# Assessing Uncertainty in Performance Modeling: Conformal Prediction vs. Bayesian Regression

Stefan Jahns
Johannes Dorn
Max Weber
Leipzig University
Leipzig, Germany

Sven Apel
Saarland University
Saarbrücken, Germany

Norbert Siegmund
ScaDS.AI
Dresden/Leipzig, Germany
Leipzig University
Leipzig, Germany

## Abstract

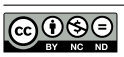
Most software systems today are configurable. As a consequence, predicting performance is inherently challenging due to the sheer number of configurations exhibiting individual performance variations. When predicting the performance of unseen configurations, it is imperative to know how reliable such predictions are. Bayesian methods have been shown to be able to quantify the uncertainty in their estimates, but require prior information about performance distributions and, more importantly, are constrained in their learning technique, effectively limiting their practical applicability. Conformal prediction, an entirely novel approach to uncertainty quantification in this area, augments existing learning algorithms via a statistical post-hoc analysis. This allows us to provide uncertainty estimates even for point-estimate algorithms, such as random forests or deep neural networks. The goal of this study is to evaluate how assumptions of conformal predictions regarding sample size and distribution, as well as the heterogeneity of learning algorithms, affect their applicability to software performance prediction. We conduct a comprehensive study comparing four conformal prediction variants against P4, the current standard for uncertainty-aware performance prediction, analyzing accuracy, scalability, and uncertainty quantification. Our results show that conformal prediction consistently outperforms the state of the art across six out of eight real world scenarios, producing correct confidence intervals that are 46–75 % narrower, and thus more certain. Notably, conformal prediction can be used to enrich existing techniques, with post-hoc uncertainty quantification in a plug and play fashion.

## CCS Concepts

• **Software and its engineering** → *Software product lines*; **Software performance**; • **Computing methodologies** → **Machine learning approaches**.

## Keywords

Conformal Prediction, Configurable Software, Performance Prediction, Uncertainty




*ASE '26, Munich, Germany*

ACM ISBN 979-8-4007-2882-2/2026/10
https://doi.org/10.1145/3832783.3837425

**ACM Reference Format:**
Stefan Jahns, Johannes Dorn, Max Weber, Sven Apel, and Norbert Siegmund. 2026. Assessing Uncertainty in Performance Modeling: Conformal Prediction vs. Bayesian Regression. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering (ASE '26), October 12–16, 2026, Munich, Germany*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3832783.3837425

## 1 Introduction

Understanding how configuration options influence the performance of software systems is essential for optimizing performance, reducing energy consumption, and improving user satisfaction [46]. As configuration spaces are typically huge, current methods measure a subset of configurations to train models that estimate the performance of unseen configurations. Usually, these estimates produce scalar values; that is, single-point performance estimates with no indication of spread or confidence [7]. While these estimations are accurate under ideal conditions (e.g., multiple, interference-free measurements), they lack the capability for uncertainty quantification, which provides a measure of the confidence or reliability of the performance estimates [31]. In real-world scenarios, incorporating uncertainty is crucial as it enhances transparency and enables more robust decision-making when dealing with imperfect or multi-modal data [7, 38].

Consider a database system for which we can configure its query optimization strategy. One strategy might lead to a bimodal performance distribution: It executes queries either fast or slow, depending on the cache hit rate. A scalar prediction would report an average of both (or just one) execution-time behavior, which, in this example, is incorrect and possesses the risk of extreme slowdowns. Uncertainty quantification would reveal this variance (as shown in Figure 1), helping users avoid strategies prone to unsteady performance.

Bayesian methods, such as P4 [7, 8], have been proposed for incorporating uncertainty quantification into software performance modeling. These methods produce predictive distributions, offering detailed insights into uncertainty, for example, whether or not a given query optimization strategy is likely to result in consistent performance. However, Bayesian methods come with notable limitations: They heavily depend on a (user-given) prior distribution, representing initial beliefs about option–performance relationships, which are often unknown or difficult to estimate accurately, potentially leading to overconfident predictions. [9, 44]. Moreover, Bayesian methods often require modifications to existing models. For instance, in neural networks, they may necessitate changes to

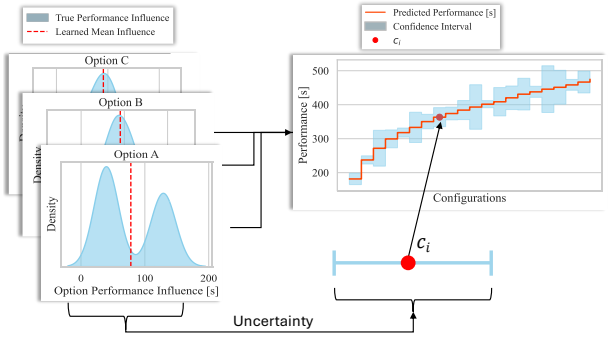



**Figure 1: System analysis. Left: True influences of configuration options (density plots; mean as dotted red line). Right: Performance predictions (red line) with confidence intervals (blue area) containing true values with certain probability.**

the activation function [9], which could affect its predictive performance. Some models, such as random forests, which have shown high accuracy in software performance modeling [14], are incompatible with Bayesian methods without extensive modifications. So, it seems one must decide between the advantages of uncertainty quantification and the most accurate models. The premise of our work is that, with conformal prediction, we can have both.

In this paper, we explore whether *conformal prediction*, a technique well-established in medical diagnosis [42] and financial forecasting [5], is able to address the challenges of uncertainty quantification in performance modeling of software configuration spaces. Conformal prediction leverages the law of large numbers to provide valid uncertainty estimates without requiring prior distributions. Crucially, it operates as a post-hoc method of any existing prediction model, preserving the model's accuracy while adding uncertainty quantification. That is, conformal prediction can be used on top of well-established models, such as random forests [4] and neural networks [30], not replacing but augmenting them.

Although conformal prediction has been used in several domains, it is unclear whether it can be applied to performance modeling of software configuration spaces, as data is high-dimensional and sample sizes are low due to high measurement effort. Specifically, we are interested in whether conformal prediction is able to provide reliable uncertainty estimates in real-world settings. As conformal prediction is an umbrella of different methods for the task of post-hoc uncertainty quantification, it is important to empirically investigate which of these methods is applicable to this domain.

We are the first to introduce this concept to performance modeling and, thereby, address the necessary steps to rate its applicability: We conduct a comprehensive evaluation of conformal prediction and compare it to Bayesian methods. To this end, we use a combination of synthetic datasets designed to simulate high-dimensional, low-sample scenarios, as well as real-world performance measurement data derived from a diverse set of configurable software systems. We test the applicability of post-hoc uncertainty quantification with traditional scalar models, such as Lasso regression, and advanced models, such as random forests and neural networks. We compare Bayesian and conformal prediction methods to examine potential performance trade-offs under varying conditions, particularly with respect to prediction accuracy and the quality

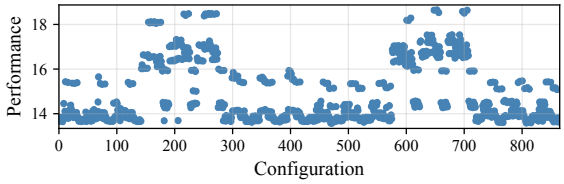



**Figure 2: Characteristics of performance data, illustrated using HSQLDB. Configurations mapped to one dimension; energy consumption (over fixed time) as metric.**

of uncertainty quantification. To better understand the practical applicability of uncertainty quantification techniques, we introduce various forms of uncertainty into our performance measurements to simulate real-world conditions and thereby assess the robustness of conformal prediction in realistic environments with unavoidable performance variance.

Our experiments demonstrate that conformal prediction is a viable approach for augmenting performance modeling of configurable software systems. In six out of eight real world scenarios, it outperforms the P4 model and achieves comparable results in the remaining ones. Among the conformal prediction methods, split-localized conformal prediction performs best across various learners. If the learner can be chosen, the combination of cross-validation+ with a random forest regressor yields the strongest results. It achieves a 60 % improvement in prediction accuracy and increases confidence level to 0.91, compared to P4's over-conservative 0.99 (target: 0.8), while having 70 % narrower intervals, capturing uncertainty more effectively than P4.

This paper makes the following key contributions:

- **Conformal Prediction**: We introduce conformal prediction as a practical, model-agnostic solution for uncertainty-aware software performance modeling.
- **Uncertainty Quantification**: We evaluate the effectiveness of uncertainty quantification methods for configurable software systems, highlighting their practical applicability and reliability.
- **Method Selection**: We identify fundamental trade-offs between Bayesian methods and conformal prediction with respect to prediction accuracy and uncertainty quantification quality, and derive actionable recommendations for their use.
- **Reproducibility**: We provide a comprehensive replication package with datasets, code, and experiment scripts.

## 2 Related Studies and Problem Statement

While there is substantial work on performance modeling of configurable software systems, quantifying the uncertainty of performance predictions has been addressed only by P4 [7]. P4 is a Bayesian linear regression approach that employs probabilistic programming with Lasso-based feature selection. Although P4 demonstrates advantages, specifically its capability to automatically approximate prior distributions, prediction accuracy is often below that of scalar models, such as random forests or deep neural networks.

Given the trade-off between accurate uncertainty quantification and predictive accuracy, we investigate alternative approaches. Conformal prediction is an alternative that is agnostic to the learning

technique and, thus, does not affect prediction accuracy. Conformal prediction has been compared against Bayesian modeling in other contexts (e.g., UCI datasets [6]) with superior results in most cases [6, 19, 45]. However, none of these studies address the characteristics of performance data and come even close to the number of input dimensions (i.e., configurations in our case), limiting the transferability of their findings. To clarify this, let us compare the characteristics of prior work with those of configurable software systems. Dewolf et al. [6] use 80% of all data for training, a proportion that is infeasible for configurable software systems with a humongous configuration space. Due to high cost of measuring a single configuration, the available training set typically represents only a small fraction of the full population.Additionally, the study neglects marginal coverage (see Section 5.5.2), an essential metric for evaluating its usefulness in performance modeling. Wang et al. [45] apply conformal prediction to housing data and other non-computer science datasets, whose distributional properties differ substantially from performance measurements. Lei et al. [19] do not evaluate conformal prediction on sparse datasets, a defining characteristic of performance measurements in configurable systems [10, 13] which we will explain next. Consequently, despite existing research on conformal prediction, it remains unclear whether specific conformal prediction techniques can be effectively applied to performance prediction of software configurations.

The limited empirical evidence from prior work becomes clear when examining the distinctive characteristics of performance data (see Figure 2). First, performance data are sparse: given the exponential number of how configuration options can be combined, only a small fraction can be practically measured [10, 13]. Second, these data are high-dimensional due to the large number of configuration options. which may be numeric or binary and can exhibit strong interactions [18, 35]. Importantly, despite this high dimensionality, configuration options often exhibit substantial collinearity [7, 8]. This complicates attributing performance influences on individual options and interactions and can degrade certain learning algorithms, and locality-based conformal-prediction approaches. The extent of this effect remains unknown due to limited empirical evidence. Third, performance influences are often locally clustered [10, 23, 37], exhibit a stair-like discrete performance landscape distinct from other settings [25]. Finally, measurement data are considerably noisier [40] than, for instance, the housing data commonly used to evaluate conformal prediction. This motivates a clear need for the empirical assessment of how and when conformal prediction can be applied to performance data. Consequently, uncertainty quantification techniques in this setting must be robust to sparse and noisy data, while being still producing valid confidence intervals across a high-dimensional and non-continuous landscape.

Concretely, the aforementioned characteristics pose several challenges to conformal prediction. They violate its core assumption of a single nonconformity distribution [43], as different regions of the configuration space can exhibit distinct performance regimes [37]. Methods such as localized conformal prediction address this issue, but they rely on sufficient data density in local regions [11], which is limited here due to measuring costs. In addition, collinearity among configuration options can further complicate modeling, as it is known to require specialized treatment for reliable uncertainty quantification in Bayesian approaches [7, 8] may indirectly affect conformal methods by degrading residual quality or local neighborhood structure. Furthermore, the stair-like and discontinuous response surface of configurable systems leads to non-smooth, heterogeneous residual distributions, reducing conformal interval efficiency and resulting in wider intervals for the same coverage guarantee [19, 28, 32].

While limited aspects of these challenges have been partially examined in isolation (e.g., [11, 28, 32]), their combined impact has not yet been systematically investigated, nor rigorously compared to Bayesian approaches, despite their central importance for practical uncertainty estimation in performance prediction for configurable software systems. Moreover, there is no empirical evidence on the sample sizes needed to effectively use conformal prediction. We therefore empirically assess its effectiveness in this data regime and evaluate its potential as an alternative to Bayesian modeling, which often struggles to maintain point prediction accuracy while incorporating uncertainty quantification in performance modeling.

# 3 Conformal Prediction

## 3.1 Overview

Conformal prediction (CP) is a statistical framework that can quantify the uncertainty of point estimates from arbitrary learning algorithms [33]. It does so by computing a confidence interval (I), within which the ground truth value lies with a certain probability. Formally, the probability that the ground truth value $v$ is within the confidence interval is:

$$\mathrm{P}(v \in \mathrm{I}) \geq 1 - \alpha$$

where $\alpha$ specifies the user-chosen error rate. For instance, setting $\alpha = 0.05$ would lead to a computed interval range such that the true value will be within this interval with a 95 % probability.

We define $C$ as the set of valid configurations of a configurable software system. A learning algorithm $l$ uses a training set $d$ to produce a prediction model $m_l^d$. The elements of the training set are tuples $(c, v)$ where $c \in C$ and $v$ is the performance value of the corresponding configuration determined by measurement. For instance, we can use Lasso regression for learning $m_{\text{Lasso}}^d = \text{Lasso}(d)$. Since $m_{\text{Lasso}}$ can predict only scalar values $\hat{y} = m_{\text{Lasso}}(c)$, we have no quantitative information about their correctness.

Conformal prediction augments a point estimate with a confidence interval derived from a trained conformal prediction model (CPM). This model estimates uncertainty based on how well it has predicted previously seen configurations. To separate the role of the data, whether they are used for training the prediction model or for deriving confidence, we can choose from different conformal prediction methods. A method defines how data are split between training the prediction model $m_l$ and computes conformal scores $s$. Depending on the method, we obtain confidence intervals with varying accuracy. There are different approaches to compute the conformal scores, which further distinguishes the different conformal methods. Here, we select a non-conformity measure $\eta$. In a nutshell, the non-conformity measure receives a set of configurations, their measured performance values, and the prediction model, and outputs a score that indicates how well a prediction fits the rest of the data. Choosing a non-conformity measure that fits the setting is crucial for accuracy [33]. The trained conformal

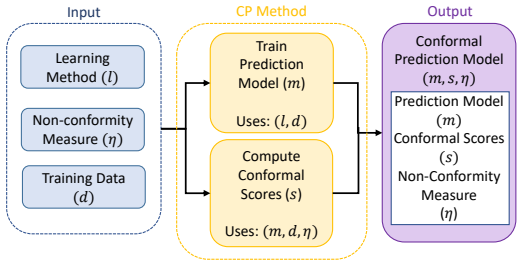



**Figure 3: Training pipeline of a conformal prediction model, illustrating inputs, processing steps, and model components.**

prediction model consists of the prediction model, conformal scores, and the non-conformity measure. Figure 3 illustrates its pipeline.

Algorithm 1 depicts how we predict a confidence interval for a new configuration. It takes as input a configuration $c$ for which it should estimate its performance and a fitting confidence interval. As the first step, it predicts the scalar value $\hat{y}$ based on the predictive model of the provided conformal prediction model. It then computes the $\alpha$-quantile threshold $s_\alpha$ from the scores $s$ provided by the conformal prediction model and the user-defined parameter $\alpha$. To construct a confidence interval, we determine which true values would yield a non-conformity score less than or equal to $s_\alpha$. This is achieved by applying the inverse non-conformity measure ($\eta^{-1}$) to $m$, $c$ and $s_\alpha$ resulting in an interval that contains all such values – this interval is the predicted confidence interval ($\mathrm{I}_c$).

The output of the inverted non-conformity measure is the interval containing all values whose conformity scores are less than or equal to $s_\alpha$ for the prediction value $m(c)$ [33]. For example, if the non-conformity measure is the absolute distance between prediction $m(c)$ and measured value $v$ (see Equation 1) and if we calculate the confidence interval for $m(c) = 5$ with $s_\alpha = 2$, the resulting confidence interval would be $[3, 7]$, as all values $v$ within this interval would yield a conformity score less than or equal to 2.

**Algorithm 1** Prediction

1: **Input:** CPM $(m, s, \eta)$, new configuration $c$, confidence level $\alpha$
2: $\hat{y} \leftarrow m(c)$
3: $s_\alpha \leftarrow \text{quantile}(s, \alpha)$
4: $\mathrm{I}_c \leftarrow \eta^{-1}(s_\alpha, c, m)$
5: **return** $\hat{y}, \mathrm{I}_c$

## 3.2 Conformal Prediction Methods

The conformal prediction framework provides four methods that are commonly used in regression settings: Conformalized quantile regression, split conformal prediction, jackknife+, and cross-validation+ [21]. The conformalized quantile regression works only for models that already produce confidence intervals and are therefore not feasible for our setting, where we want to add uncertainty quantification to models that are already used for configurable software systems. Nevertheless, as it represents a promising alternative to Bayesian modeling, we include it in our evaluation as well. The jackknive+ is often seen as a special case of cross-validation+ with similar results but a higher computational cost [21]. So, we evaluate the remaining two methods: *split conformal prediction* [26] and *cross-validation+* [2, 43]. Split conformal prediction is computationally efficient, and cross-validation+ utilizes all available data for prediction and calibration of the confidence intervals [20].

### 3.2.1 *Split Conformal Prediction.*

The *split conformal prediction* method (Algorithm 2) employs a split function to partition the training set $d$ into two non-empty, disjoint sets: $d_{\text{learn}}$ and $d_{\text{conf}}$. In our experiment, we perform a random shuffle and then split the data using a fixed split factor. The first set, $d_{\text{learn}}$, is used to train the prediction model $m$. After training, we use the model and the set $d_{\text{conf}}$ to calculate the conformity scores based on the selected non-conformity measure.

**Algorithm 2** Split Conformal Prediction

1: **Input:** Learning method $l$, data $d$, non-conformity measure $\eta$, split function $f$
2: $(d_{\text{learn}}, d_{\text{conf}}) \leftarrow f(d)$
3: $m \leftarrow l(d_{\text{learn}})$
4: $s \leftarrow \eta(m, d_{\text{conf}})$
5: **return** $m, s, \eta$

### 3.2.2 *Cross-Validation+.*

The method *cross-validation+* (Algorithm 3) uses a $k$-fold strategy to partition the training data $d$ into $k$ disjoint subsets of equal size. For each subset $d_i \subset d$, we train a prediction model $m_i$ using the remaining $k-1$ subsets $d \setminus d_i$. Next, we use prediction model $m_i$ and subset $d_i$ to calculate conformity scores $s_i$ with the selected non-conformity measure. This process results in $k$ prediction models and $k$ sets of conformity scores. Finally, a model $m_0$ is trained on the entire dataset $d$, and the models and scores are combined into a tuple to produce the final prediction model $m$ and the combined conformity scores $s$ for the conformal prediction model. The final model $m$ acts as a single model ($m_0$) when predicting the scalar value $\hat{y}$, but as $k$ separate models when computing the inverse non-conformity measure, yielding $k$ different predictions $m_{1..k}(c)$. The confidence interval is constructed based on these predictions and the distribution of conformity scores. Specifically, we use the $\alpha$-smallest prediction to calculate the lower bound and the $\alpha$-largest prediction to calculate the upper bound of the final confidence interval.

**Algorithm 3** Cross-validation+

1: **Input:** Learning method $l$, data $d$, non-conformity measure $\eta$, number of folds $k$,
2: $(d_1..d_k) \leftarrow k\text{-fold}(d)$
3: **for** $i = 1$ to $k$ **do**
4: $\quad m_i \leftarrow l(d \setminus d_i)$
5: $\quad s_i \leftarrow \eta(m_i, d_i)$
6: **end for**
7: $m_0 \leftarrow l(d)$
8: $m \leftarrow (m_0, m_1, ..., m_k)$
9: $s \leftarrow \bigcup_1^k (s_i)$
10: **return** $m, s, \eta$

## 3.3 Non-Conformity Measure

We use two non-conformity measures in our experiments: the absolute residual score and a localized version of it [11]. The first is the simplest non-conformity measure and is used as a baseline for our setup; the second one allows for capturing uncertainties induced by individual options because it takes the configurations into account and not just the predicted values. The *absolute score* is defined as the absolute difference between the predicted value and the measured one:

$$\text{absolute-score}\,(m, (c, v)) := |m(c) - v|, \tag{1}$$

whereas the *localized score* uses an auxiliary model $\hat{\sigma}$ trained to predict the residuals of $m$ for normalization. $\hat{\sigma}$ is trained on a subset of $d_{\text{conf}}$ and the corresponding predictions of $m$:

$$\text{localized-score}\,(m, (c, v)) = \frac{|m(c) - v|}{\hat{\sigma}(c)}. \tag{2}$$

The absolute score is straightforward and robust but results in confidence intervals of equal width for all predictions when using the split method, limiting potential improvements in interpretability and ignoring the nature of software performance values, which may span from milliseconds of execution time to minutes and hours for the same software system, depending on the configuration. The localized score accounts for the distance between configurations, offering variations in the intervals. However, it has two drawbacks: It can be used only with split conformal prediction due to the additional model $\hat{\sigma}$; moreover, inaccuracies in $\hat{\sigma}$ predictions can reduce the accuracy of the resulting confidence intervals.

### 3.4 Guarantees

Conformal prediction as a statistical framework has been proved to provide valid confidence intervals as long as two assumptions hold: exchangeability and an infinite data set [43]. *Exchangeability* is a relaxation of the commonly used assumption of independent and identically distributed random variables [33]. It assumes that the joint probability distribution of a dataset remains unchanged under any permutation of the indices of its items. Regarding the second assumption, an infinite data set is required to achieve an exact interval coverage of $1 - \alpha$. However, since the coverage probability asymptotically approaches the target level as the data set size increases [1], a large enough dataset is sufficient. Whether the typically available sample set sizes of configurable software systems are sufficient for this approach remains unclear, which is why we study the applicability of conformal prediction.

## 4 Research Questions

In this section, we present our research questions addressing whether conformal prediction can be a useful tool for augmenting existing performance-prediction approaches.

**$RQ_1$: *Is conformal prediction feasible for performance modeling of configurable software systems?***

Our main goal is to assess whether and how reliable conformal prediction can be applied to performance modeling of software configurations, and at the same time enabling uncertainty quantification for a number of modeling approaches that are already existing. To demonstrate feasibility and address the breadth of this question, we formulate the following research objectives:
*Robustness to Data Limitations:* Since conformal prediction relies on the law of large numbers, a key challenge in applying it to software performance modeling is the typically small number of measurements available relative to the whole number of possible configurations. As a consequence, such data sets are sparse in nature; that is, we obtain only a few measurements per combination of options within this large configuration space.

**$RQ_{1.1}$** *How accurate are the confidence intervals generated by conformal prediction for real-world performance datasets of configurable software systems?*

*Consistency Across Configuration Options:* In software performance modeling, it is crucial to understand which options influence the system's overall performance. Transferred to our setting, we need to evaluate whether uncertainties induced by specific options are captured within the confidence intervals.

**$RQ_{1.2}$** *How consistent does conformal prediction capture uncertainties across different options in configurable software systems?*

*Informativeness of Confidence Intervals:* To be useful, predicted confidence intervals must be both accurate and informative, meaning they should be sufficiently narrow to support decision making. To assess what constitutes a sufficiently narrow interval, we evaluate interval width relative to the dispersion of the measurement data.

**$RQ_{1.3}$** *Does conformal prediction produce confidence intervals in a reasonable width in comparison to the predicted values?*

To answer $RQ_1$, we conduct experiments on both synthetic datasets and real-world systems, examining the impact of several variables on the evaluation metrics. Based on our research objectives, we vary the following five independent variables: software system, sample size, machine learning algorithm, conformal prediction method, and non-conformity measure. The metrics and specific values chosen for each variable are detailed in the experimental setup (see Section 5).

**$RQ_2$: *How do the selected conformal prediction methods compare to alternative approaches in terms of uncertainty quantification, prediction accuracy, and scalability?***

We divide the comparison into two parts: (1) P4, an established uncertainty quantification approach for performance modeling of configurable software systems, and (2) CQR, a conformal method that uses quantile prediction models rather than augmenting an existing point-estimate model.

Conformal prediction offers greater flexibility than Bayesian regression by being able to augment any given machine learning model. Additionally, split conformal prediction may provide scalability benefits, as its computational overhead increases linearly with the sample size. While our goal is to equip established performance models for scalar estimates with uncertainty quantification, we also consider interval-based conformal methods, such as conformalized quantile regression (CQR), as a possible alternative. This comparison separates the benefits of augmenting established models from those of using models for interval estimation.

We apply all approaches to the same systems and training sets used in $RQ_1$. For P4, confidence intervals come from the model's posterior predictive distribution, representing the probability of future observations given the training data. All resulting intervals are evaluated using the same correctness and usefulness metrics. We use three complementary metrics: (i) scalar prediction accuracy, capturing differences in the underlying machine learning models; (ii) probabilistic accuracy, assessing the combination of prediction precision and uncertainty quantification; and (iii) computational efficiency evaluating scalability with training set size (see Sec. 5.5).

## 5 Experiment Setup

This section describes the experimental setup used to answer our research questions. Each experiment is repeated 30 times to ensure statistical reliability and reduce the impact of randomness in training and evaluation.

### 5.1 Subject Systems

We evaluate our approach on ten software systems—two synthetic and eight real-world (see Table 1)—selected to cover a diverse range of characteristics and complexity levels.

**Table 1: Overview of subject systems. $|O_B|$: number of binary options; $|O_N|$: number of numeric options; $|C|$: number of valid configurations. $p_{opt}$ and $p_{flat}$ summarize the fitness landscape (local optima and plateau-like configurations in %), $L$ is the number of distinct fitness levels, $B_{max}$ the largest basin of attraction in %, COR the max pairwise correlation. Sample Size: number of samples used for the experiments.**

| System | # Options | | # Configs | Fitness-landscape | | | | | Sample Sizes | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\|O_B\|$ | $\|O_N\|$ | $\|C\|$ | $p_{opt}$ | $p_{flat}$ | $L$ | $B_{max}$ | COR | n | 2n | 5n | pw |
| x264 | 22 | 0 | 4608 | 8.51 | 0.33 | 434 | 2.19 | 0.5 | 22 | 44 | 110 | 101 |
| hsqldb | 18 | 0 | 864 | 9.38 | 10.8 | 210 | 7.52 | 1 | 18 | 36 | 90 | 73 |
| Apache | 19 | 0 | 581 | 3.45 | 1.38 | 237 | 5.52 | 1 | 19 | 38 | 95 | 48 |
| 7z | 11 | 3 | 68642 | 1.05 | 58.2 | 315 | 1.35 | 0.25 | 20 | 40 | 100 | 279 |
| nginx | 14 | 2 | 4417 | 5.19 | 11.4 | 84 | 2.76 | 1 | 20 | 40 | 100 | 213 |
| vp8 | 10 | 4 | 41040 | 0.19 | 1.43 | 612 | 6.32 | 0.9 | 22 | 44 | 110 | 140 |
| postgreSQL | 6 | 3 | 19008 | 0.31 | 0.00 | 610 | 7.05 | 0 | 15 | 30 | 75 | 144 |
| lrzip | 10 | 3 | 5184 | 1.79 | 60.7 | 179 | 9.16 | 0.2 | 19 | 38 | 95 | 243 |
| Synth. 1 | 19 | – | 2560 | 4.18 | 68.6 | 197 | 3.75 | 0.33 | 19 | 38 | 95 | 138 |
| Synth. 2 | 142 | – | $> 500k$ | – | – | – | – | – | 142 | 284 | 710 | – |

*5.1.1 Synthetic Systems.* The goal of a synthesized data set is twofold: First, we have a ground truth for which we can accurately assess whether conformal prediction aligns. Second, we can arbitrarily scale the data set in contrast to actual measurements.

We synthesize the data to be as close to realistic performance data as possible. To this end, we use configurations from real-world systems and define only the performance influences of the configuration options and their interactions. This allows us to introduce different types and severities of uncertainty:

- *5 % normally distributed noise across all values:* simulates measurement errors commonly found in real-world systems.
- *10 % normally distributed noise on a single option or interaction:* models uncertainty introduced by individual options.
- *10 % bimodal noise on a single option or interaction:* captures uncertainty from options with distinct performance modes.

Each type of option-specific uncertainty is applied to a quarter of the influencing options or interactions.

Configurations originate from the real-world system BerkeleyDBC and out of a general collection [39]. The small variant is BerkeleyDBC with $|O_B| = 19$. For its ground truth option influences, we use the values of related work [37]. The larger variant is a finance-decision system with $|O_B| = 142$, we generate its ground truth using *Thor*, a tool for generating option influences [37].

*5.1.2 Real-World Systems.* We selected eight real-world systems from prior work [7, 15, 47], covering four domains (i.e., video encoding, databases, web servers, data compression) with two software systems per domain to enhance the generalizability of our results. Table 1 gives an overview of these systems, including the sample sizes we use for our experiments and four fitness-landscape metrics.

$p_{opt}$ is the fraction of sampled configurations that are local optima, i.e., configurations whose fitness is not worse than that of any Hamming-distance-1 neighbor. Higher values of $p_{opt}$ indicate more locally peaked, rugged landscapes.

$p_{flat}$ is the fraction of plateau-like configurations whose neighbors are predominantly neutral, with fitness differences below a small tolerance. Higher $p_{flat}$ values point to large plateaus where many neighboring configurations have similar fitness.

$L$ is the number of distinct fitness levels after rounding fitness to a bin width of one hundredth of the global standard deviation. A larger $L$ means that the fitness values occupy more distinct levels at our chosen resolution, reflecting a finer-grained stair structure.

$B_{max}$ is the percentage of configurations that reach the largest optimum under greedy hill-climbing. A larger $B_{max}$ implies that one optimum attracts a substantial portion of the sampled configurations, whereas a smaller $B_{max}$ suggests that attraction basins are more fragmented across multiple optima.

As a collinearity diagnostic, we report the maximum absolute pairwise correlation between configuration options after removing exact duplicate option columns, which indicates how strongly any two remaining options move together and thus how much redundancy there is in the option space. Full reports are provided in the supplementary material (see Data Availability).

## 5.2 Sampling Size

To examine the impact of training data size $|d|$ on the accuracy of the confidence intervals and prediction values, we use true random sampling with four sizes: $n$, $2n$, $5n$ and pair-wise. The first three sizes are based on the number of configuration options n, following prior work [34]. To account for the complexity of numerical options, we weigh each numerical option by a factor of three—reflecting the common practice of sampling the minimum, maximum, and mean values [34]. For example, in PostgreSQL, which has 6 binary and 3 numerical options, we compute $n = 6 + 3 \cdot 3 = 15$.

The pair-wise size is derived using the T-wise sampling method [12] with an interaction depth of 2, commonly used to capture option interactions in configurable software systems. For numerical options, we follow the same count used in Plackett–Burman sampling [27] with $|O_N| + 1$ measurements.

## 5.3 Machine Learning Algorithms

We use three machine learning algorithms that have been used in prior work for performance predictions of software configurations:

- *Lasso regression:* an interpretable baseline [41].
- *Random forest:* a non-linear ensemble method [4].
- *DeepPerf:* a deep-learning model designed for software performance prediction [13].

as well as two machine learning algorithms that are feasible for conformalized quantile regression:

- *Linear quantile regression:* a linear model that estimates conditional quantiles using the pinball loss [17].
- *LightGBM:* a fast, non-linear gradient boosting method that handles feature interactions and supports quantile prediction [16].

These algorithms span a range of complexity and generalization capabilities, providing insights into how model architecture interacts with the conformal prediction. To ensure well-calibrated models, we apply hyperparameter optimization via random search [3] with cross-validation and 100 repetitions on the training set of each run.

## 5.4 Conformal Prediction Methods and Non-Conformity Measure Combinations

We evaluate several combinations of conformal prediction methods and non-conformity measures:

- *Split conformal prediction* with both the absolute-score (Equation 1) and the localized-score (Equation 2).
- *Cross-validation+* only with the absolute-score as, it is incompatible with the localized-score.
- *Conformalized quantile regression* uses its own score, which is adapted to quantile regressors [29].

We refer to the split conformal method using the absolute score as *split-absolute*, and the variant using the localized score as *split-localized*. For the split conformal prediction method, we evaluate five train-test split ratios: 0.25, 0.5, 0.75, 0.9, and 0.95. This allows us to analyze how the size of the calibration set influences the validity and efficiency of the predictions and the confidence intervals. In particular, machine learning algorithms that have no increase in accuracy with larger training sets size [24] may benefit from allocating more data to calibration. For the auxiliary model $\hat{\sigma}$ used in the localized-score, we employ a random forest regressor.

For cross-validaion+, we use three $k$-fold values: 10, 20 and $|d|$. The first is a standard choice of the Python library MAPIE [1], the second tests whether a fixed increase in $k$ improves performance, and the third generates a leave one out model for every training value, generating the maximum number of models.

## 5.5 Metrics

In this section, we present different metrics used to compare different uncertainty quantification methods in terms of accuracy, correctness, and usefulness, as well as the metrics to assess scalar prediction accuracy and scalability.

*5.5.1 Interval Coverage Rate.* We evaluate the accuracy of a uncertainty quantification method using the *interval coverage rate* (COV), a key metric for assessing the accuracy of confidence intervals. It measures the proportion of observed values $v$ that fall within the predicted intervals $\mathrm{I}_c$ for each configuration $c$. We use a validation set $d_{\text{val}}$ which contains all data points not used during training. Given a nominal coverage level of $(1-\alpha)$ (e.g., 90 % with $\alpha = 0.1$), an accurate method should yield a coverage rate close to this target on $d_{\text{val}}$. Overshooting this target (i.e., COV = 0.99) would mean that the interval is too broad, limiting the usability and information of the interval. Undershooting this target (i.e., COV = 0.5) would mean that the interval is too small, giving a false sense of certainty.

$$\mathrm{COV}(d, \mathrm{I}) = \frac{1}{|d|} \sum_{(c,v)\in d} \mathbb{1}\{v \in \mathrm{I}_c\},\ \mathrm{I}_c \in \mathrm{I} \tag{3}$$

*5.5.2 Feature-Stratified Coverage.* To evaluate the consistency of uncertainty quantification across different configuration option values $o$ in the validation set (marginal coverage), we use the *feature-stratified coverage* (FSC) metric [1]. In our case, features correspond to configuration options. Feature-stratified coverage provides a more granular view than the interval coverage rate by measuring the worst-case conditional coverage across subsets of the validation data, ensuring that coverage holds across all option values and not just on average. To compute the feature-stratified coverage, the validation set $d_{\text{val}}$ is partitioned by option values. For binary options, two subsets are created ($o = 0$ and $o = 1$); for numeric options, values are binned into five intervals spanning the option's range, yielding five subsets. Let $d_{1..g} \subset d_{\text{val}}$ denote the $g$ non-empty subsets. Interval coverage rate is computed for each, and feature-stratified coverage is defined as:

$$\mathrm{FSC}(d_{1..g}, \mathrm{I}) = \min_{j=1..g} \mathrm{COV}(d_j, \mathrm{I}) \tag{4}$$

A well-calibrated uncertainty quantification method should achieve a feature-stratified coverage close to the nominal coverage level, indicating robustness against option-induced uncertainty. It should be noted that if we look at the results, the evaluation of a subset $d_{1..g}$ does violate the rule, that the training and evaluation data is exchangeable. A more detailed analysis is provided in the supplementary material, where we use not only the minimum value but the coverage value of every subset.

*5.5.3 Mean Percentage Interval Width.* The metrics COV and FSC indicate whether the user-specified proportion of measured values lies within the predicted confidence intervals, thus verifying the correctness of the intervals. However, they do not account for how informative the intervals are. For example, for a measured value $v = 6$, both $\mathrm{I} = [5, 7]$ and $\mathrm{I} = [0, 15]$ are treated equally. Given equivalent coverage, narrower intervals are preferred, as they offer more precise information about the true value while maintaining reliability and avoiding underconfidence. Thus we use the *mean percentage interval width* (MIW) as a secondary ranking criterion, calculated over the confidence intervals I for predicted values $\hat{Y}$ on the validation set:

$$\mathrm{MIW}(\mathrm{I}, \hat{Y}) = \frac{1}{|\mathrm{I}|} \sum_{\mathrm{I}_c \in \mathrm{I}} \frac{(\mathrm{I}_c^+ - \mathrm{I}_c^-)}{|\hat{y}(c)|}, \tag{5}$$

where $\mathrm{I}_c^+$ and $\mathrm{I}_c^-$ are the upper and lower bounds of the interval for the predicted value $\hat{y}$, respectively.

*5.5.4 Normalized Quantile Range.* To assess whether confidence intervals are sufficiently narrow, we compare the MIW to the normalized quantile range ($\mathrm{NQR}_\alpha$) of the ground-truth measurement data. For each subject system, $\mathrm{NQR}_\alpha$ is computed from the empirical distribution of measured performance values as the central $1-\alpha$-quantile range, normalized by the mean performance value:

$$\mathrm{NQR}_\alpha = \frac{Q_{1-\alpha/2} - Q_{\alpha/2}}{\bar{v}}. \tag{6}$$

This metric captures the relative dispersion of the measurement data independent of the absolute scale of the system. If the MIW is smaller than $\mathrm{NQR}_\alpha$, we regard the predicted intervals as reasonably narrow with respect to the inherent variability of the ground truth.

*5.5.5 Mean Absolute Percentage Error.* To quantify differences in the scalar prediction accuracy we use the *mean absolute percentage error* (MAPE), which quantifies the relative error of point predictions $\hat{y}$ of configurations and allows for a direct comparison of scalar prediction accuracy across different approaches. It is commonly used in the literature [7, 13, 36]:

$$\mathrm{MAPE}(d, \hat{Y}) = \frac{100}{|d|} \sum_{(c,v)\in d} \frac{|v - \hat{y}_c|}{|v|} \tag{7}$$

*5.5.6 Probabilistic MAPE.* To assess how well an approach models not only a point estimate but also predictive uncertainty, we use the *probabilistic MAPE* (pMAPE) [8]. pMAPE generalizes MAPE to probabilistic predictions by replacing the absolute error $|y(c) - \hat{y}(c)|$ with the *continuous ranked probability score* (CRPS) [22]. As Bayesian methods directly provide predictive distributions, CRPS can be computed from their posterior predictive densities. For conformal prediction, which yields confidence intervals instead of predictive

[1] https://mapie.readthedocs.io/en/stable/

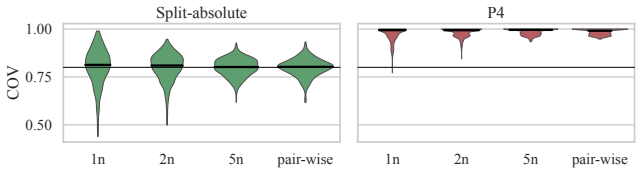



**Figure 4: Interval coverage rate (COV) for split-absolute (left) and P4 (right) across four sample sizes. Violins summarize 30 runs over all real-world systems; the black line marks the 0.8 target**

densities, we approximate a predictive distribution from intervals at multiple significance levels. We define pMAPE as follows:

$$\text{pMAPE}(d, \hat{Y}, I) = \frac{100}{|d|} \sum_{(c,v)\in d} \left| \frac{\text{CRPS}(c)}{v} \right|, \tag{8}$$

*5.5.7 Training and Prediction Time.* To assess the computational scalability of uncertainty quantification methods, we measure the wall-clock time required for (1) training the model on the training set and (2) generating predictions with confidence intervals for all configuration in the validation set. We report the total time in seconds, which allows for direct comparison of the computational overhead introduced by different methods.

## 5.6 Evaluation

All experiments were executed independently on uniform hardware: an AMD EPYC 7513 (32 cores, 32GB RAM). Experiments involving DeepPerf used an NVIDIA A100 GPU with 40GB memory.

The evaluation metrics are computed on a dedicated validation set. This set consists of the complete collection of measured configurations, excluding those included in the training set $d$. Prediction intervals are generated for error levels ($\alpha$) of 0.1 to 0.9 in increments of 0.1. These are used to approximate the CDF required for the pMAPE metric.For all other evaluation metrics, $\alpha = 0.2$ is used, as higher error levels produce intervals that are less likely to include the observed values, diminishing their practical applicability in software performance prediction. We computed Cohen's effect sizes to compare the baseline with the conformal prediction methods. For COV and FSC, we determined the absolute error relative to the target level. Additionally, we applied a Kruskal-Wallis significance test with Holm correction to assess statistical significance.

# 6 Results

This section reports the results for both research questions. Table 2 summarizes the mean results across all real-world systems, while the figures highlight selected aspects of these findings. The complete result set is available in the supplementary material.

## 6.1 Applicability of Conformal Prediction ($RQ_1$)

We evaluate the applicability of conformal prediction to software performance modeling ($RQ_1$) along three dimensions: confidence interval accuracy, consistency across configuration options, and interval informativeness.

*6.1.1 Accuracy of the Confidence Intervals ($RQ_{1.1}$):* We first examine the mean interval coverage rate (COV) for $\alpha = 0.2$ (Figure 4 left). A reasonable interval coverage rate matches the target level 0.8, which means that 80 % of all measured values are within the interval. An interval coverage rate higher than 0.8 implies that the intervals are too broad and thus less informative. By contrast, an interval coverage rate below 0.8 indicates intervals that are too narrow and therefore reflect an overestimation of prediction reliability. We further report the split ratios and $k$ values yielding coverage closest to the target: A 0.75 split for the absolute and 0.9 for the localized non-conformity measure, and $k = 10$ for cross-validation+. No significant differences can be observed across different $k$ values (Kruskal-Wallis: $p > 0.11$).

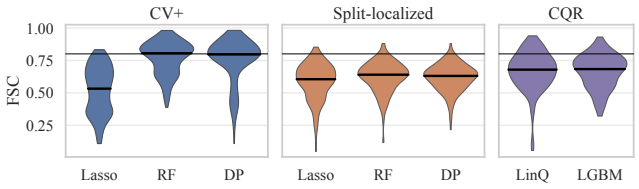



**Figure 5: Feature-stratified coverage (FSC) for CV+, split-localized and CQR across the underlying learning algorithms for pair-wise sampling. Violins show 30 runs over all real-world systems; the black line marks the 0.8 target.**

Cross-validation+ typically yields confidence intervals that are too wide, especially with accurate point-estimate models such as random forests or neural networks (cf. values above 0.8 of CV+ in row 1 in Table 2). By contrast, split conformal prediction consistently aligns coverage with the target, regardless of the system or learning algorithm, and performs well with both absolute and localized non-conformity measures. All methods benefit from larger training sets: interval coverage converges to the target and variance across repetitions decreases (Figure 4 left). This effect is most pronounced between small and medium sample sizes, indicating that modest data increases interval stability. These results show that conformal prediction is able to provide reliable confidence intervals even under data constraints typical in configurable software systems. Split-based methods, in particular, offer robust calibration and low sensitivity to variation across datasets and learners.

Summarizing, conformal prediction is feasible for post-hoc uncertainty quantification in software performance modeling: It achieves near-target interval coverage ($COV \approx 0.8$) even with small datasets, which was unclear before. Split conformal prediction performs reliably across methods, while cross-validation+ tends to produce overly conservative intervals.

*6.1.2 Option-induced Uncertainties ($RQ_{1.2}$):* Feature-stratified coverage indicates how well intervals capture uncertainties of individual options. It should match interval coverage if option-induced uncertainties are captured well and be lower if they are not.

Our results show inconsistent improvement with increasing dataset size. Split-absolute varies across systems, whereas cross-validation+ varies primarily with the learning method. This is an interesting result, as it shows how the specifics of the underlying prediction models yield varying uncertainty quantification accuracies for individual options. For instance, the largest deviation can be observed for Lasso and near-target values with random forests

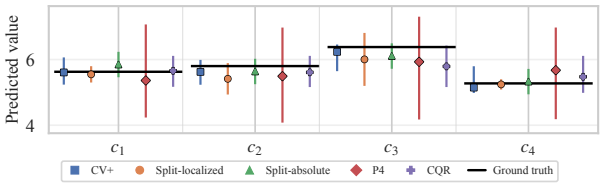



**Figure 6: Example prediction intervals for four random x264 configurations under pair-wise sampling, comparing conformal prediction with random forest to P4. Dots indicate predicted performance values, and the black line marks the measured performance.**

and DeepPerf (Figure 5 left), indicating that the latter two methods capture the influence of individual options and possible interactions better. Split-localized shows a more similar FSC behavior across learners than cross-validation+ (Figure 5 right), but it remains below the target level and requires more data to improve. FSC increases only gradually and more slowly than COV (Table 2).

Capturing option-induced uncertainties consistently is more challenging than achieving overall coverage. Cross-validation+ performs well for this objective with suitable learners, while split-localized becomes a competitive choice when sufficient training data is available.

*6.1.3 Informativeness of Confidence Intervals ($RQ_{1.3}$):* To assess the informativeness of the predicted confidence intervals (RQ1.3), we analyze the mean percentage interval width (MIW). Since the amount of inherent uncertainty differs across software systems, interval widths are not compared directly across systems, but only within each system. Lower MIW values indicate more informative intervals, provided that coverage remains close to the target level. As a reference for reasonable interval width, we compare MIW to the normalized quantile range (NQR) of the ground truth.

At a sampling size $5n$, MIW is smaller than the NQR in 38 of 72 evaluated combinations of system, conformal prediction method, and learning algorithm, indicating that conformal prediction often yields reasonably narrow intervals. As an illustration, for x264 at a sampling size $5n$, MIW ranges from 0.10 to 0.20 depending on the non-conformity measure and learning algorithm, while the corresponding NQR is 0.19.

Across sampling sizes, absolute-score methods show little improvement for Lasso, whereas for random forests, both split conformal prediction and cross-validation tend to converge toward narrower intervals as more data become available. Localized scores generally yield wider intervals but also show a more consistent narrowing trend across learners. Although cross-validation with Lasso achieves the smallest MIW (Table 2), its low feature-stratified coverage indicates that these intervals fail to capture option-induced uncertainty adequately.

Conformal prediction can produce reasonably informative confidence intervals for software performance prediction. The most promising choices are cross-validation+ with random forests or DeepPerf, and split-localized conformal prediction, which offers the most balanced trade-off between narrowness and reliability.

To summarize, conformal prediction is feasible for software performance modeling, producing accurate confidence intervals even under data restrictions. Split-localized and cross-validation+ methods with random forests capture option-specific uncertainties effectively, though the performance of cross-validation+ is model-dependent. These two methods yield the most accurate and narrow confidence intervals.

*6.1.4 Discussion.* Overall, we found that all conformal prediction methods are suitable for uncertainty quantification, even when provided only with small amounts of data. This is an important insight since the theoretical guaranties rely on the law of large numbers. In essence, we recommend using between $2n$ and $5n$ samples, balancing accuracy and measurement cost. To put this into perspective, for larger software systems, a small fraction of the configuration space (e.g., 7z: $< 1\,\%$) already suffices for high quality confidence intervals. Moreover, precise intervals at the level of individual option values need more measurements to cover uncertainties induced by the configuration landscape, and this certainly must be covered by the measurement set.

As a further notable insight, we found that not all methods perform equally well. The most robust conformal prediction methods for the individual metrics are: Split conformal prediction for COV, split-localized or cross-validation+ with Random Forest or DeepPerf for FSC, and split-localized or cross-validation+ for the MIW. Figure 6 illustrates, for four example configurations of x264, how interval width and adaptivity vary across methods, from the uniform widths produced by split-absolute to the more adaptive intervals produced by split-localized. Additionally, split-localized should be avoided when the measurement budget is very low. If the application scenario allows for different prediction models, we know that certain methods, such as cross-validation+, are sensitive to this selection, and that depending on the metric of interest, the best model should be selected.

The reliability of uncertainty estimates varies with the learning method. A more detailed analysis of Synth. 1 suggests that low feature-stratified coverage often indicates that the model fails to capture relevant system behaviors, such as alternating option groups, rather than merely reflecting data sparsity or injected uncertainty; the corresponding analysis is included in the supplementary material. This suggests that conformal prediction may serve as a diagnostic tool to assess model adequacy. This opens future research to exploit this property: With the model-agnostic nature of conformal prediction, it may be possible to guide model selection based on system characteristics, providing developers with a practical tool for choosing the optimal prediction model.

## 6.2 Comparison with Alternative Appr. ($RQ_2$)

We next compare conformal prediction with P4 and conformalized quantile regression (CQR); for CQR, only the better-performing LGBM model is shown in Table 2. In addition to the correctness and informativeness of the predicted confidence intervals, this comparison considers scalar prediction accuracy, probabilistic accuracy, and computational efficiency.

*6.2.1 Uncertainty Quantification:* Comparing the methods, all systems exhibit a larger gap between the interval coverage rate and the

**Table 2: Mean-aggregated metrics across all runs and real-world software systems. Best values per sample size marked in blue.**

| | Cross-validations+ | | | | | | | | | Split absolute | | | | | | | | | Split localized | | | | | | | | | CQR | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Lasso | | | RF | | | DP | | | Lasso | | | RF | | | DP | | | Lasso | | | RF | | | DP | | | LGBM | | | P4 | | |
| | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n | 1n | 2n | 5n |
| COV | 0.80 | 0.80 | 0.80 | 0.90 | 0.91 | 0.91 | 0.84 | 0.88 | 0.89 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.80 | 0.79 | 0.80 | 0.80 | 0.80 | 0.80 | 0.79 | 0.80 | 0.80 | 0.84 | 0.85 | 0.82 | 0.97 | 0.98 | 0.99 |
| FSC | 0.51 | 0.51 | 0.51 | 0.73 | 0.77 | 0.78 | 0.62 | 0.72 | 0.76 | 0.51 | 0.50 | 0.50 | 0.54 | 0.55 | 0.57 | 0.53 | 0.56 | 0.58 | 0.54 | 0.52 | 0.55 | 0.54 | 0.55 | 0.60 | 0.54 | 0.57 | 0.59 | 0.66 | 0.69 | 0.66 | 0.90 | 0.92 | 0.93 |
| MIW | 0.41 | 0.39 | 0.39 | 0.56 | 0.52 | 0.49 | 0.51 | 0.50 | 0.46 | 0.52 | 0.43 | 0.41 | 0.62 | 0.52 | 0.46 | 0.61 | 0.51 | 0.59 | 0.95 | 0.56 | 0.48 | 1.05 | 0.72 | 0.61 | 1.00 | 0.85 | 0.69 | 0.73 | 0.54 | 0.51 | 1.97 | 2.11 | 1.61 |
| MAPE | 0.21 | 0.22 | 0.22 | 0.16 | 0.15 | 0.13 | 0.18 | 0.16 | 0.13 | 0.22 | 0.22 | 0.21 | 0.21 | 0.19 | 0.15 | 0.23 | 0.19 | 0.17 | 0.24 | 0.23 | 0.20 | 0.25 | 0.23 | 0.18 | 0.37 | 0.22 | 0.16 | 0.18 | 0.15 | 0.13 | 0.30 | 0.32 | 0.32 |
| pMAPE | 0.13 | 0.14 | 0.14 | 0.08 | 0.06 | 0.04 | 0.10 | 0.08 | 0.06 | 0.14 | 0.14 | 0.13 | 0.13 | 0.12 | 0.09 | 0.14 | 0.12 | 0.11 | 0.15 | 0.15 | 0.13 | 0.16 | 0.14 | 0.12 | 0.29 | 0.14 | 0.11 | 0.10 | 0.07 | 0.06 | 0.19 | 0.20 | 0.20 |

**Table 3: Effect sizes by metric for CV+ with Random Forest compared to the P4 baseline and CQR with LGBM. For COV and FSC, effect sizes are based on the error relative to the target level. Positive values indicate improvement over the alternative approach. Underlined means not significant.**

| **P4** | 7z | Apache | HSQLDB | LRZIP | Nginx | PostgreSQL | VP8 | X264 |
|---|---|---|---|---|---|---|---|---|
| COV | 1.06 | 1.52 | 1.70 | 1.52 | 1.75 | 1.76 | 2.22 | 1.61 |
| FSC | -0.66 | 0.97 | 1.44 | -1.04 | 0.21 | 1.54 | 0.66 | 1.89 |
| MAPE | 3.08 | 4.36 | 5.84 | 2.73 | 6.84 | 1.64 | 3.30 | 4.99 |
| PMAPE | 4.48 | 5.89 | 5.10 | 3.19 | 7.96 | 2.66 | 3.97 | 6.94 |
| MIW | 3.79 | 5.66 | 7.58 | 3.07 | 6.45 | 3.28 | 0.36 | 9.40 |
| **CQR** | 7z | Apache | HSQLDB | LRZIP | Nginx | PostgreSQL | VP8 | X264 |
| COV | -0.65 | -0.30 | -0.20 | -0.47 | -0.18 | -0.30 | -0.17 | -0.73 |
| FSC | 0.58 | 0.37 | 0.38 | 0.25 | 0.25 | 0.43 | 0.51 | 0.28 |
| MAPE | -0.38 | -0.39 | 0.19 | 0.13 | 0.69 | 0.45 | -0.01 | -0.12 |
| PMAPE | 0.56 | 0.53 | 0.43 | 0.94 | 1.15 | 1.13 | 0.53 | 0.78 |
| MIW | 0.08 | 0.06 | 0.59 | 0.07 | 0.20 | 0.43 | 0.17 | 0.27 |

target level under P4, with values approaching the upper limit of 1 (Figure 4 right and Table 2). Only LRZIP and 7Z meet the target for feature-stratified coverage; for all other systems, P4 yields values close to 1 with no variation as the dataset size increases. The mean interval width of P4 consistently exceeds that of the conformal prediction methods for all systems, up to ten times the predicted value. These large intervals result in the overestimation reflected by the interval coverage rate. No reduction in interval width is observed with increasing dataset size. The quantile-based approach has interval coverage and feature-stratified coverage closer to the target level than P4 (2). Its interval width is comparable to that of the other conformal prediction methods. Although it is not the best method for any metric, it is also never the worst (5, 2). To support these observations, Table 3 reports the effect sizes of the best-performing method (cross-validation+) relative to P4 and CQR, where 0.1, 0.3, and 0.5 denote small, medium, and large effects, following Cohen's conventions. Relative to P4, we observe very large positive effect sizes for mean interval width across all systems, indicating that conformal prediction yields substantially tighter intervals. Coverage metrics (COV and FSC) also show improved reliability for all but two systems (7z and LRZIP), where FSC performs worse than the baseline. All effect sizes are statistically significant based on a Kruskal–Wallis test with Holm correction. In comparison to CQR, the pattern is more varied. Cross-validation+ shows positive effect sizes for FSC and MIW, but not for COV. The effect sizes also range from small to large, depending on the metric. P-values and effect sizes for all methods are available in the supplement.

Overall, conformal prediction provides tighter and better-calibrated confidence intervals than P4. P4 consistently overestimates uncertainty, with interval coverage far exceeding the target and no improvement as dataset size increases. The CQR method yields uncertainty quantification performance intermediate between the scalar-based conformal prediction methods.

*6.2.2 Prediction Accuracy:* Regarding prediction accuracy, *Cross-validation+* achieves the lowest MAPE, whereas P4 shows the highest, except for the split-localized method under sparse data. Split-based conformal methods perform slightly worse than cross-validation+. Among learning models, Lasso yields the second highest MAPE after the P4, showing minimal improvement. By contrast, random forest and DeepPerf achieve up to 68 % lower errors and benefit from larger datasets. P4 exhibits the opposite trend—MAPE increases with more data. The quantile-based methods with the linear quantile algorithm performed comparably to the split-based method with Lasso, while those with the LGBM algorithm performed only marginally worse than cross-validation+ with random forest, and in some software systems even produced better results.

While pMAPE values are closer to zero, their trends closely follow MAPE, with minor differences across learning methods within each scalar-based conformal prediction approach. For cross-validation+, pMAPE favors random forest over DeepPerf. The comparison with P4 remains consistent—P4 yields higher (worse) pMAPE than conformal prediction in nearly all cases. Compared with the other conformal prediction methods, CQR shows slightly worse pMAPE relative to MAPE. Statistically, P4 performs worse than cross-validation+ with random forest across all systems, as indicated by the positive effect sizes in Table 3. Although CQR achieves a better MAPE for four systems, two of which are significant, it shows significantly worse pMAPE for all systems.

In summary, cross-validation+ with random forest achieves the lowest MAPE in most cases, and the lowest pMAPE in all cases, indicating superior point and probabilistic prediction accuracy. P4 shows the higher errors, worsening with more data. Conformal methods consistently outperform P4, particularly when using complex learners.

*6.2.3 Scalability:* Figure 7 presents training and prediction times for P4, CQR with LGBM and conformal methods using random forest as underlying model, evaluated on three dataset sizes.

All conformal prediction methods train in under 0.5 seconds, with sub-linear growth for random forest (linear growth for Lasso). In contrast, P4 is over ten times slower and exhibits steep growth as the training set size increases.

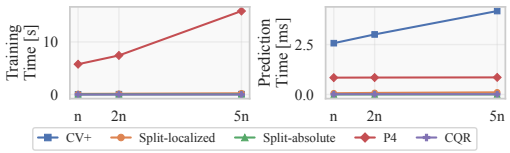



**Figure 7: Training time (left) and prediction time per configuration (right) for conformal prediction (random forest), CQR (LGBM) and P4 across training set sizes $n$, $2n$, and $5n$.**

For prediction time of 25,000 configurations, the split method, CQR and P4 show nearly constant times: 2–12 seconds for the split method and around 21 seconds for P4. In comparison, the cross-validation+ method exhibits linearly increasing prediction time with larger training sets. In all cases, conformal prediction methods are strongly influenced by the underlying model.

In summary, the methods differ significantly in scalability for training and prediction, ranging from constant growth in split conformal prediction to beyond-linear growth in P4. Since measuring a single configuration typically takes seconds to minutes, none of the methods, including P4, hinder practical applicability.

*6.2.4 Discussion.* Our results draw a clear picture: Conformal prediction is generally the better choice, not only due to improved prediction accuracy (as shown by MAPE) but also due to its ability to provide meaningful uncertainty estimates. This result, together with the model-agnostic nature of conformal prediction, suggests that the selected conformal prediction methods are effective for uncertainty quantification in performance modeling. The quantile-based conformal prediction method also shows strong results, but it never achieves the best performance. This may be because there is no known underlying model that has been adapted for the performance modeling of configurable software systems.

P4 has two main limitations: It does not improve as more data become available, and it produces overly wide confidence intervals. Figure 6 illustrates this for four example configurations of x264, where P4 yields substantially wider intervals than the conformal prediction methods. P4's interval often exceeds the full range of observed training values and, therefore, provides only limited insight. Since conformal prediction with Lasso also shows no MAPE improvement, the first limitation of the P4 method likely stems from the linear model underlying P4 rather than the Bayesian approach.

## 6.3 Threats to Validity

*6.3.1 Internal Validity.* We mitigate threats to internal validity through several measures. All experiments are repeated 30 times with random seeds to minimize the influence of initialization effects. We use randomized hyperparameter search with cross-validation to ensure optimal configurations for each learning algorithm, thereby preventing unfair advantages due to suboptimal model settings. We optimized P4 to avoid biased comparisons in favor of conformal prediction. Consistent performance across repetitions and evaluation metrics indicates the robustness of our findings and suggests the absence of setting-specific artifacts. These measures collectively ensure that the observed effects are attributable to the modeling techniques rather than experimental artifacts.

*6.3.2 External Validity.* Several design choices influence the generalizability of our findings. Our selection of three representative learning models spans a range of complexity and approximation capabilities (from simple and interpretable to state-of-the-art performance) and reflects practical modeling choices in software performance prediction. Our subject systems cover several application domains, much like in prior work [7, 15, 37]. To explore high-dimensional settings, we include two synthetic systems with uncertainty based on the largest measurement error in the real systems. For conformal prediction, we evaluate three combinations of techniques and non-conformity measures, chosen for their practical relevance and coverage of key dimensions in the design space. We restrict the comparison to only CQR as an interval-based method and consider two underlying learning models for it. While these combinations demonstrate applicability and superiority over P4, one cannot conclude that the optimal combination has been identified; this, however, is not our objective.

## 7 Conclusion

Uncertainty quantification is essential in performance prediction for software systems, where scalar point estimates alone fail to reflect the risk of extreme or unstable behavior, particularly under complex option interactions. Confidence intervals provide a way to express prediction reliability, enabling more informed system tuning and decision-making in real-world environments.

In this work, we propose conformal prediction as a model-agnostic alternative to the Bayesian P4 method, addressing its limitations, such as reliance on prior distributions and restricted model compatibility. Conformal prediction augments any machine learning model with statistically valid uncertainty estimates without altering the base prediction mechanism. We evaluate its feasibility and merits through an extensive empirical study across ten configurable software systems, using multiple machine learning models, data sizes, conformal prediction methods, and non-conformity measures. We compare conformal prediction against the state of the art (P4) using standard metrics for uncertainty quantification, prediction accuracy, and scalability. Our results show that split conformal prediction with a localized non-conformity measure delivers tight and accurate confidence intervals. This is comparable to cross-validation+, paired with robust learners such as random forest or DeepPerf. Conformal prediction intervals are narrower and more reliable and benefit from larger datasets, and split conformal prediction achieved faster runtimes. Notably, while P4 produced overly broad intervals, often exceeding the full range of observed values, it fails to improve with additional data. Conformal prediction yields meaningful, data-efficient uncertainty estimates. Its model-agnostic nature allows seamless integration into existing performance modeling pipelines, making it a versatile tool for practitioners.

## Acknowledgments

This work was supported by the Federal Ministry of Research, Technology and Space of Germany within the Center of Excellence for AI research "ScaDS.AI"; the European Social Fund and the Free State of Saxony under grant A.100760692 (project: Teaching-AI); and the German Research Foundation under grant AP 206/11-2 and grant 389792660 as part of TRR 248 "CPEC".

## Data Availability

All experiments, including data preparation, performance measurement, model training, conformal prediction calibration, and evaluation metric computation, are available in a public repository; DOI: https://doi.org/10.5281/zenodo.19185890.

## References


[1] Anastasios N. Angelopoulos and Stephen Bates. 2023. Conformal Prediction: A Gentle Introduction. *Foundations and Trends in Machine Learning* 16, 4 (2023), 494–591. https://doi.org/10.1561/2200000101

[2] Rina Foygel Barber, Emmanuel J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. 2021. Predictive Inference with the Jackknife+. *The Annals of Statistics* (2021). https://doi.org/10.1214/20-aos1965

[3] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.* 13, null (2012), 281–305. https://dl.acm.org/doi/10.5555/2503308.2188395

[4] Leo Breiman. 2001. Random Forests. *Machine Learning* (2001). https://doi.org/10.1023/a:1010933404324

[5] Zifan Chen, Mengdie Wang, Zhuoqi Zeng, and Wenjie Ping. 2026. Uncertainty-Aware Financial Forecasting: Leveraging Conformal Prediction for Risk-Adjusted Models. *IEEE Access* 14 (2026), 90053–90077. https://doi.org/10.1109/ACCESS.2026.3702666

[6] Nicolas Dewolf, Bernard De Baets, and Willem Waegeman. 2023. Valid Prediction Intervals for Regression Problems. *Artificial Intelligence Review* 56, 1 (2023), 577–613. https://doi.org/10.1007/s10462-022-10178-5

[7] Johannes Dorn, Sven Apel, and Norbert Siegmund. 2023. Mastering Uncertainty in Performance Estimations of Configurable Software Systems. *Empirical Software Engineering* (2023). https://doi.org/10.1007/s10664-022-10250-2

[8] Johannes Dorn, Stefan Mühlbauer, Stefan Jahns, Sven Apel, and Norbert Siegmund. 2026. Bayesian Multi-Level Performance Models for Multi-Factor Variability of Configurable Software Systems.. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1145/3744916.3773131

[9] Ethan Goan and Clinton Fookes. 2020. *Bayesian Neural Networks: An Introduction and Survey*. Springer, 45–87. https://doi.org/10.1007/978-3-030-42553-1_3

[10] Jingzhi Gong and Tao Chen. 2023. Predicting Software Performance with Divide-and-Learn. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 858–870. https://doi.org/10.1145/3611643.3616334

[11] Leying Guan. 2023. Localized Conformal Prediction: a Generalized Inference Framework for Conformal Prediction. *Biometrika* 110 (2023), 33–50. https://doi.org/10.1093/biomet/asac040

[12] Jianmei Guo, Dingyu Yang, Norbert Siegmund, Sven Apel, Atrisha Sarkar, Pavel Valov, Krzysztof Czarnecki, Andrzej Wasowski, and Huiqun Yu. 2017. Data-efficient Performance Learning for Configurable Systems. *Empirical Software Engineering* (2017). https://doi.org/10.1007/s10664-017-9573-6

[13] Huong Ha and Hongyu Zhang. 2019. DeepPerf: Performance Prediction for Configurable Software with Deep Sparse Neural Network. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1109/icse.2019.00113

[14] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, and Sven Apel. 2020. The Interplay of Sampling and Machine Learning for Software Performance Prediction. *IEEE Software* (2020). https://doi.org/10.1109/ms.2020.2987024

[15] Christian Kaltenecker, Alexander Grebhahn, Norbert Siegmund, Jianmei Guo, and Sven Apel. 2019. Distance-Based Sampling of Software Configuration Spaces. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1109/icse.2019.00112

[16] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, Vol. 30. Curran Associates Inc., 3149–3157.

[17] Roger Koenker and Gilbert Bassett. 1978. Regression Quantiles. *Econometrica* 46, 1 (1978), 33. https://doi.org/10.2307/1913643

[18] Sergiy Kolesnikov, Norbert Siegmund, Christian Kästner, Alexander Grebhahn, and Sven Apel. 2018. Tradeoffs in modeling performance of highly configurable software systems. *Software & Systems Modeling* 18, 3 (2018), 2265–2283. https://doi.org/10.1007/s10270-018-0662-9

[19] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. 2017. Distribution-Free Predictive Inference For Regression. arXiv:1604.04173 [stat.ME] https://arxiv.org/abs/1604.04173

[20] Valery Manokhin. 2022. *Machine Learning for Probabilistic Prediction*. Ph. D. Dissertation. Royal Holloway, University of London. https://doi.org/10.5281/ZENODO.6727505

[21] Valeriy Manokhin. 2024. *Practical Guide to Applied Conformal Prediction*. Packt Publishing, Limited.

[22] James E. Matheson and Robert L. Winkler. 1976. Scoring Rules for Continuous Probability Distributions. *Management Science* (1976). https://doi.org/10.1287/mnsc.22.10.1087

[23] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. 2018. Faster Discovery of Faster System Configurations with Spectral Learning. *Automated Software Engineering* 25, 2 (2018), 247–277. https://doi.org/10.1007/s10515-017-0225-2

[24] Tim Oates and David Jensen. 1997. The Effects of Training Set Size on Decision Tree Complexity. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, David Madigan and Padhraic Smyth (Eds.). Morgan Kaufmann Publishers Inc., 254–262.

[25] Jeho Oh, Don Batory, Margaret Myers, and Norbert Siegmund. 2017. Finding near-optimal configurations in product lines by random sampling. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 61–71. https://doi.org/10.1145/3106237.3106273

[26] Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. 2002. Inductive Confidence Machines for Regression. In *Proceedings of the European Conference on Machine Learning (ECML)*. Springer, 345–356. https://doi.org/10.1007/3-540-36755-1_29

[27] Robin L. Plackett and John P. Burman. 1946. The Design of Optimum Multifactorial Experiments. *Biometrika* (1946). https://doi.org/10.1093/biomet/33.4.305

[28] Vincent Plassier, Nikita Kotelevskii, Aleksandr Rubashevskii, Fedor Noskov, Maksim Velikanov, Alexander Fishkov, Samuel Horvath, Martin Takac, Eric Moulines, and Maxim Panov. 2024. Efficient Conformal Prediction under Data Heterogeneity. https://doi.org/10.48550/arXiv.2312.15799 arXiv:2312.15799 [stat.ML]

[29] Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. 2019. Conformalized Quantile Regression. https://doi.org/10.48550/arXiv.1905.03222 arXiv:1905.03222 [stat.ME]

[30] Frank Rosenblatt. 1958. The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* (1958). https://doi.org/10.1037/h0042519

[31] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. 2008. *Global Sensitivity Analysis*. John Wiley. https://doi.org/10.1002/9780470725184

[32] Nabeel Seedat, Alan Jeffares, Fergus Imrie, and Mihaela van der Schaar. 2023. Improving Adaptive Conformal Prediction Using Self-Supervised Learning. https://doi.org/10.48550/arXiv.2302.12238 arXiv:2302.12238 [cs.LG]

[33] Glenn Shafer and Vladimir Vovk. 2007. A tutorial on conformal prediction. https://doi.org/10.48550/arXiv.0706.3188 arXiv:0706.3188 [cs.LG]

[34] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. 2015. Performance-influence Models for Highly Configurable Systems. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM. https://doi.org/10.1145/2786805.2786845

[35] Norbert Siegmund, Sergiy S. Kolesnikov, Christian Kastner, Sven Apel, Don Batory, Marko Rosenmuller, and Gunter Saake. 2012. Predicting performance via automated feature-interaction detection. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE, 167–177. https://doi.org/10.1109/icse.2012.6227196

[36] Norbert Siegmund, Marko Rosenmüller, Martin Kuhlemann, Christian Kästner, Sven Apel, and Gunter Saake. 2011. SPL Conqueror: Toward Optimization of Non-functional Properties in Software Product Lines. *Software Quality Journal* (2011). https://doi.org/10.1007/s11219-011-9152-9

[37] Norbert Siegmund, Stefan Sobernig, and Sven Apel. 2017. Attributed Variability Models: Outside the Comfort Zone. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM. https://doi.org/10.1145/3106237.3106251

[38] Ralph C. Smith. 2013. *Uncertainty Quantification: Theory, Implementation, and Applications*. Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611973228

[39] Chico Sundermann, Vincenzo Francesco Brancaccio, Elias Kuiter, Sebastian Krieter, Tobias Heß, and Thomas Thüm. 2024. Collecting Feature Models from the Literature: A Comprehensive Dataset for Benchmarking. In *Proceedings of the International Software Product Line Conference (SPLC)*. ACM. https://doi.org/10.1145/3646548.3672590

[40] John R. Taylor. 1982. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements* (2nd ed.). University Science Books.

[41] Robert Tibshirani. 1996. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society* (1996). https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

[42] Janette Vazquez and Julio C. Facelli. 2022. Conformal Prediction in Clinical Medical Sciences. *Journal of Healthcare Informatics Research* (2022). https://doi.org/10.1007/s41666-021-00113-8

[43] Vladimir Vovk. 2012. Conditional Validity of Inductive Conformal Predictors. https://doi.org/10.48550/ARXIV.1209.2673

[44] Bo Wang and D. M. Titterington. 2005. Inadequacy of interval estimates corresponding to variational Bayesian approximations. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 373–380.

[45] Yiren Wang and Dimitris N. Politis. 2026. Model-free Bootstrap and Conformal Prediction in Regression: Conditionality, Conjecture Testing, and Pertinent Prediction Intervals. *Journal of Nonparametric Statistics* 38, 1 (2026), 311–345. https://doi.org/10.1080/10485252.2025.2603398

[46] Max Weber, Christian Kaltenecker, Florian Sattler, Sven Apel, and Norbert Siegmund. 2023. Twins or False Friends? A Study on Energy Consumption and Performance of Configurable Software. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1109/icse48619.2023.00177

[47] Niklas Werner. 2019. *Energy and Performance Evolution of Configurable Systems: Case Studies and Experiments*. Master's thesis. University of Passau.

Received 2026-03-25; accepted 2026-06-18