdot\vec{U}$, where $\vec{U}$ is the influence function. Similarly, let $\vec{V}$ be the influence function of $\theta_j(\vec{p})|_{\vec{p}_0}$. Because $\sum_i U_i = \sum_i V_i = 0$, the covariance reduces as shown below

$$\begin{aligned}\mathrm{Cov}_{\mathbb{M}}[\theta(\vec{p}),\phi(\vec{p})] &= \mathbb{E}_{\mathbb{M}}[(\theta(\vec{p})-\bar{\theta}(\vec{p}))(\phi(\vec{p})-\bar{\phi}(\vec{p}))]\\ &= \mathbb{E}_{\mathbb{M}}[(\vec{p}-\vec{p}_0)\cdot\vec{U}(\vec{p}-\vec{p}_0)\cdot\vec{V})]\\ &= \mathbb{E}_{\mathbb{M}}[(\vec{p}\cdot\vec{U})(\vec{p}\cdot\vec{V})]\\ &= \mathbb{E}_{\mathbb{M}}\left[\sum_{i=1}^{N} p_i^2 U_i V_i + \sum_{i=1}^{N}\sum_{l\neq i} p_i p_l U_i V_l\right.\\ &= \sum_{i=1}^{N} U_i V_i \mathbb{E}_{\mathbb{M}}[p_i^2] + \sum_{i=1}^{N}\sum_{l\neq i} U_i V_l \mathbb{E}_{\mathbb{M}}[p_i p_l].\end{aligned}$$

The expectation values of p_i^2 and $p_i p_j$ are those of $\mathbb{M}$, for which the mean of each term is $1/N$, the variance of each term is $1/N^2 - 1/N^3$, and the covariance between any pair of terms is $-1/N^3$. This yields the following

$$\mathrm{Cov}_{\mathbb{M}}[\theta_j(\vec{p}),\theta_k(\vec{p})] = \frac{1}{N^2}\sum_{i=1}^{N} U_i V_i.$$

What are U_i and V_i? By taking the derivative, Efron (2014) derives them to be equal to $N\mathrm{Cov}_*[Y_{bi}, t_b(\vec{x})]$. The asterisk indicates that the covariance is taken over the bootstrap samples. The IJ covariance estimate between outputs j and k is therefore given by equation (16), which is a natural generalization of the IJ variance

$$\mathrm{Cov}_{IJ}[j,k](\vec{x}) = \sum_{i=1}^{N} \mathrm{Cov}_*[Y_{bi}, t_{bj}(\vec{x})]\mathrm{Cov}_*[Y_{bi}, t_{bk}(\vec{x})]. \tag{16}$$

When expanding this covariance explicitly we need to know the mean values of Y_{bi} and t_b, which are 1 and $\hat{\theta}$.

C.1.2. Bias correction term

Equation (16) is exact as $B \to \infty$, but in practice we only consider a subset of bags. Let $A_j^B = \mathrm{Cov}_*[Y_{bi}, t_{bj}(\vec{x})]$ and A_j^∞ be the same for infinite B. One term of the IJ sum is, in expectation:

$$\begin{aligned}\mathbb{E}_*[A_j^B A_k^B] &= \mathbb{E}_*[A_j^B]\mathbb{E}_*[A_k^B] + \mathrm{Cov}_*[A_j^B, A_k^B]\\ &= A_j^\infty A_k^\infty + \mathrm{Cov}[A_j^B, A_k^B].\end{aligned}$$

The difference between the finite-B value and the ideal value is therefore given by $-N\mathrm{Cov}_*[A_j^B, A_k^B]$. Writing this out in detail we get expression (17), where the expectation is now taken over the training data

$$-\frac{N}{B}\mathbb{E}\left[\mathrm{Cov}_*[Y_{bi} * (t_{bj}(\vec{x})) - \hat{\theta}_j(\vec{x}), Y_{bi} * (t_{bk}(\vec{x}) - \hat{\theta}_j(\vec{x}))]\right]. \tag{17}$$

As in Wager *et al* (2014) we treat Y_{bi} and t_b as independent, meaning we can pull out the Y_{bi} terms into their own variance term $V_*[Y_{bi}]$, which is equal to the variance of a multinomial distribution, or $(N-1)/N$. The remaining covariance between the trees t_{bj} and t_{bk} is written out explicitly, and we arrive at the following bias-correction term

$$-\frac{N-1}{B}\sum_{b=1}^{B}(t_{bj}(\vec{x}) - \hat{\theta}_j(\vec{x}))(t_{bk}(\vec{x}) - \hat{\theta}_k(\vec{x})). \tag{18}$$

Note that this differs slightly from the bias-correction in Wager *et al* (2014), because they mistakenly set the variance of Y_{bi} equal to 1 and hence have a factor of N instead of $N-1$ (this occurs right before equation 11 in Wager *et al* (2014)).

Combining equations (16) and (18) we arrive at equation (15).

C.2. JaB covariance

We show that the bias-corrected JaB covariance between outputs j and k at point $\vec{x}$ is given by equation (19)

$$\begin{aligned}\mathrm{Cov}_{JaB}[j,k](\vec{x}) \approx &\frac{N-1}{N}\sum_{i=1}^{N}\left(\hat{\theta}_{j(-i)}(\vec{x})-\hat{\theta}_j(\vec{x})\right)\left(\hat{\theta}_{k(-i)}(\vec{x})-\hat{\theta}_k(\vec{x})\right)\\ &-(e-1)\frac{N-1}{B}\sum_{b=1}^{B}(t_{bj}(\vec{x})-\hat{\theta}_j(\vec{x}))(t_{bk}(\vec{x})-\hat{\theta}_k(\vec{x})).\end{aligned} \tag{19}$$

C.2.1. Main term

The derivation of the JaB in Efron (1982) is identical to that of the IJ, but the influence function is different. The generalization to covariance therefore works out the same way and we get equation (20)

$$\mathrm{Cov}_{JaB}[j,k](\vec{x}) = \frac{N-1}{N}\sum_{i=1}^{N}\left(\hat{\theta}_{j(-i)}(\vec{x})-\hat{\theta}_j(\vec{x})\right)\left(\hat{\theta}_{k(-i)}(\vec{x})-\hat{\theta}_k(\vec{x})\right). \tag{20}$$

C.2.2. Bias correction term

The derivation of the JaB bias term is the same as it is for the IJ, above, but instead of A_j^B the relevant term for some training datum i is now $\hat{\Delta}_{ij} \equiv \hat{\theta}_j^B - \hat{\theta}^B_{(-i)j}$ (the dependence on $\vec{x}$ is elided for simplicity) and there is also an overall factor of $(N-1)/N$. In order to evaluate $\mathrm{Cov}_*[\hat{\Delta}_{ij}, \hat{\Delta}_{ij}]$ we follow appendix A of Wager *et al* (2014) but generalize from variance to covariance.

Using the law of total covariance, we can expand this term out as follows:

$$\mathrm{Cov}_*[\hat{\Delta}_{ij}, \hat{\Delta}_{ik}] = \mathbb{E}_*\left[\mathrm{Cov}_*[\hat{\Delta}_{ij}|B_i, \hat{\Delta}_{ik}|B_i]\right] + \mathrm{Cov}_*\left[\mathbb{E}_*[\hat{\Delta}_{ij}|B_i], \mathbb{E}_*[\hat{\Delta}_{ik}|B_i]\right].$$

The 'covariance of expectation' term simplifies just as the analogous 'variance of expectation' term in Wager *et al* (2014), to $\Delta_{ij}\Delta_{ik}\mathcal{O}(1/B)$, where the Δ terms without a hat indicate the average over all possible bootstrap samples.

To evaluate the 'expectation of covariance' term we re-write $\hat{\Delta}$ as a sum over B_i bags that do not contain point i and a subsequent sum over $B - B_i$ bags that do contain point i. On average, $B_i = B/e$

$$\hat{\Delta}_{ij} = \frac{1}{B_i}\sum_{b=1}^{B_i} t_{bj} - \frac{1}{B}\sum_{b=1}^{B} t_{bj} = \frac{B-B_i}{BB_i}\sum_{b=1}^{B_i} t_{bj} - \frac{1}{B}\sum_{b=B_i+1}^{B} t_{bj}.$$

The covariance between $\hat{\Delta}_{ij}$ and $\hat{\Delta}_{ik}$ then breaks out into the sum of many covariance terms, all of the form $\mathrm{Cov}[t_{bj}, t_{b'k}]$. But because the bags are independent, all of these terms are 0 unless $b = b'$. We therefore have B_i instances of $\mathrm{Cov}[t_{bj}|B_i = 0, t_{bk}|B_i = 0]$ and $B - B_i$ instances of $\mathrm{Cov}[t_{bj}|B_i \neq 0, t_{bk}|B_i \neq 0]$, which we denote $\mathrm{cov}_i^{(0)}$ and $\mathrm{cov}_i^{(+)}$

$$\mathrm{Cov}_*[\hat{\Delta}_{ij}|B_i, \hat{\Delta}_{ik}|B_i] = \left(\frac{B-B_i}{BB_i}\right)^2 B_i \mathrm{cov}_i^{(0)} + \frac{1}{B^2}(B-B_i)\mathrm{cov}_i^{(+)}. \tag{21}$$

Equation (21) is identical to the analogous equation in Wager *et al* (2014) except that we have the covariance between t_{bj} and t_{bk} instead of the variance of t_b. Calculating the expectation therefore proceeds the same way and we wind up with, to lowest order, $\frac{e-1}{B}\mathrm{Cov}_*[t_{bj}, t_{bk}]$. Including the factor of $(N-1)/N$ and the sum over N, we wind up with expression (22)

$$-(e-1)\frac{N-1}{B}\sum_{b=1}^{B}(t_{bj}(\vec{x})-\hat{\theta}_j(\vec{x}))(t_{bk}(\vec{x})-\hat{\theta}_k(\vec{x})). \tag{22}$$

Note that this is again slightly different from Wager *et al* (2014), who convert the factor of $N-1$ into a factor of N for unclear reasons.

Combining equations (20) and (22) we arrive at equation (19).

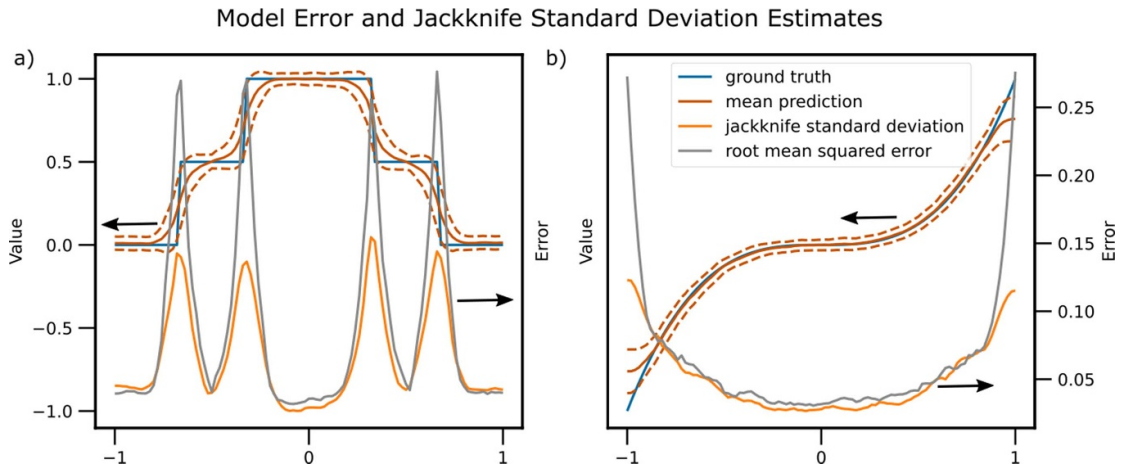

Figure 7. Jackknife standard deviation and true model root mean squared error (RMSE) for two one dimensional test functions: (a) tophat and (b) cubic. The ground truth and predicted value (with ± 1 standard deviation) are plotted on the left axis. The jackknife standard deviation and the RMSE are both plotted on the right axis.

Table 2. Comparison of the 'recalibrated bootstrap' and 'empirical OOB residuals' Zhang *et al* (2020) prediction interval methods on a problem for which the training and test sets are drawn from different distributions.

	MNLPD	Standard RMSE	Standard confidence
Empirical OOB residuals	7.37 ± 0.06	2.12 ± 0.07	0.38 ± 0.02
Recalibrated bootstrap	7.39 ± 0.02	$\mathbf{1.35 \pm 0.07}$	$\mathbf{0.61 \pm 0.03}$

Appendix D. More numerical experiments

D.1. The Jackknife underestimates the true model error

We consider two one-dimensional test problems, a double tophat and a cubic on the domain $[-1, 1]$ (see appendix A for details). We draw 64 training data uniformly, train an RF model, and calculate its predictions on 100 points evenly spaced throughout the domain. We also calculate the jackknife standard deviation (see appendix C) and the squared error at these points. This is averaged over 250 trials. The results, in figure 7, show that a jackknife-based prediction interval can be highly over-confident, in particular when the model is biased.

D.2. Performance on imbalanced data

Table 2 shows univariate uncertainty metrics when the training and test sets are drawn from highly imbalanced distributions. We use the Young's modulus output of the mechanical properties data set. In these data there are points that were measured under tension and points that were measured under compression. The training set consists of 60 tension points and 4 compression points. The test set consists of 32 compression points. We calculate prediction interval metrics for both the recalibrated bootstrap method and the method of Zhang *et al* (2020), which uses the OOB residuals to compute an unconditional prediction interval. We refer to this method as 'OOB constant'. 50 trials were run, each involving a different draw of training/test data. Models were trained with 64 bags. Error bars are one standard error.

We see that the OOB constant method is highly over-confident—the true residual is on average more than twice as large as the 1-σ error bar. The recalibrated bootstrap is also over-confident but significantly less so. Interestingly the MNLPD values are equivalent, highlighting the fact that MNLPD must be considered as one of several metrics. The recalibration method of Palmer *et al* (2022) essentially minimizes NLPD, and hence may also have trouble with imbalanced data.

D.3. Recalibration factors

Figure 8 shows the distribution of standard residuals and recalibration factors (as in figure 1) for four more test problems: Friedman–Grosse without noise, Friedman–Silverman, the elongation output of the mechanical properties data set, and the ZT output of the thermoelectrics data set.

We see that the other synthetic problems are well calibrated but also have some high-residual outliers. The real-world data sets are skewed and more sharply peaked.

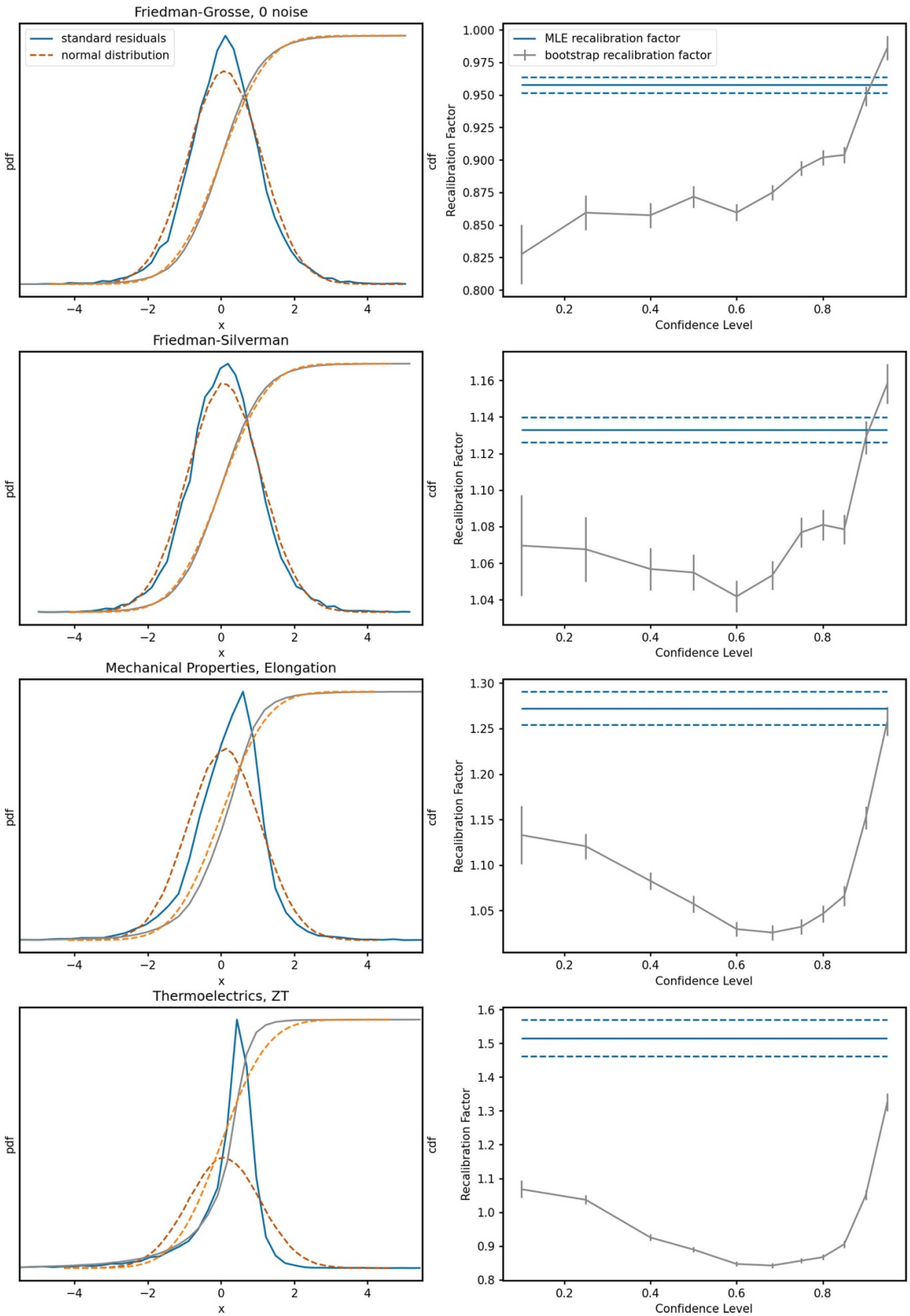

Figure 8. Investigation of the standard OOB residuals and recalibration factors for several test problems.

D.4. Prediction interval metrics

Figure 9 shows the univariate prediction interval metrics (as in figure 2 for four more test problems: Friedman–Grosse without noise, Friedman–Silverman with noise of magnitude 2.0, the Young's modulus output of the mechanical properties data set, and the ZT output of the thermoelectrics data set.

Figure 10 shows MNLPD for the multivariate Friedman–Grosse test problem with 128 test and training points, varying the number of bags. The number of bags is largely irrelevant. The fact that a small, constant

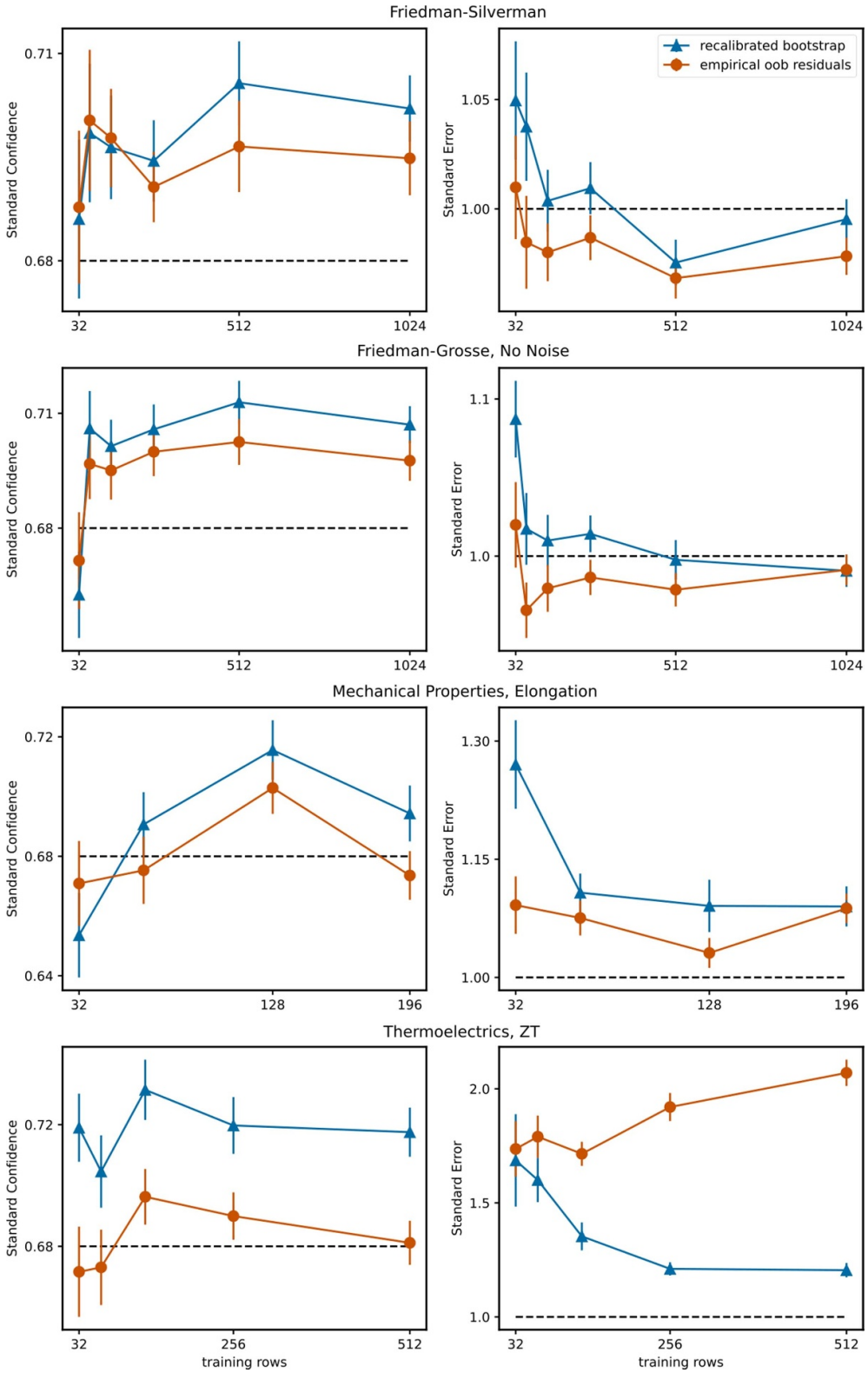

Figure 9. Univariate prediction interval metrics for several test problems, computed using the recalibrated bootstrap method proposed here and the 'empirical OOB residuals' method of Zhang *et al* (2020). Synthetic problem have 128 test points; real data test problems have 64 test points. All results are averaged over 64 trials. Each trial involves a different set of training and test data. Error bars are 1 standard error.

number of bags suffices to create a well-calibrated prediction interval is one of the benefits of the recalibrated bootstrap method.

Figure 11 shows the effect of varying the noise level for the Friedman–Grosse test problem. For a single output we consider the size of the mean 1-σ prediction interval generated by the recalibrated bootstrap, divided by the noise level. Ideally this would approach 1 as the noise becomes dominant. Instead it

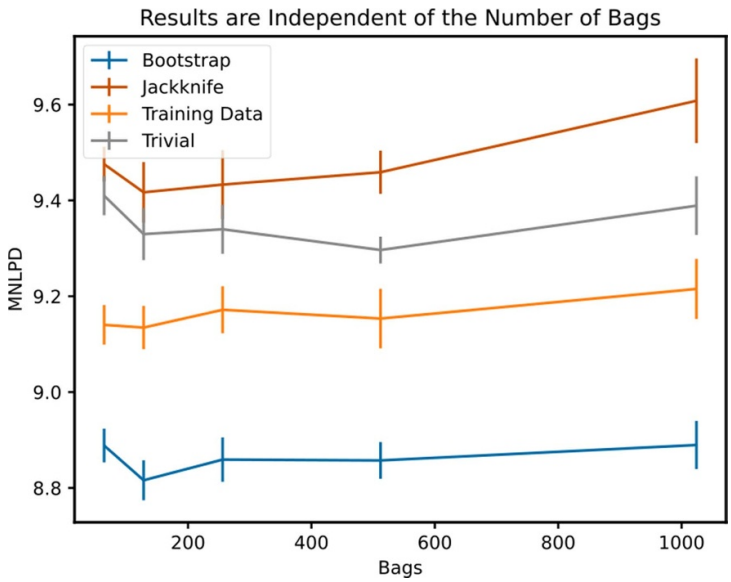

Figure 10. Increasing the number of bags does not change the relative accuracy of the correlation estimation methods. This test is on the Friedman–Grosse function with noise level 1.0, 128 training and test points, and 16 trials. Error bars are one standard error.

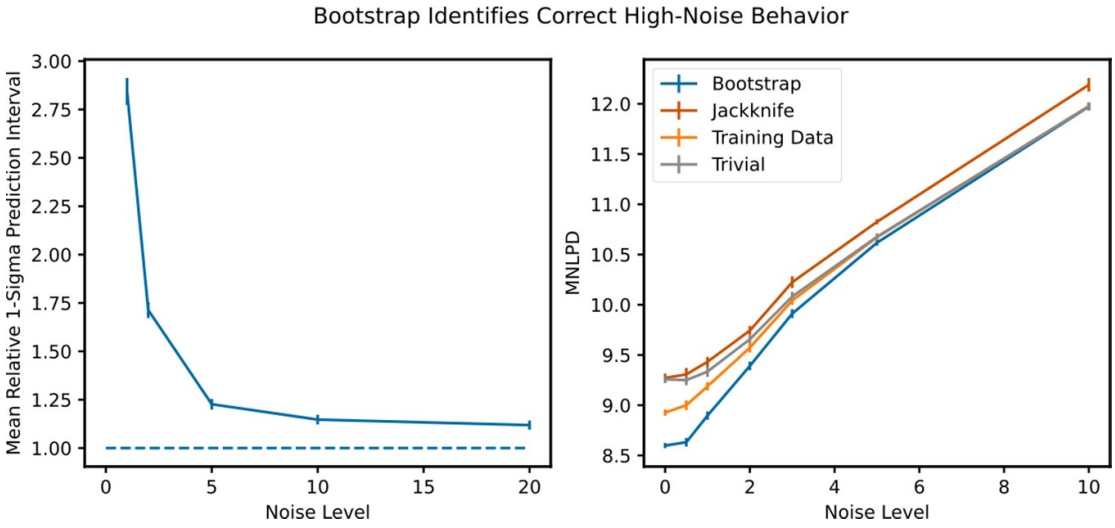

Figure 11. (a) Recalibrated bootstrap prediction interval and (b) MNLPD as the noise level varies for the Friedman–Grosse test problem. There are 128 training points, 64 bags, and 100 trials. Error bars are one standard error.

approaches 1.1. The standard residual (not shown) approaches 0.9, so we see that while the recalibrated bootstrap largely picks up on the noise it is not quantitatively exactly correct.

For the multi-output problem we see that the recalibrated bootstrap and trivial methods of estimating correlation converge to the same result in the high-noise limit. Since the noise is generated independently, the correlation coefficients of the prediction distribution should be uniformly 0 in this limit. The trivial method is therefore correct, and the fact that the recalibrated bootstrap method converges to the same result indicates that it is hitting upon this correct answer as well.

D.5. Impact of confidence level parameter on SL

Table 3 shows the impact of varying the confidence level parameter in the recalibrated bootstrap algorithm. Changing the value of p can change the recalibration factor and lead to a slight uniform rescaling of the prediction intervals. For the synthetic problem based on the Friedman–Grosse function (appendix A.3), we know from figure 1(b) that setting $p = 0.95$ generally leads to a larger recalibration ratio and wider prediction intervals. However this is not seen to impact the results of simulated SL.

Table 3. Mean number of iterations required to find a satisfactory candidate in the synthetic sequential learning problem, comparing two values of the confidence level parameter from algorithm 1. Uncertainty values are one standard error computed over 64 independent trials.

	$p = 0.683$ (1-σ)	$p = 0.95$ (2-σ)
Trivial	36.0 ± 2.6	35.1 ± 2.8
Training data	41.8 ± 3.1	41.7 ± 2.9
Jackknife	11.2 ± 1.5	11.9 ± 1.6
Bootstrap	7.5 ± 1.0	7.5 ± 1.0

ORCID iD

Brendan Folie https://orcid.org/0000-0002-0844-4111

References

Abroshan H, Chandrasekaran A, Winget P, An Y, Kwak S, Brown C and Halls M D 2021 Accelerated design and optimization of novel OLED materials via active learning *Proc. SPIE* **11808**

Antono E, Matsuzawa N N, Ling J, Saal J E, Arai H, Sasago M and Fujii E 2020 Machine-learning guided quantum chemical and molecular dynamics calculations to design novel hole-conducting organic materials *J. Phys. Chem.* A **124** 8330–40

Attia P M *et al* 2020 Closed-loop optimization of fast-charging protocols for batteries with machine learning *Nature* **578** 397–402

Awal A, Masud M, Hossain S, Al-Mamun Bulbul A, Mahmud H and Kumar Bairagi A 2021 A novel Bayesian optimization-based machine learning framework for COVID-19 detection from inpatient facility data *IEEE Access* **9** 10263–81

Barber R F, Candes E J, Ramdas A and Tibshirani R J 2021 Predictive inference with the jackknife+ *Ann. Stat.* **49** 486–507

Borg C K H, Frey C, Moh J, Pollock T M, Gorsse S, Miracle D B, Senkov O N, Meredig B and Saal J E 2020 Expanded dataset of mechanical properties and observed phases of multi-principal element alloys *Sci. Data* **7** 430

Carrella E 2021 No free lunch when estimating simulation parameters *J. Artif. Soc. Soc. Simul.* **24** 7

Chandak A, Dey D, Mukhoty B and Kar P 2020 Epidemiologically and socio-economically optimal policies via Bayesian optimization *Trans. Indian Natl Acad. Eng.* **5** 117–27

Dave A, Mitchell J, Kandasamy K, Wang H, Burke S, Paria B, Póczos B, Whitacre J and Viswanathan V 2020 Autonomous discovery of battery electrolytes with robotic experimentation and machine learning *Cell Rep. Phys. Sci.* **1** 100264

del Rosario Z, Rupp M, Kim Y, Antono E and Ling J 2020 Assessing the frontier: active learning, model accuracy and multi-objective candidate discovery and optimization *J. Chem. Phys.* **153** 024112

Dewolf N, De Baets B and Waegeman W 2023 Valid prediction intervals for regression problems *Artif. Intell. Rev.* **56** 577–613

Efron B 1982 The jackknife, the bootstrap and other resampling plans *The Society for Industrial and Applied Mathematics* (Philadelphia, PA: Society for Industrial and Applied Mathematics)

Efron B 2014 Estimation and accuracy after model selection *J. Am. Stat. Assoc.* **109** 991–1007

Fakhrmoosavi F, Kamjoo E, Kavianipour M, Zockaie A, Talebpour A and Mittal A 2022 A stochastic framework using Bayesian optimization algorithm to assess the network-level societal impacts of connected and autonomous vehicles *Transp. Res.* C **139** 103663

Fong A Y, Pellouchoud L, Davidson M, Walroth R C, Church C, Tcareva E, Wu L, Peterson K, Meredig B and Tassone C J 2021 Utilization of machine learning to accelerate colloidal synthesis and discovery *J. Chem. Phys.* **154** 224201

Friedman J H, Grosse E and Stuetzle W 1983 Multidimensional additive spline approximation *SIAM J. Sci. Statist. Comput.* **4** 291–301

Friedman J H and Silverman B W 1989 Flexible parsimonious smoothing and additive modeling *Technometrics* **3** 3–39

Gaultois M W, Sparks T D, Borg C K H, Seshadri R, Bonificio W D and Clarke D R 2013 Data-driven review of thermoelectric materials: performance and resource considerations *Chem. Mater.* **25** 2911–20

Ghosal I 2021 Model combinations and the infinitesimal jackknife: how to refine models with boosting and quantify uncertainty *PhD Thesis* (Cornell University)

Hutchinson M 2016 Lolo (available at: https://github.com/CitrineInformatics/lolo)

Kim B, Xu C and Barber R F 2020 Predictive inference is free with the jackknife+-after-bootstrap *NeurIPS* **34**

Kuchibhotla A K and Berk R A 2023 Nested conformal prediction sets for classification with applications to probation data *Ann. Appl. Stat.* **17** 761–85

Lei J, G'Sell M, Rinaldo A, Tibshirani R J and Wasserman L 2018 Distribution-free predictive inference for regression *J. Am. Stat. Assoc.* **113** 1094–111

Lepioufle J-M, Marsteen L and Johnsrud M 2021 Error prediction of air quality at monitoring stations using random forest in a total error framework *Phys. Sens.* **21** 2160

Ling J, Hutchinson M, Antono E, Paradiso S and Meredig B 2017 High-dimensional materials and process optimization using data-driven experimental design with well-calibrated uncertainty estimates *Integr. Mater. Manuf. Innov.* **6** 207–17

Liu R, Jiang Y and Lima Azevedo C 2021 Bayesian optimization of area-based road pricing *Int. Conf. on Models and Technologies for Intelligent Transportation Systems* vol 7

Liu Z, Rolston N, Flick A C, Colburn T W, Ren Z, Dauskardt R H and Buonassisi T 2022 Machine learning with knowledge constraints for process optimization of open-air perovskite solar cell manufacturing *Joule* **6** 834–49

Lu B and Hardin J 2021 A unified framework for random forest prediction error estimation *J. Mach. Learn. Res.* **22** 8

Marcotte D 2012 Revisiting the linear model of coregionalization *Geostatistics Oslo* **17** 67–78

Mathern A, Steinholtz O S, Sjöberg A, Önnheim M, Ek K, Rempling R, Gustavsson E and Jirstrand M 2021 Multi-objective constrained Bayesian optimization for structural design *Struct. Multidiscip. Optim.* **63** 689–701

Meinshausen N 2006 Quantile regression forests *J. Mach. Learn. Res.* **7** 983–99

Mentch L and Hooker G 2016 Quantifying uncertainty in random forests via confidence intervals and hypothesis tests *J. Mach. Learn. Res.* **17** 1–41

Meredig B *et al* 2018 Can machine learning identify the next high-temperature superconductor? examining extrapolation performance for materials discovery *Mole. Syst. Des. Eng.* **3** 819–25
Palmer G, Du S, Politowicz A, Emory J P, Yang X, Gautam A, Gupta G, Li Z, Jacobs R and Morgan D 2022 Calibration after bootstrap for accurate uncertainty quantification in regression models *npj Comput. Mater.* **8** 115
Rohr B, Stein H S, Guevarra D, Wang Y, Haber J A, Aykol M, Suram S K and Gregoire J M 2020 Benchmarking the acceleration of materials discovery by sequential learning *Chem. Sci.* **11** 2696–706
Roman D, Saxena S, Robu V, Pecht M and Flynn D 2021 Machine learning pipeline for battery state-of-health estimation *Nat. Mach. Intell.* **3** 447–56
Ruesch A, Yang J, Schmitt S, Acharya D, Smith M A and Kainerstorfer J M 2020 Estimating intracranial pressure using pulsatile cerebral blood flow measured with diffuse correlation spectroscopy *Biomed. Opt. Express* **11** 1462
Russell R L and Reale C 2022 Multivariate uncertainty in deep learning *IEEE Trans. Neural Netw. Learn. Syst.* **33** 7937–43
Shahriari B, Swersky K, Wang Z, Adams R P and de Freitas N 2016 Taking the human out of the loop: a review of Bayesian optimization *Proc. IEEE* **104** 148–75
Verduzco J C, Marinero E E and Strachan A 2021 An active learning approach for the design of doped LLZO ceramic garnets for battery applications *Integr. Mater. Manuf. Innov.* **10** 299–310
Wager S, Hastie T and Efron B 2014 Confidence intervals for random forests: the jackknife and the infinitesimal jackknife *J. Mach. Learn. Res.* **15** 1625–51
Wahab H, Jain V, Tyrrell A S, Seas M A, Kotthoff L and Johnson P A 2020 Machine-learning-assisted fabrication: Bayesian optimization of laser-induced graphene patterning using in-situ Raman analysis *Carbon* **167** 609–19
Wang Z, Gehring C, Kohli P and Jegelka S 2018 Batched large-scale Bayesian optimization in high-dimensional spaces *Proc. Mach. Learn. Res.* **84** 745–54
Ward L *et al* 2018 Matminer: an open source toolkit for materials data mining *Comput. Mater. Sci.* **152** 60–69
Ward L, Agrawal A, Choudhary A and Wolverton C 2016 A general-purpose machine learning framework for predicting properties of inorganic materials *npj Comput. Mater.* **2** 16028
Zhang H, Zimmerman J, Nettleton D and Nordman D J 2020 Random forest prediction intervals *Am. Statistician* **74** 392–406
Zhang Y-M, Wang H, Mao J-X, Xu Z-D and Zhang Y-F 2021 Probabilistic framework with Bayesian optimization for predicting typhoon-induced dynamic responses of a long-span bridge *J. Struct. Eng.* **147** 04020297