

# PCS-UQ: Uncertainty Quantification via the Predictability-Computability-Stability Framework

Abhineet Agarwal<sup>a,1</sup>, Michael Xiao<sup>a,1</sup>, Rebecca Barter<sup>c,1</sup>, Omer Ronen<sup>a</sup>, Boyu Fan<sup>a</sup>, and Bin Yu<sup>a,b,2</sup>

<sup>a</sup>Department of Statistics, University of California, Berkeley; <sup>b</sup>Department of Electrical Engineering and Computer Science, University of California, Berkeley; <sup>c</sup>Department of Epidemiology, University of Utah

As machine learning (ML) models are increasingly deployed in high-stakes domains, trustworthy uncertainty quantification (UQ) is critical for ensuring the safety and reliability of these models. Traditional UQ methods rely on specifying a true generative model and are not robust to misspecification. On the other hand, conformal inference allows for arbitrary ML models but does not consider model selection, which leads to large interval sizes. We tackle these drawbacks by proposing a UQ method based on the predictability, computability, and stability (PCS) framework for veridical data science proposed by Yu and Kumbier. Specifically, PCS-UQ addresses model selection by using a prediction check to screen out unsuitable models. PCS-UQ then fits these screened algorithms across multiple bootstraps to assess inter-sample variability and algorithmic instability, enabling more reliable uncertainty estimates. Further, we propose a novel calibration scheme that improves local adaptivity of our prediction sets. Experiments across 17 regression and 6 classification datasets show that PCS-UQ achieves the desired coverage and reduces width over conformal approaches by  $\approx 20\%$ . Further, our local analysis shows PCS-UQ often achieves target coverage across subgroups while conformal methods fail to do so. For large deep-learning models, we propose computationally efficient approximation schemes that avoid the expensive multiple bootstrap trainings of PCS-UQ. Across three computer vision benchmarks, PCS-UQ reduces prediction set size over conformal methods by 20%. Theoretically, we show a modified PCS-UQ algorithm is a form of split conformal inference and achieves the desired coverage with exchangeable data.

## 1. Introduction

Recent decades have seen tremendous growth in machine learning (ML) and artificial intelligence (AI). As these systems increasingly inform high-stakes decisions, ensuring their reliability and safety has become a central concern. Failures in trust and reproducibility—exemplified by the replication crisis in biomedical research (1–3)—highlight the risks of reaching conclusions based on questionable models. A key component of establishing confidence and enabling responsible decision-making from data and models is trustworthy uncertainty quantification (UQ). Accurate estimates of uncertainty allow practitioners to reach reliable data-driven conclusions, and accurately assess risk to mitigate downstream consequences. Indeed, researchers believe that poor estimates of uncertainty were a significant cause of the biomedical replication crisis (1).

Current approaches to UQ are based on a traditional statistical modeling framework pioneered by R.A. Fisher and others a century ago. This framework relies on specifying a probabilistic generative (i.e., “true”) model whose parameters we estimate via observed data. While this framework provides tractable mathematical models to analyze (4, 5), it does not capture the complexities of modern ML models and

datasets. For example, large language models (LLMs) (6, 7) consist of hundreds of billions of parameters and are trained on web-scale datasets that consist of various modalities, e.g., tables, text, images. In these settings, simple generative models and the assumptions required for valid statistical inference are unlikely to hold. The shortcomings of this traditional framework puts into question conclusions drawn from analyses relying on conventional UQ methods, and call for new tools to quantify uncertainty to support responsible, data-driven decision-making.

To tackle model misspecification ignored by traditional statistical frameworks, statisticians have proposed conformal inference (8–10). Assuming exchangeable data, conformal inference is a distribution- and model-free framework that produces valid prediction sets. Conformal inference has been the subject of intense study over the past decade, and has led to many impressive theoretical results. However, despite the theoretical guarantees, there has been a lack of large-scale empirical validation of conformal methods. A major drawback of conformal methods is that they do not explicitly consider *model selection*. By ignoring model selection, conformal inference allows analysts to use inaccurate models that lead to large and unstable prediction sets. Moreover, as shown later, popular conformal methods do not achieve the desired coverage on subgroups—a key concern for practitioners.

While conformal inference methods have made progress towards tackling model misspecification ignored by traditional frameworks, robust data analysis will require a broader view of UQ. It will require considering uncertainty in every aspect of the data science life cycle (DSLCL), from problem formulation and data collection to exploratory analyses, data cleaning, modeling, interpretation, and even visualization (11). To tackle uncertainty generated from the entire DSLCL, Yu and Kumbier (11) proposed the Predictability-Computability-Stability (PCS) framework for veridical data science, which recognizes that data-driven conclusions are the result of multiple steps and human judgment calls. Examples of these choices are data cleaning methods, model or algorithm choices, hyperparameter tuning, visualization, and more. These choices can have a large effect on analyses, results, and conclusions. For example, Breznau et al. (12) showed that even when given identical *cleaned* datasets, different teams of social scientists made choices that led to opposite conclusions.

The vast number of choices available to researchers \* creates a hidden universe of uncertainty that is often ignored

\* Gelman and Loken (13) refer to this as the “garden of forking paths”

The authors declare no conflicts of interest.

<sup>1</sup> A.A. (Author One) and M.X. (Author Two), R.B. (Author three) contributed equally to this work.

<sup>2</sup> To whom correspondence should be addressed. E-mail: binyu@berkeley.edu

(13, 14). The PCS framework provides a philosophical and practically structured approach to guide these choices by unifying, streamlining, and expanding on the ideas and best practices of statistics and machine learning. Specifically, the PCS framework first requires that ML models used are *predictive*. That is, PCS uses (broadly interpreted) predictability as a proxy to ensure that the ML models being used to estimate uncertainty sufficiently capture reality. PCS formally considers computation both in terms of time/memory complexity, and the use of data-inspired simulations to further augment data analyses. The stability principle (15) requires that conclusions are robust to reasonable perturbations and choices made in the DSLC. Moreover, it requires meticulous documentation of the various choices made. PCS has empirically proven to be effective across a range of challenging domain problems from developmental biology (16), genomics (17), stress-testing clinical decision rules (17), subgroup discovery in causal inference (18), and cardiology (19). Recently, Yu and Barter published a textbook (20) that covers the practice of PCS-driven VDS.

In this paper, **our goal** is to develop a *PCS-driven* uncertainty quantification that goes beyond traditional statistical methods and addresses the shortcomings of conformal prediction methods. We focus on assessing uncertainty that arises from inter-sample variability and model selection, and leave uncertainty from data-cleaning choices and other human judgment calls for future work. Our contributions are as follows.

**PCS-UQ for Regression** We build upon a recently proposed PCS-UQ method for regression in (20). For simplicity, we provide a summary of the original method here, and describe the procedure used in the paper in Section 3. (1) We split data into training and validation sets, and fit multiple models on the training set. In accordance with the “P” principle, we drop candidate algorithms that perform poorly on the validation set. (2) Next, following the stability principle, we fit the filtered set of algorithms on multiple bootstrapped training datasets. These discrete sets of bootstraps create a *pseudo-population* that allows us to assess finite-sample uncertainty. (3) Lastly, we perform a multiplicative calibration that extends interval lengths to achieve the desired coverage. As we show later, this multiplicative calibration allows us to adapt to subgroups in the data, while typical conformal inference calibration methods fail to do so. We perform extensive experiments across 17 real-world regression datasets. Results show that PCS-UQ achieves the desired coverage, while reducing the length of intervals over leading conformal approaches by over 20% on average. Further, we show that PCS-UQ achieves desired coverage across subgroups, while conformal inference approaches fail to do so. Our empirical evaluation also serves as one of the first large-scale comparisons of conformal methods.

**PCS-UQ for Classification** We extend PCS-UQ for regression to multi-class classification. Experiments across 6 multi-class datasets and improve upon conformal approaches by over 20% in prediction set size.

**Approximation Methods for Deep-Learning** PCS UQ for regression and classification requires fitting multiple models across bootstraps, which can be prohibitively expensive for large deep-learning (DL) models. We propose two approximation methods to avoid fitting multiple DL models. Specifically, we only train one model and either apply dropout on activations or add normal noise to the weights to create multiple perturbed

models. Experiments across three computer vision benchmarks show that our approximation schemes maintain the computational efficiency of conformal inference while achieving valid coverage and reducing prediction set sizes by 20%.

**Theoretical Connection to Conformal Inference** We show that a modified PCS-UQ regression procedure is a form of split-conformal inference. Under an exchangeability assumption, we show this modified algorithm achieves the desired coverage.

## 2. Related Work

**Classical Parametric Inference** As discussed, classical statistical approaches consider uncertainty under a fixed generative, often linear model (4, 5). Typically, these approaches focus on proving confidence intervals of parameters of linear models (21–25). Another significant line of work is on post-selection inference, which focuses on statistical inference in the best linear approximation of an underlying regression function (26–29). These methods, while influential, are not the focus of our work since we focus on *prediction intervals* without specifying an underlying generative model.

**Resampling** Resampling to assess uncertainty has been widely studied in statistics. Prominent among resampling methods are the jackknife (30, 31) and bootstrap (32, 33). The bootstrap is a key component of PCS UQ since we use it to assess finite-sample variability for our screened-models. There have also been a number of related papers that use leave-one-out approaches for constructing prediction intervals (34, 35). These methods typically do not address model selection, and require re-fitting the model for every training sample, which renders them infeasible for modern ML models. See Efron and Gong (36) for a comprehensive overview of different approaches.

**Conformal Inference for Regression** Proposed by Vovk (8, 9), conformal prediction for regression has been a major focus of theoretical study. If the underlying data is exchangeable, conformal methods achieve target coverage. Split conformal prediction (10) is the most widely used form of conformal inference, and is based on a simple, efficient idea. First, split the data into two halves, using one half for fitting a model, and the other to calibrate prediction intervals to achieve the desired coverage. Recent work (37, 38) has also combined resampling techniques such as the jackknife and bootstrap with conformal inference to reduce interval lengths. The works discussed above achieve desired coverage on average, but might fail to cover locally, i.e., conditional on covariates. As a result, different methods such as studentized conformal inference (10) and kernel-weighted conformal methods have been proposed to improve local coverage (39). Other extensions include techniques to tackle covariate shift (40), time-series (41), and treatment effect estimation in causal inference (42). Since the conformal literature is too broad to cover comprehensively, we refer readers to (43) for a detailed overview.

**Conformal Classification** Romano et al. (44) extend split conformal inference to the multi-class classification setting. They propose an approach called adaptive prediction sets (APS) which is based on a cumulative likelihood score. For a given sample, APS creates prediction sets by greedily adding classes in order of the predicted probability till the cumulative score of the set reaches a threshold. This threshold is calibrated to achieve the desired coverage. Angelopoulos et al. define a

regularized version of APS called RAPS that has been shown to improve set size in practice (45).

### 3. PCS Regression Prediction Intervals

We detail the PCS-UQ procedure for generating prediction intervals in the regression setting. Our method is closely related to, and builds upon the procedure proposed in chapter 13 of (20); see Appendix D for an overview of this method. Extensions to multi-class classification are discussed in Section 5. This paper does not focus on uncertainty generated from data cleaning choices, and instead focuses on uncertainty resulting from label noise and finite samples. Before detailing our algorithm, we establish necessary notation.

**Notation** We work in the typical supervised regression setting with data  $\mathcal{D} = \{(\mathbf{X}_i, Y_i)\}_{i=1}^n$ , where  $\mathbf{X}_i \in \mathbb{R}^d$ , and  $Y_i \in \mathbb{R}$ . For  $\alpha \in (0, 1)$ , the goal is to produce intervals that achieve  $1 - \alpha$  coverage. That is, we aim to produce prediction intervals that contain the true response for  $1 - \alpha$  proportion of future data points. Let  $f_1 \dots f_M$  denote candidate predictive algorithms, e.g., ordinary least squares (OLS), Random Forests (RFs), etc. Finally, let  $l$  denote a loss, e.g., mean-squared error.

**Step 1: Data-Splitting and Prediction-Check** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Train each algorithm on the training set to obtain fitted models  $\hat{f}_1(\cdot; \mathcal{D}_{\text{tr}}), \dots, \hat{f}_M(\cdot; \mathcal{D}_{\text{tr}})$ . Choose the top- $k$  performing algorithms according to loss  $l$ . Without loss of generality, let  $f_1 \dots f_k$  denote the top- $k$  performing algorithms. The number of models to include,  $k$ , serves as a hyper-parameter in PCS-UQ; we discuss data-driven choices for  $k$  later.

**Step 2: Bootstrapping** Bootstrap the *entire* dataset  $B$  times to obtain bootstrapped samples  $\mathcal{D}^{(1)} \dots \mathcal{D}^{(B)}$ . Fit all algorithms chosen in the previous step on every bootstrapped dataset  $\mathcal{D}^{(b)}$  to obtain bootstrapped models  $\{\hat{f}_j(\cdot; \mathcal{D}^{(b)}), j \in [k], b \in [B]\}$ . For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}$ , let  $T_i \subseteq [B]$  be the set of bootstrap indices such that  $(\mathbf{X}_i, Y_i) \notin \mathcal{D}^{(b)}$ ,

**Step 3: Calibration** First, for each  $(\mathbf{X}_i, Y_i)$ , form a prediction set  $\mathcal{P}_i = \{\hat{f}_j(\mathbf{X}_i; \mathcal{D}^{(b)}); j \in [k], b \in T_i\}$ . Then, form an uncalibrated interval  $[q_{\alpha/2}(\mathcal{P}_i), q_{1-\alpha/2}(\mathcal{P}_i)]$ , where  $q_\beta(S)$  is the  $\beta$  quantile for a set  $S$ . For a multiplicative scaling factor  $\gamma$ , generate a scaled interval

$$\mathcal{I}_i = \left[ q_{0.5}(\mathcal{P}_i) - \gamma \times (q_{0.5}(\mathcal{P}_i) - q_{\alpha/2}(\mathcal{P}_i)), \right. \\ \left. q_{0.5}(\mathcal{P}_i) + \gamma \times (q_{1-\alpha/2}(\mathcal{P}_i) - q_{0.5}(\mathcal{P}_i)) \right].$$

We choose the scaling factor  $\hat{\gamma}$  such that we achieve  $1 - \alpha$  coverage on the data  $\mathcal{D}$ .

**Step 4: Generating PCS Prediction Interval for Test-Point** For a new test point  $\mathbf{X}$ , let  $\mathcal{P} = \{\hat{f}_j(\mathbf{X}; \mathcal{D}^{(b)}); j \in [k], b \in B\}$ . Then, we produce prediction interval

$$\mathcal{I} = \left[ q_{0.5}(\mathcal{P}) - \gamma \times (q_{0.5}(\mathcal{P}) - q_{\alpha/2}(\mathcal{P})), \right. \\ \left. q_{0.5}(\mathcal{P}) + \gamma \times (q_{1-\alpha/2}(\mathcal{P}) - q_{0.5}(\mathcal{P})) \right]. \quad [1]$$

The PCS UQ algorithm consists of a few key steps that contribute to its strong performance; see Section 4. We discuss the motivation behind these design choices and how they compare to conformal methods.

1. **Prediction-check.** Unlike current conformal prediction methods, PCS *explicitly* includes model selection by screening out models with poor prediction performance. This is done in order to ensure that uncertainty is only assessed using algorithms that sufficiently capture the underlying data-generating process. Experiments in Section 9 show that excluding models with poor predictive performance lead to significantly smaller intervals.
2. **Assessing local uncertainty via bootstraps** To assess uncertainty from finite-samples, PCS simulates the data-collection process by constructing a set of perturbed datasets via the bootstrap. This universe of discrete datasets creates a *pseudo-population* that allows us to assess finite-sample uncertainty. On the other hand, split-conformal methods only utilize one random train-calibration split which creates instability since results often vary greatly based on split.
3. **Data-efficiency via out-of-bag samples** Traditional split-conformal methods, and the PCS method proposed in (20) propose a data-split which leads to less data being used for both fitting models, and calibration. In this paper, we use out-of-bag (OOB) samples to utilize samples efficiently. Results in Appendix C show use of OOB samples reduces interval length by  $\approx 5\%$  on average.
4. **Multiplicative calibration** Instead of additive calibration (i.e., expanding intervals by a fixed constant) as is common in conformal inference, we do so multiplicatively. Since additive calibration expands intervals for every sample by a fixed length, it does not adjust the interval according to how uncertain the model(s) prediction is for that sample. Instead, multiplicative calibration expands intervals more aggressively for samples with high uncertainty which can improve coverage for poorly-represented subgroups. Experiments in Section 9 and Appendix S2 show that replacing multiplicative with additive calibration in the PCS procedure leads to poorer subgroup coverage across datasets.

**Hyper-parameter choices** PCS-UQ depends on two hyper-parameters,  $k$  and  $B$ . We use the following procedure to choose  $k$ . Set  $k = 3$  from the prediction screening step. Then, choose the combination of models from this screened set that lead to the smallest prediction intervals after calibration. We choose  $B$  to be as large as computationally feasible.

## 4. Regression Experiments

**A. Experimental Set-up.** This section details the experimental set-up for our regression experiments displayed below.

**Datasets** We use 17 regression datasets commonly found in tabular benchmarks (46). These datasets reflect a range of sample sizes and dimensions. To avoid uncertainty associated with data-cleaning, our datasets do not contain any missing values. We use 80% to train and fit various UQ methods, and 20% as our test set.

**Baseline Methods** We compare PCS against three popular conformal regression methods: split conformal regression (10), studentized conformal regression (10), and the majority vote (63) procedure. Split and studentized conformal regression require the choice of a particular ML model, while majority

Name	Samples	Features
Energy (47)	768	10
Concrete (48)	1030	8
Insurance (49)	1338	8
Airfoil (50)	1503	5
Debutanizer (51)	2394	7
QSAR (52)	5742	500
Parkinsons (53)	5875	18
Kin8nm (54)	8192	8
Computer (55)	8192	21
Powerplant (56)	9568	4
Naval (57)	11934	24
Miami (58)	13932	28
Elevator (59)	16599	18
CA Housing (60)	20640	8
Superconductor (61)	21263	79
Protein (62)	45730	9
Diamond (46)	53940	23

Table 1. Datasets used for regression experiments.

vote is an ensemble method. We generate prediction intervals for both methods with the following ML models: Ordinary Least Squares (OLS), Ridge regression (64), Lasso (65), Elastic Net (66), Random Forests (67), AdaBoost (68), XGBoost (69), and a 1-hidden layer multi-layer perceptron (MLP). We choose regularization parameters in Ridge, Lasso, and Elastic Net via three-fold cross-validation. For other ML models, we use the default hyper-parameters from `scikit-learn` (70). For majority vote, we combine prediction intervals from all models listed above.

**PCS Hyper-parameters** We use all models listed above as candidate model and use  $B = 1000$  bootstraps. The number of models  $k$  is chosen as discussed in Section 3.

**Metrics** We measure coverage and width of intervals on the test set. We aim for 90% coverage, i.e., we set  $\alpha = 0.1$ . Interval width is normalized by the range of the responses on the test set. Results are averaged across 10 train-test splits.

**B. Results.** This section establishes the empirical results for our experiments described in the previous section. Due to the large number of conformal methods we ran, we only report results for the split conformal trained with XGBoost, and Studentized conformal regression with Random Forests<sup>†</sup>. These models were chosen since they achieve the desired coverage and have the smallest average width across datasets. This information is *unavailable* in practice since we do not know which estimator will achieve the smallest intervals.

**All Methods Achieve Desired Coverage** Test-set coverage is reported for all methods and datasets in Table S1. All conformal regression methods, and PCS achieve the desired 90% coverage for every dataset.

**PCS Significantly Reduces Interval Size** In Figure 1, we display the average interval size of PCS as compared to split conformal regression (XGBoost), Studentized conformal regression (Random Forest), and Majority Vote. We also plot the distribution of the percentage reduction in width PCS achieves across all

17 datasets. PCS significantly improves upon conformal regression approaches and reduces length by over 20% on average. Moreover, PCS often has smaller error bars than conformal methods, highlighting its stability. We once again emphasize that we compare to conformal regression methods that achieve the smallest average width on the test set, which is impossible to determine pre-hoc.

**PCS adapts to subgroup structure** While the results above show PCS produces efficient intervals that achieve desired coverage on average, practitioners are often interested in determining if intervals are valid for heterogeneous subgroups. For each dataset in Table 1, we construct natural subgroups (details in Appendix B) and evaluate the coverage and width of each UQ method across subgroups. Results for the Miami housing dataset (58) are shown in Fig. 2, where we formed subgroups based on the square footage of the house. Split conformal (XGBoost) and majority vote do not achieve the desired coverage across subgroups. PCS, and Studentized conformal (Random Forest) adapt the width of their intervals to achieve valid coverage; however, PCS is significantly more efficient. Further, PCS efficiently achieves coverage on subgroups that were *not* pre-specified, which demonstrates its robustness. We observe similar results for our 16 other datasets; see Appendix B.

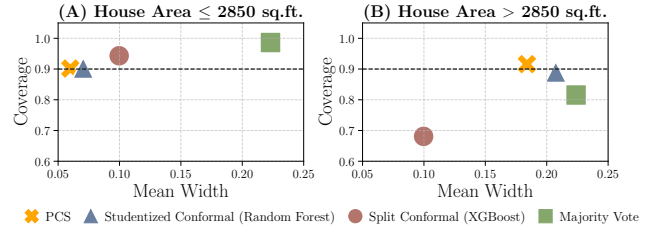


Fig. 2. Coverage and width for PCS, and conformal regression approaches on subgroups in the Miami Housing dataset (58). Panels (A) and (B) demonstrate performance on subgroups formed by square footage of the house. PCS adapts width of intervals to maintain coverage across subgroups. Other conformal methods either do not achieve subgroup coverage or have larger width.

## 5. PCS-UQ for Multi-Class Classification

We detail the PCS-UQ procedure for generating prediction sets in the multi-class classification setting.

**Notation** We adopt much of the same notation as in Section 3, except that we assume responses belong to one of  $C$  classes, i.e.,  $y \in \mathcal{Y} = \{1, \dots, C\}$ . Additionally, let  $\hat{f}^{(c)}(\cdot)$  be the predicted probability that a sample belongs to class  $c \in \mathcal{Y}$ . Lastly, for numbers  $a_1, a_2, \dots, a_l$ , let  $\pi$  be the permutation of the indices that sorts the numbers in descending order. That is,  $a_{\pi(1)} > a_{\pi(2)} > \dots > a_{\pi(l)}$ .

**Step 1: Data-Splitting and Prediction-Check** Repeat step 1 from Section 3.

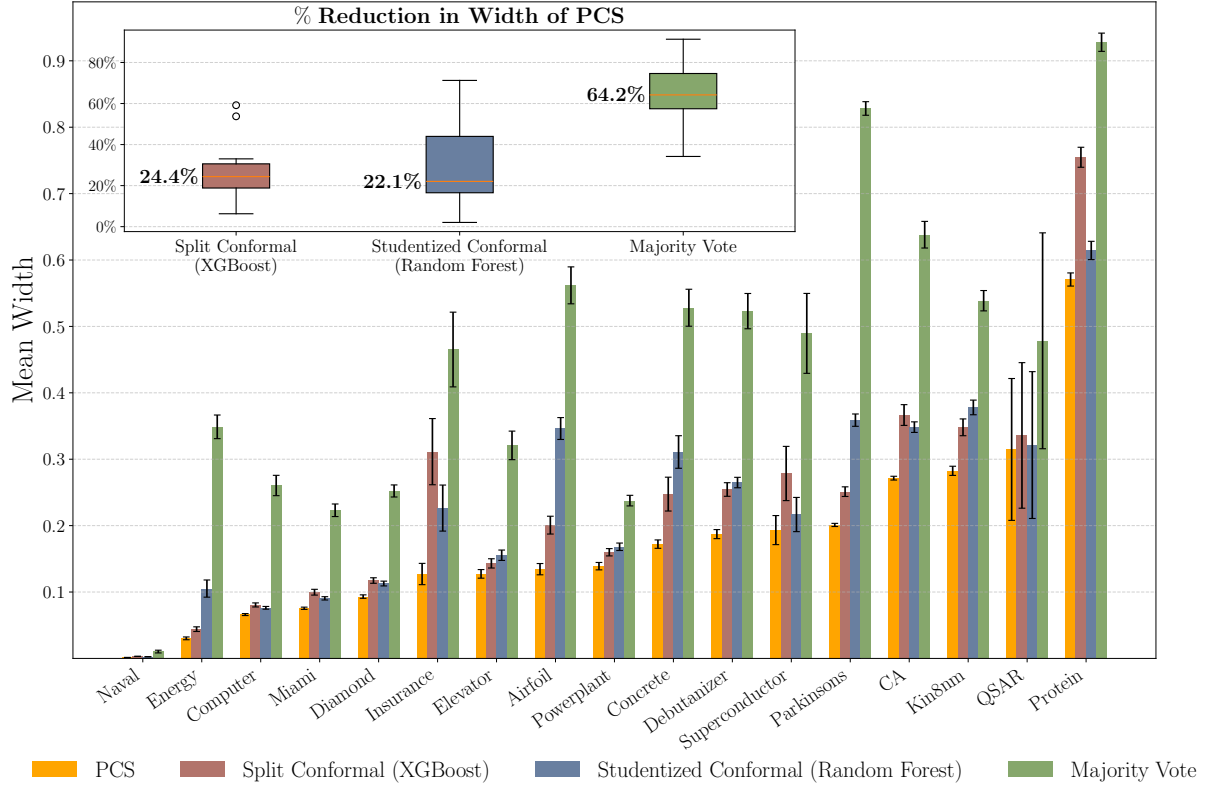
**Step 2: Bootstrapping** Repeat step 2 from Section 3.

**Step 3: Generate Uncalibrated Predictions** First, for each  $(\mathbf{X}_i, Y_i)$ , compute the mean prediction across all bootstrapped models. That is, for class  $c \in \mathcal{Y}$ , let

$$\hat{y}_i^{(c)} = \frac{1}{|T_i|k} \sum_{j \in [k]} \sum_{b \in T_i} \hat{f}_j^{(c)}(\mathbf{x}_i; \mathcal{D}^{(b)}), \quad [2]$$

<sup>†</sup> Results for all models can be found on our Github [https://github.com/aagarwal1996/PCS\\_UQ](https://github.com/aagarwal1996/PCS_UQ)





**Fig. 1.** Comparison of PCS against Majority Vote, and two best performing conformal methods: Split conformal (XGBoost), Studentized conformal (Random Forest) across 17 datasets. We display the distribution of % improvement of PCS in the inset plot. PCS displays a significant improvement over conformal approaches.

where recall that  $T_i$  denotes bootstrap indices where  $(\mathbf{X}_i, Y_i)$  is out-of-bag. This is similar to the ensemble method proposed in chapter 13 of Yu and Barter (20).

**Step 4: Calibration** We follow the adaptive prediction set (APS) procedure introduced in (44). Obtain APS score  $S_i = \sum_{c=1}^r \hat{y}_i^{\pi(c)}$ , where  $\pi(r) = Y_i$ . For  $\mathcal{S} = \{S_i : i \in \mathcal{D}\}$ , let  $q$  denote the  $1 - \alpha$  quantile of  $\mathcal{S}$ .

**Step 5: Generating PCS Prediction Sets for Test Point** For a new test point  $\mathbf{X}$ , produce the prediction set

$$\mathcal{S} = \{\pi(1), \pi(2), \dots, \pi(r)\}, \text{ where } r = \min \left\{ t : \sum_{c=1}^t \hat{y}^{\pi(c)} \geq q \right\}$$

## 6. Multi-Class Classification Experiments

**A. Experimental Set-up.** This section details the experimental set-up for our multi-class classification experiments.

**Datasets** We use 6 datasets commonly found in tabular benchmarks (46). These datasets reflect a range of sample-sizes, and dimensions, and number of classes. We use 80% to train and fit various UQ methods, and 20% as our test-set.

**Baseline Methods** We compare PCS against four popular conformal multi-class classification methods: Adaptive Prediction Sets (APS) (44), Regularized Adaptive Prediction Sets (RAPS) (45), TopK, and Majority Vote (63). Apart from Majority Vote, we use the implementation of all conformal methods from the software package `MAPIE` (77). We generate prediction

Name	Samples	Features	Classes
Language (71)	1000	19	30
Yeast (72)	1484	8	10
Isolet (73)	7797	613	26
Cover Type (74)	10000	13	100
Chess (75)	28056	34	18
Dionis (76)	30000	60	355

**Table 2.** Datasets used for multi-class classification experiments.

intervals for both methods with the following ML models:  $\ell_2$  regularized Logistic Regression, Random Forests (67), Adaboost (68), XGBoost (69), and a 1-hidden layer multi-layer perceptron (MLP). We choose regularization parameters in  $\ell_2$  regularized Logistic Regression via 3-fold cross-validation. For other ML models, we use the default hyper-parameters from `scikit-learn` (70). For Majority Vote, we combine prediction intervals from all models listed above.

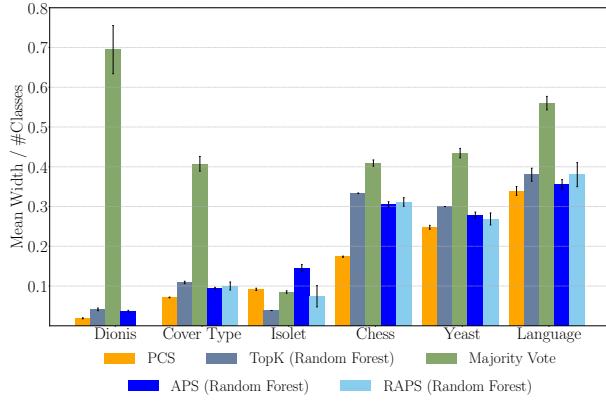
**PCS Hyper-parameters** We use all models listed above as candidate models, and choose  $k$  as we did in the regression experiments; see Section 3 for a description. We generate intervals using  $B = 1000$  bootstraps.

**Metrics** We measure coverage and average size of prediction sets on the test set. We aim for 90% coverage, i.e., we set  $\alpha = 0.1$ . Size of prediction sets is normalized by the number of classes,  $C$ . Results are averaged across 10 train-test splits.

**B. Results.** This section details results for experiments described in previous results. For APS, RAPS, and TopK, we report performance using Random Forests as the estimator since these estimators achieve coverage, and have the smallest width on average across our 6 datasets. We emphasize that we choose the best-performing estimators for conformal methods — information that is unavailable in practice.

**All Methods Achieve Desired Coverage** Test-set coverage is reported for all methods and datasets in Table S1. All methods achieve the desired coverage.

**PCS Produces Smaller Intervals than Conformal Approaches** Fig. 3 displays average prediction set size for all methods<sup>‡</sup>. PCS outperforms all methods on 5 out of 6 dataset, apart from Isolet. The table below summarizes the average reduction in width by PCS over all conformal methods.



**Fig. 3.** Comparison of average prediction set size of PCS against best-performing conformal methods. PCS significantly reduces width across 5 out of 6 datasets.

TopK (RF)	APS (RF)	RAPS (RF)	Majority Vote
21.6%	22.5%	30.8%	63.6%

**Table 3.** Average % reduction in width by PCS over best-performing conformal approaches across 6 multi-class classification datasets.

## 7. PCS Uncertainty Quantification for Deep-Learning

While PCS significantly reduces width in our experiments, training multiple bootstrap models can be prohibitively expensive for large deep-learning models. In this section, we discuss computationally efficient methods to generate prediction intervals for deep-learning models via PCS and experimental results on large-scale deep-learning datasets.

**A. Approximate PCS UQ.** Instead of training DL models across  $B$  different bootstrapped datasets, we proceed as follows. First, we perform a simple train-calibration data-split and train a *single* DL model on the training set  $\mathcal{D}_{\text{train}}$ . Throughout this description, we assume that the DL model achieves sufficient predictive accuracy on the calibration set. If not, we recommend trying a different DL architecture or training algorithm.

<sup>‡</sup>The RAPS implementation in MAPIE is unable to be run for Dionis due to the large number of imbalanced classes.

This emphasizes that establishing strong predictability is key for trustworthy UQ. Next, we create  $B$  perturbed DL models as follows: (1) *Weighted Monte-Carlo Dropout*. We create  $B$  perturbed models by randomly dropping out nodes in a DL model (78). The probability of drop-out is set to be proportional to the activation. (2) *Additive Noise Perturbation*. Create  $B$  perturbed models by adding mean-zero Gaussian noise to the weights. The noise variance of the added noise is set to be the initialization variance.

**B. Experimental Results.** We perform experiments comparing the original PCS UQ for multi-class classification, our approximation methods, and the conformal inference methods on three computer vision benchmarks,

**Datasets** We use the three following standard computer vision benchmarks. Descriptions of the datasets, and details of training, validation, and test splits are in Appendix S5 are as follows. Summary statistics of the datasets are as follows.

Name	Samples	Classes
CIFAR-100 (79)	60000	100
Caltech Birds (80)	11788	200
Imagenet-Small (81)	100000	200

**Table 4.** Datasets used for deep-learning classification experiments.

**Model Details** For all datasets, we use a Res-net 18 (82).

**UQ Methods** We compare PCS-UQ for multi-class classification described in Section 5, and the approximation methods described above. For the original multi-class PCS-UQ, we use  $B = 100$  bootstraps. Since PCS-UQ does not require a separate validation set due to the use of OOB samples, we combine training and validation sets. For the approximation methods described above, we create  $B = 1000$  models.

**Metrics** We aim for 90% coverage, and measure average prediction set size on the test-set. Further, we measure the time taken (rounded to the nearest minute) to produce prediction sets for each UQ method. Results are averaged across 10 train-test splits.

**Results** The results for each dataset are presented as follows. As seen in the table below, all UQ methods achieve the desired coverage. Original PCS-UQ improves upon conformal methods by producing prediction sets that are 26% smaller on average. Both PCS approximation methods improve upon conformal methods by approximately 20%, but do not match the performance of the original PCS method. However, the approximation schemes are approximately 30 to 100× faster than the original PCS method. As such, the approximation methods strike a balance between computational efficiency and improving the size of prediction sets.

## 8. Theoretical Connection to Conformal Inference

This section provides theoretical results that show a modified PCS-UQ procedure for regression achieves valid coverage if the data is exchangeable. We establish that this modified PCS-UQ procedure can be regarded as a form of split conformal inference, and thus achieves  $1 - \alpha$  coverage. We first summarize

	Method	CIFAR 100			Caltech Birds			Imagenet-Small		
		Cov.	Av. Size	Time	Cov.	Av. Size	Time	Cov.	Av. Size	Time
PCS	APS	90.6	6.8	2	91.4	16.6	2	90.6	14.4	3
	RAPS	90.8	6.5	2	91.1	12.8	2	90.5	11.2	3
	TopK	92.3	8.5	2	91.6	17.6	2	92.3	13	3
	Original	90.5	<b>3.7</b>	350	92.4	<b>8.3</b>	100	90.4	<b>8.8</b>	500
	Dropout	91.3	4.4	4	92.1	9.8	3	90.6	9.8	5
	Noise	91.1	4.2	3	91.9	9.4	3	90.5	9.6	5

**Table 5. Coverage (%), average prediction set size, and runtime (minutes) across multiple computer vision benchmarks. PCS-UQ out-performs conformal approaches in terms of size of prediction sets. Both proposed approximation schemes strike a balance between computational efficiency and size of prediction sets.**

the modified PCS-UQ procedure, and then provide an informal discussion of our theoretical results. Appendix S6 details the modified procedure, formal result, and proof.

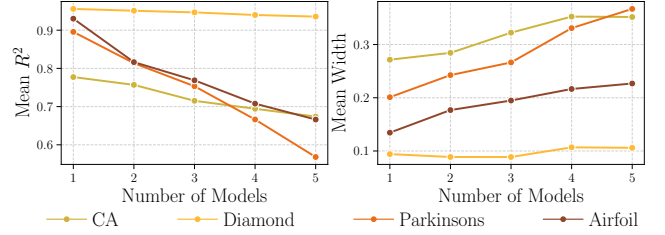
**Summary of Modified PCS Procedure** The algorithm proposed in Section 3 uses the validation data for screening prediction algorithms and calibration. Doing so makes it difficult to establish theoretical guarantees that the produced interval is statistically valid. The modified PCS-UQ procedure overcomes this issue by randomly splitting the data into a training, validation, and calibration set. The training and validation data are used for prediction-check and fitting the bootstrapped models, while the calibration set is used *solely* to learn the scaling factor  $\gamma$ . This ensures statistical validity at the cost of lower sample efficiency. A detailed description of this modified algorithm is in Appendix S6. Next, we describe our theoretical results for this modified procedure.

**Theoretical Results** Our result first establishes that this modified procedure can be regarded as a form of split conformal inference. Specifically, we show that the scaling factor  $\gamma$  can be regarded as a valid conformal “score function”. The conformal score function measures the quality of the prediction, e.g., residuals are typically used in regression as a valid conformal score. In our setting, a larger  $\gamma$  indicates poorer prediction. Given this connection, we utilize previous results that any prediction interval formed using a valid score function achieves the desired coverage. The formal result is presented in Theorem 1.

## 9. Ablation Experiments

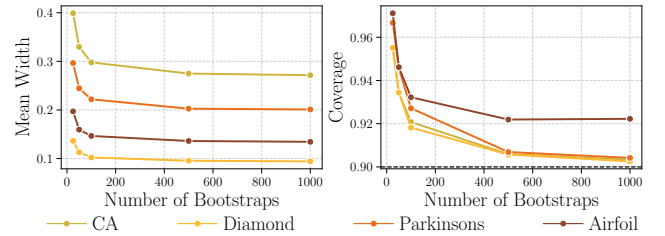
As discussed in Section 3, generating PCS prediction intervals consists of a few key steps: prediction-checking, bootstrapping, and multiplicative calibration. We perform a number of ablation experiments to demonstrate the utility of each of these steps as follows.

**Effect of prediction-screen** We vary the number of screened models (i.e.,  $k$ ) and measure average width, and  $R^2$  on the test set for 4 datasets. Results are displayed in Figure 4, which shows that including poor-performing models leads to larger intervals – highlighting the importance of screening models via their prediction performance.



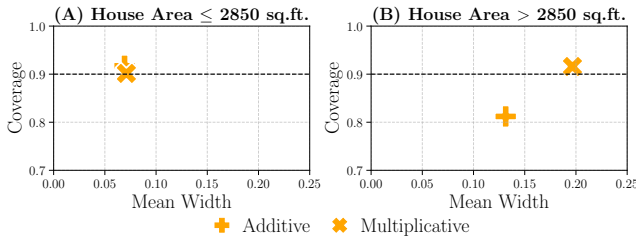
**Fig. 4.** Performance of PCS with varying number of selected models over 4 datasets. The left panel displays the average  $R^2$  of selected models; the right panel displays the average interval width. As the number of selected model increases, the  $R^2$  decreases while the interval width increases.

**Varying number of bootstraps** A key step in the PCS procedure is creating a pseudo-universe of datasets via the bootstrap. In Figure 5, we display the average interval size and coverage as we vary the number of bootstraps. Performance stabilizes after 100 bootstraps. Bootstrapping allows one to simulate and capture uncertainty during the data collection process.



**Fig. 5.** Performance of PCS-UQ with varying number of bootstraps over 4 datasets. The left panel displays the average interval width; the right panel displays the coverage. Both metrics stabilize after 100 bootstraps.

**Multiplicative Calibration** To investigate the effectiveness of multiplicative calibration for subgroup coverage, we replace multiplicative calibration with additive calibration. That is, we enlarge intervals by adding a fixed constant to both ends, instead of scaling the interval widths multiplicatively. We examine subgroup coverage for the Miami housing dataset (58) as in Section 4. Fig. 6 shows that additive calibration is unable to achieve target coverage for houses larger than 2850 square feet. Similar results hold for other datasets, as seen in Appendix S2. Additive calibration is unable to enlarge intervals sufficiently for samples with high uncertainty, while multiplicative scaling does so effectively.



**Fig. 6.** Subgroup coverage of additive and multiplicative calibration on the Miami housing dataset [58]. Additive calibration is unable to achieve target coverage for large houses, while multiplicative calibration adjusts length to do so effectively.

## 10. Discussion

Our approach builds upon key PCS principles to develop prediction intervals. Extensive empirical comparisons of PCS-UQ to conformal methods show we reduce the size of prediction sets by over 20% across a variety of settings. Our paper also establishes theoretical connections to conformal inference that might be of independent interest. While our paper takes a step towards establishing PCS-driven UQ, there are many extensions and improvements to explore for future work. We detail some of these as follows.

**Uncertainty from Data-Cleaning & Judgment Calls** This paper only focuses on uncertainty due to inter-sample variability and model selection. As discussed in Section 1, data cleaning choices and other judgment calls due to inter-researcher variability can lead to drastically different conclusions. Interesting future work is finding approaches to assess uncertainty from every part of the DSLC to create a more stable UQ method.

**Extension to Binary Classification** Our approach for classification only produces prediction sets that are often unsuitable for binary classification. In the binary setting, producing intervals that contain the true  $\mathbb{P}(Y = 1|\mathbf{X})$  is often more relevant to practitioners. As a simple example, producing intervals that state there is a 40% – 60% chance of rain is more instructive than a prediction set that consists of both rain and no rain. Constructing intervals for the underlying class probability is difficult because we do not observe empirical class probabilities, but only binary labels. Observations of class labels also makes evaluation of probability intervals challenging.

**Extension to LLMs and Generative Models** An exciting future direction is using PCS to assess the uncertainty of LLMs and other generative models. Doing so requires defining appropriate notions of prediction sets and coverage. We believe that robust UQ for LLMs and generative models has the potential to improve hallucinations and factuality (83).

- Ioannidis JP (2005) Why most published research findings are false. *PLoS medicine* 2(8):e124.
- Begley CG, Ellis LM (2012) Raise standards for preclinical cancer research. *Nature* 483(7391):531–533.
- Collaboration OS (2015) Estimating the reproducibility of psychological science. *Science* 349(6251):aac4716.
- Cox DR (2006) *Principles of statistical inference*. (Cambridge university press).
- Reid N, Cox DR (2015) On some principles of statistical inference. *International Statistical Review* 83(2):293–308.
- Vaswani A, et al. (2017) Attention is all you need. *Advances in neural information processing systems* 30.
- Radford A, Narasimhan K, Salimans T, Sutskever I, et al. (2018) Improving language understanding by generative pre-training.
- Vovk V, Gammerman A, Shafer G (2005) *Algorithmic learning in a random world*. (Springer) Vol. 29.
- Shafer G, Vovk V (2007) A tutorial on conformal prediction.

- Lei J, G'Sell M, Rinaldo A, Tibshirani RJ, Wasserman L (2018) Distribution-free predictive inference for regression. *Journal of the American Statistical Association* 113(523):1094–1111.
- Yu B (2020) Veridical data science in *Proceedings of the 13th international conference on web search and data mining*. pp. 4–5.
- Breznau N, et al. (2022) Observing many researchers using the same data and hypothesis reveals a hidden universe of uncertainty. *Proceedings of the National Academy of Sciences* 119(44):e2203150119.
- Gelman A, Loken E (2013) The garden of forking paths: Why multiple comparisons can be a problem, even when there is no “fishing expedition” or “p-hacking” and the research hypothesis was posited ahead of time. *Department of Statistics, Columbia University* 348(1-17):3.
- Simmons JP, Nelson LD, Simonsohn U (2011) False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological science* 22(11):1359–1366.
- Yu B (2013) Stability. *Bernoulli* 19(4):1484–1500.
- Wu S, et al. (2016) Stability-driven nonnegative matrix factorization to interpret spatial gene expression and build local gene networks. *Proceedings of the National Academy of Sciences* 113(16):4290–4295.
- Basu S, Kumbier K, Brown JB, Yu B (2018) Iterative random forests to discover predictive and stable high-order interactions. *Proceedings of the National Academy of Sciences* 115(8):1943–1948.
- Dwivedi R, et al. (2020) Stable discovery of interpretable subgroups via calibration in causal studies. *International Statistical Review* 88:S135–S178.
- Wang Q, et al. (2023) Epistasis regulates genetic control of cardiac hypertrophy. *Research square* pp. rs–3.
- Yu B, Barter RL (2024) *Veridical data science: The practice of responsible data analysis and decision making*. (MIT Press).
- Belloni A, Chen D, Chernozhukov V, Hansen C (2012) Sparse models and methods for optimal instruments with an application to eminent domain. *Econometrica* 80(6):2369–2429.
- Bühlmann P (2013) Statistical significance in high-dimensional linear models. *Bernoulli* 19(4).
- Zhang CH, Zhang SS (2014) Confidence intervals for low dimensional parameters in high dimensional linear models. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 76(1):217–242.
- van de Geer S, Bühlmann P, Ritov Y, Dezeure R (2014) On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics* 42(3).
- Javanmard A, Montanari A (2014) Confidence intervals and hypothesis testing for high-dimensional regression.
- Fithian W, Sun D, Taylor J (2014) Optimal inference after model selection. *arXiv preprint arXiv:1410.2597*.
- Tibshirani RJ, Taylor J, Lockhart R, Tibshirani R (2016) Exact post-selection inference for sequential regression procedures. *Journal of the American Statistical Association* 111(514):600–620.
- Lee JD, Sun DL, Sun Y, Taylor JE (2016) Exact post-selection inference, with application to the lasso.
- Tian X, Taylor J (2017) Asymptotics of selective inference. *Scandinavian Journal of Statistics* 44(2):480–499.
- Quenouille MH (1949) Approximate tests of correlation in time-series 3 in *Mathematical Proceedings of the Cambridge Philosophical Society*. (Cambridge University Press), Vol. 45, pp. 483–484.
- Quinlan JR (1986) Induction of decision trees. *Machine learning* 1(1):81–106.
- Efron B (1992) Bootstrap methods: another look at the jackknife in *Breakthroughs in statistics: Methodology and distribution*. (Springer), pp. 569–593.
- Stine RA (1985) Bootstrap prediction intervals for regression. *Journal of the American Statistical Association* 80(392):1026–1031.
- Stone M (1974) Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)* 36(2):111–133.
- Butler R, Rothman ED (1980) Predictive intervals based on reuse of the sample. *Journal of the American Statistical Association* 75(372):881–889.
- Efron B, Gong G (1983) A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistician* 37(1):36–48.
- Barber RF, Candès EJ, Ramdas A, Tibshirani RJ (2021) Predictive inference with the jackknife+. *The Annals of Statistics* 49(1):486–507.
- Kim B, Xu C, Barber R (2020) Predictive inference is free with the jackknife+-after-bootstrap. *Advances in Neural Information Processing Systems* 33:4138–4149.
- Guan L (2020) Conformal prediction with localization.
- Tibshirani RJ, Foygel Barber R, Candès E, Ramdas A (2019) Conformal prediction under covariate shift. *Advances in neural information processing systems* 32.
- Angelopoulos AN, Bates S, Jordan M, Malik J (2021) Uncertainty sets for image classifiers using conformal prediction in *International Conference on Learning Representations*.
- Lei L, Candès EJ (2021) Conformal inference of counterfactuals and individual treatment effects. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 83(5):911–938.
- Angelino E, Larus-Stone N, Alabi D, Seltzer M, Rudin C (2017) Learning certifiably optimal rule lists for categorical data. *arXiv preprint arXiv:1704.01701*.
- Romano Y, Sesia M, Candès E (2020) Classification with valid and adaptive coverage. *Advances in neural information processing systems* 33:3581–3591.
- Angelopoulos A, Bates S, Malik J, Jordan MI (2022) Uncertainty sets for image classifiers using conformal prediction.
- Feurer M, et al. (2021) Openml-python: an extensible python api for openml. *Journal of Machine Learning Research* 22(100):1–5.
- Tsanas A, Xifara A (2012) Energy Efficiency (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C51307>.
- Yeh IC (1998) Concrete Compressive Strength (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C5PK67>.
- Ali M (2020) *PyCaret: An open source, low-code machine learning library in Python*. PyCaret



version 1.0.

50. Brooks T, Pope D, Marcolini M (1989) Airfoil Self-Noise (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C5VW2C>.
51. Fortuna L, Graziani S, Rizzo A, Xibilia MG (2007) *Soft Sensors for Monitoring and Control of Industrial Processes*, Advances in Industrial Control. (Springer London, London).
52. Olier I, et al. (2015) QSAR-TID-11 (OpenML Datasets). URL: <https://www.openml.org/search?type=data&id=3050&sort=runs&status=active>.
53. Tsanas A, Little M (2009) Parkinsons Telemonitoring (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C5ZS3N>.
54. Torgo L (2014) Kinematics (Collection of Regression Datasets). URL: <https://www.dcc.fc.up.pt/ltorgo/Regression/kin.html>.
55. Torgo L (2014) Computer Activity (Collection of Regression Datasets). URL: <https://www.dcc.fc.up.pt/ltorgo/Regression/DataSets.html>.
56. Tfekci P, Kaya H (2014) Combined Cycle Power Plant (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C5002N>.
57. Coraddu A, et al. (2014) Condition Based Maintenance of Naval Propulsion Plants (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C5K31K>.
58. Bourassa S (2021) MiamiHousing2016 (OpenML Datasets). URL: <https://www.openml.org/search?type=data&status=active&id=43093&sort=runs>.
59. Torgo L (2014) Elevators (Collection of Regression Datasets). URL: <https://www.dcc.fc.up.pt/ltorgo/Regression/elevators.html>.
60. Kelley Pace R, Barry R (1997) Sparse spatial autoregressions. *Statistics & Probability Letters* 33(3):291–297.
61. Hamidieh K (2018) Superconductivity Data (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C53P47>.
62. Rana P (2013) Physicochemical Properties of Protein Tertiary Structure (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C5QW3H>.
63. Gasparin M, Ramdas A (2024) Merging uncertainty sets via majority vote.
64. Hoerl AE, Kennard RW (1970) Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* 12(1):55–67.
65. Tibshirani R (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* pp. 267–288.
66. Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67(2):301–320.
67. Breiman L (2001) Random forests. *Machine learning* 45(1):5–32.
68. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55(1):119–139.
69. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. pp. 785–794.
70. Pedregosa F, et al. (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.
71. Collins J (2003) collins (OpenML Datasets). URL: <https://www.openml.org/search?type=data&status=active&id=40971&sort=runs>.
72. Horton P, Nakai K (1996) A probabilistic classification system for predicting the cellular localization sites of proteins. *Proceedings. International Conference on Intelligent Systems for Molecular Biology* 4:109–115.
73. Cole R, Fanty M (1991) ISOLET (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C51G69>.
74. Blackard J (1998) Covertypes (UCI Machine Learning Repository). DOI: <https://doi.org/10.24432/C50K5N>.
75. Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S (2011) Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Multiple Valued Log. Soft Comput.* 17:255–287.
76. Guyon I, et al. (2019) Analysis of the autotml challenge series 2015-2018 in *AutoML*, Springer series on Challenges in Machine Learning.
77. Cordier T, et al. (2023) Flexible and Systematic Uncertainty Estimation with Conformal Prediction via the MAPIE library in *Conformal and Probabilistic Prediction with Applications*.
78. Gal Y, Ghahramani Z (2016) Dropout as a bayesian approximation: Representing model uncertainty in deep learning in *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, eds. Balcan MF, Weinberger KQ. (PMLR, New York, New York, USA), Vol. 48, pp. 1050–1059.
79. Krizhevsky A, Hinton G, , et al. (2009) Learning multiple layers of features from tiny images.
80. Welinder P, et al. (2010) Caltech-UCSD Birds 200, (California Institute of Technology), Technical Report CNS-TR-2010-001.
81. Le Y, Yang X (2015) Tiny imagenet visual recognition challenge. *CS 231N* 7(7):3.
82. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition in *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 770–778.
83. Cherian JJ, Gibbs I, Candès EJ (2024) Large language model validity via enhanced conformal prediction methods.
84. Deng J, et al. (2009) ImageNet: A Large-Scale Hierarchical Image Database in *CVPR09*.

## S1. Overview of Uncertainty Quantification Methods for Regression

**A. Split Conformal Regression.** We describe the Split Conformal procedure from (10). We use the same notation as established in Section 3.

**Step 1: Data-Splitting and Model Training** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Fit algorithm  $f$  on training set to obtain fitted model  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$ .

**Step 2: Calibration** For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , make prediction using  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$  and obtain conformal score  $S_i = |Y_i - \hat{f}(\mathbf{X}_i; \mathcal{D}_{\text{tr}})|$ . Let  $q$  be the  $1 - \alpha$  quantile of the set  $\{S_i : i \in |\mathcal{D}_{\text{val}}|\}$ .

**Step 3: Generate Split Conformal Prediction Interval** For a new test point  $\mathbf{X}$ , produce the prediction interval

$$\mathcal{I} = [\hat{f}(\mathbf{X}; \mathcal{D}_{\text{tr}}) - q, \hat{f}(\mathbf{X}; \mathcal{D}_{\text{tr}}) + q]$$

**B. Studentized Conformal Regression.** We describe the Studentized Conformal procedure from (10). We use the same notation as established in Section 3.

**Step 1: Data-Splitting and Model Training** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Fit algorithm  $f$  on training set to obtain fitted model  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$ . Then let  $\mathcal{D}_{\text{tr}}^{\text{res}} = \{(\mathbf{X}_i, |Y_i - \hat{f}(\mathbf{X}_i; \mathcal{D}_{\text{tr}})|) : (\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{tr}}\}$  be the training set with residuals as the response. Fit algorithm  $\sigma$  on  $\mathcal{D}_{\text{tr}}^{\text{res}}$  to obtain fitted model  $\hat{\sigma}(\cdot; \mathcal{D}_{\text{tr}}^{\text{res}})$ .

**Step 2: Calibration** For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , make predictions using  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$ ,  $\hat{\sigma}(\cdot; \mathcal{D}_{\text{tr}}^{\text{res}})$ ; obtain conformal score

$$S_i = \frac{|Y_i - \hat{f}(\mathbf{X}_i; \mathcal{D}_{\text{tr}})|}{\hat{\sigma}(\mathbf{X}_i; \mathcal{D}_{\text{tr}}^{\text{res}})}$$

Let  $q$  be the  $1 - \alpha$  quantile of the set  $\{S_i : i \in |\mathcal{D}_{\text{val}}|\}$ .

**Step 3: Generate Studentized Conformal Prediction Interval** For a new test point  $\mathbf{X}$ , produce the prediction interval

$$\mathcal{I} = [\hat{f}(\mathbf{X}; \mathcal{D}_{\text{tr}}) - q \times \hat{\sigma}(\mathbf{X}; \mathcal{D}_{\text{tr}}^{\text{res}}), \hat{f}(\mathbf{X}; \mathcal{D}_{\text{tr}}) + q \times \hat{\sigma}(\mathbf{X}; \mathcal{D}_{\text{tr}}^{\text{res}})]$$

**C. Majority Vote.** We describe the Majority Vote procedure from (63). We use the same notation as established in Section 3.

**Step 1: Conformal Procedures Training and Calibration** For algorithms  $f_1, \dots, f_M$ , follow Procedure A to train and calibrate Split Conformal procedures  $\hat{C}_1, \dots, \hat{C}_M$  using  $\mathcal{D}$  at level  $1 - \alpha/2$ . For a point  $\mathbf{X}$ ,  $\hat{C}_i(\mathbf{X})$  would output the prediction interval using the Split Conformal procedure with algorithm  $f_i$ .

**Step 2: Generate Majority Vote Prediction Interval** For a new test point  $\mathbf{X}$ , generate prediction intervals  $\hat{C}_1(\mathbf{X}), \dots, \hat{C}_M(\mathbf{X})$ . Then produce the merged prediction interval

$$\mathcal{I} = \left\{ y : \frac{1}{M} \sum_{i=1}^M \mathbf{1}\{y \in \hat{C}_i(\mathbf{X})\} > \frac{1}{2} \right\}$$

**D. PCS procedure from Chapter 13 of (20).** We describe the PCS procedure from Chapter 13 of (20) for generating prediction intervals in the regression setting. Henceforth, we refer to this procedure as PCS (Ch 13). We use the same notation as established in Section 3.

**Step 1: Data-Splitting and Prediction-Check** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Train each algorithm on the training set to obtain fitted models  $\hat{f}_1(\cdot; \mathcal{D}_{\text{tr}}), \dots, \hat{f}_M(\cdot; \mathcal{D}_{\text{tr}})$ . Choose the top- $k$  performing algorithms according to loss  $l$ . Without loss of generality, let  $f_1 \dots f_k$  denote the top- $k$  performing algorithms.

**Step 2: Bootstrapping** Bootstrap the *training* set  $B$  times to obtain bootstrapped samples  $\mathcal{D}_{\text{tr}}^{(1)} \dots \mathcal{D}_{\text{tr}}^{(B)}$ . Fit all algorithms chosen in the previous step on every bootstrapped dataset  $\mathcal{D}_{\text{tr}}^{(b)}$  to obtain bootstrapped models  $\{\hat{f}_j(\cdot; \mathcal{D}_{\text{tr}}^{(b)}), j \in [k], b \in [B]\}$ . For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , we form a prediction set  $\mathcal{P}_i = \{\hat{f}_j(\mathbf{X}_i; \mathcal{D}_{\text{tr}}^{(b)}); j \in [k], b \in [B]\}$ .

**Step 3: Calibration** First, for each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , we form an uncalibrated interval  $[q_{\alpha/2}(\mathcal{P}_i), q_{1-\alpha/2}(\mathcal{P}_i)]$ , where  $q_\beta(S)$  is the  $\beta$  quantile for a set  $S$ . For a multiplicative scaling factor  $\gamma$ , generate a scaled interval

$$\mathcal{I}_i = \left[ q_{0.5}(\mathcal{P}_i) - \gamma \times (q_{0.5}(\mathcal{P}_i) - q_{\alpha/2}(\mathcal{P}_i)), \quad q_{0.5}(\mathcal{P}_i) + \gamma \times (q_{1-\alpha/2}(\mathcal{P}_i) - q_{0.5}(\mathcal{P}_i)) \right]$$

We choose the scaling factor  $\gamma$  such that we achieve  $1 - \alpha$  coverage on the data  $\mathcal{D}_{\text{val}}$ .

**Step 4: Generating PCS Prediction Interval** For a new test point  $\mathbf{X}$  let  $\mathcal{P} = \{\hat{f}_j(\mathbf{X}; \mathcal{D}_{\text{tr}}^{(b)}); j \in [k], b \in [B]\}$ . Then, we produce prediction interval

$$\mathcal{I} = \left[ q_{0.5}(\mathcal{P}) - \gamma \times (q_{0.5}(\mathcal{P}) - q_{\alpha/2}(\mathcal{P})), \quad q_{0.5}(\mathcal{P}) + \gamma \times (q_{1-\alpha/2}(\mathcal{P}) - q_{0.5}(\mathcal{P})) \right]$$

## S2. Additional Regression Results

In this section, we provide additional results for our regression experiments.

**A. Coverage.** We report coverage for the best-performing (as measured by average width) across our 17-real world datasets. All methods achieve desired coverage. Interestingly, Majority Vote (63) often over-covers, possibly explaining its larger width.

Method	PCS	Split Conformal (XGBoost)	Studentized Conformal (Random Forest)	Majority Vote
Naval	0.906	0.902	0.905	0.981
Energy	0.919	0.901	0.923	0.973
Computer	0.899	0.902	0.903	0.973
Miami	0.904	0.905	0.902	0.962
Diamond	0.902	0.898	0.899	0.953
Insurance	0.907	0.901	0.901	0.963
Elevator	0.901	0.896	0.898	0.965
Airfoil	0.922	0.914	0.915	0.972
Powerplant	0.904	0.901	0.903	0.963
Concrete	0.915	0.907	0.901	0.966
Debutanizer	0.907	0.892	0.902	0.951
Superconductor	0.902	0.900	0.897	0.967
Parkinsons	0.904	0.900	0.901	0.956
CA	0.903	0.905	0.898	0.958
Kin8nm	0.905	0.902	0.899	0.954
QSAR	0.901	0.900	0.898	0.965
Protein	0.902	0.900	0.896	0.962

Table S1. Coverage for PCS, and best-performing conformal methods across our 17 real-world datasets. All methods achieve desired coverage.

**B. Additional Subgroup Results.** We provide subgroup results for additional datasets.

**Subgroup Construction Procedure** We manually construct subgroups for each dataset. First, we fit a Random Forest over the dataset and identify the feature with the highest importance. If the most important feature is binary or categorical, we form the subgroups by partitioning the data into each category. If the feature is numerical, we inspect natural breaks in its distribution and partition the data accordingly (e.g. splitting between modes if the distribution is multi-modal). For a feature whose distribution does not have natural breaks, we partition the data into quartiles based on that feature.

**Insurance (49)** Dataset consists of predicting insurance charges on customers. We form subgroups by partitioning the data into smokers and non-smokers. PCS adapts its width to maintain coverage across subgroups. Split Conformal (XGBoost) and Majority Vote achieve coverage but produce larger intervals. Studentized Conformal (Random Forest) fails to achieve coverage in the Smoker subgroup and produces larger intervals.

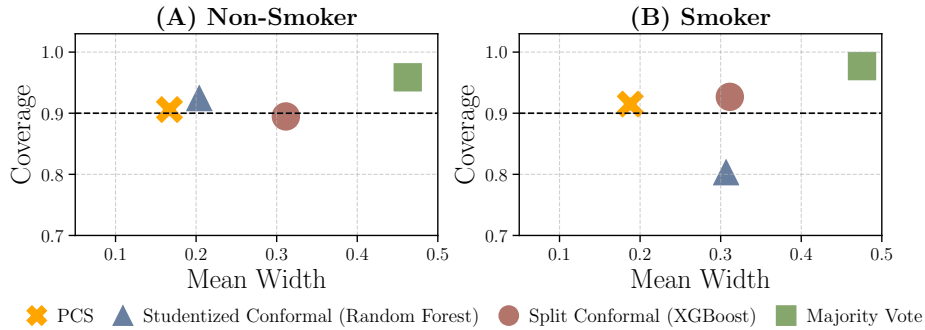
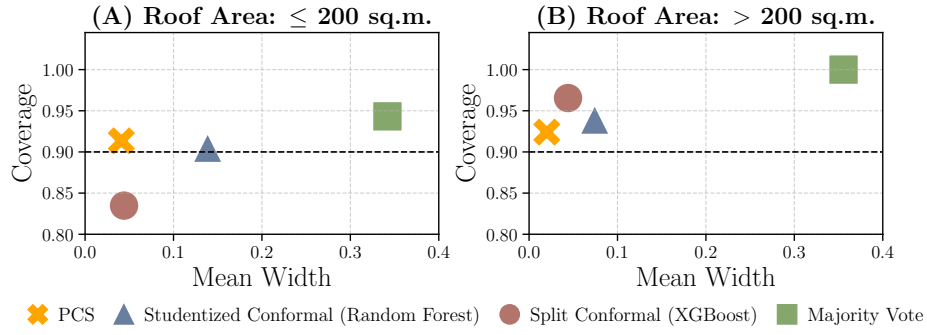


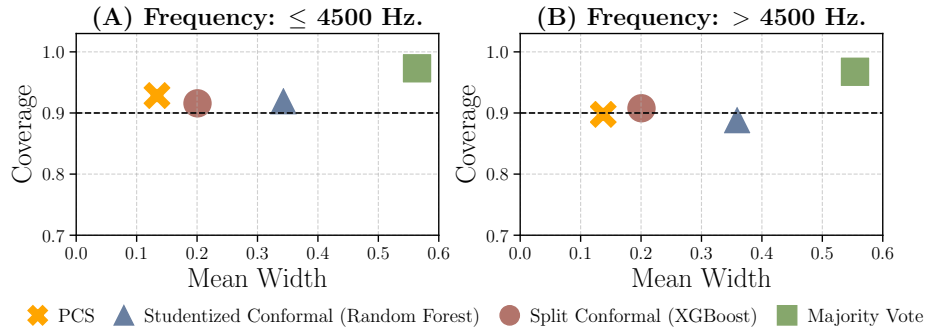
Fig. S1. Coverage and width for PCS, and conformal regression approaches on subgroups in the Insurance dataset (49). Panels (A) and (B) demonstrate performance for non-smokers and smokers respectively. PCS adapts its width to maintain coverage, while other methods fail to achieve desired coverage or produce larger intervals.

**Energy (47)** Dataset consists of predicting heating load requirements for buildings. We form subgroups based on roof area of the house. PCS adapts its width to maintain coverage across subgroups. Split Conformal (XGBoost) fails to achieve coverage in the subgroup with smaller roof areas. Studentized Conformal (Random Forest) and Majority Vote adapt widths to achieve valid coverage but produce larger intervals.



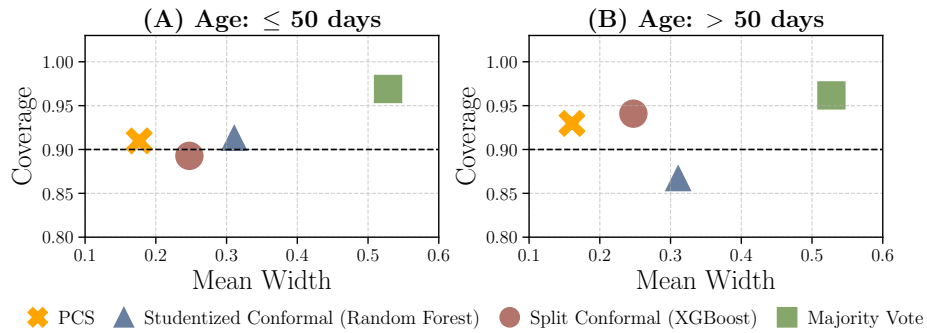
**Fig. S2.** Coverage and width for PCS, and conformal regression approaches on subgroups in the Energy Efficiency dataset (47). Panels (A) and (B) demonstrate performance on subgroups formed by roof area of the house. PCS adapts its width to maintain coverage, while other methods fail to achieve desired coverage or produce larger intervals.

**Airfoil (50)** Dataset consists of predicting sounds pressure level of airfoil blades. We form subgroups based on the frequency of the sound of the airfoil. PCS adapts its width to maintain coverage across subgroups and produces shorter intervals than all conformal methods. Studentized Conformal (Random Forest) slightly under-covers in the large-frequency subgroup, while the other conformal methods achieve desired coverage.



**Fig. S3.** Coverage and width for PCS, and conformal regression approaches on subgroups in the Airfoil dataset (50). Panels (A) and (B) demonstrate performance on subgroups formed by frequency of the sound of the airfoil. PCS adapts its width to maintain coverage, while other methods fail to achieve desired coverage or produce larger intervals.

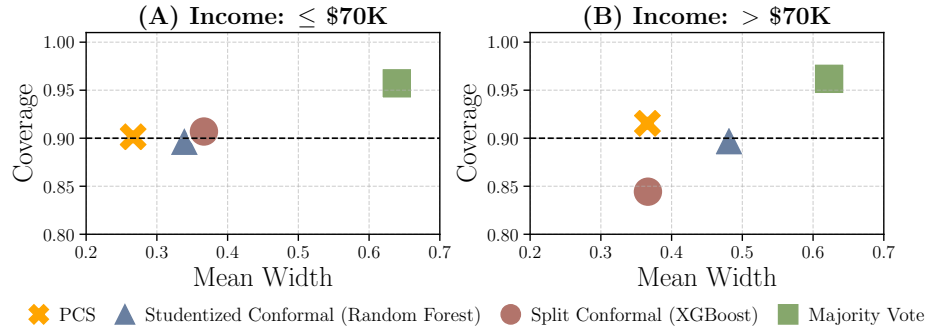
**Concrete (48)** Dataset consists of predicting concrete compressive strength. We form subgroups based on the age of the concrete block. PCS adapts its width to maintain coverage across subgroups. Split Conformal (XGBoost) and Majority Vote achieve coverage but produce larger intervals. Studentized Conformal (Random Forest) fails to achieve coverage in the large-age subgroup and produces larger intervals.



**Fig. S4.** Coverage and width for PCS, and conformal regression approaches on subgroups in the Concrete dataset (48). Panels (A) and (B) demonstrate performance on subgroups formed by age of the concrete block. PCS adapts its width to maintain coverage, while other methods fail to achieve desired coverage or produce larger intervals.

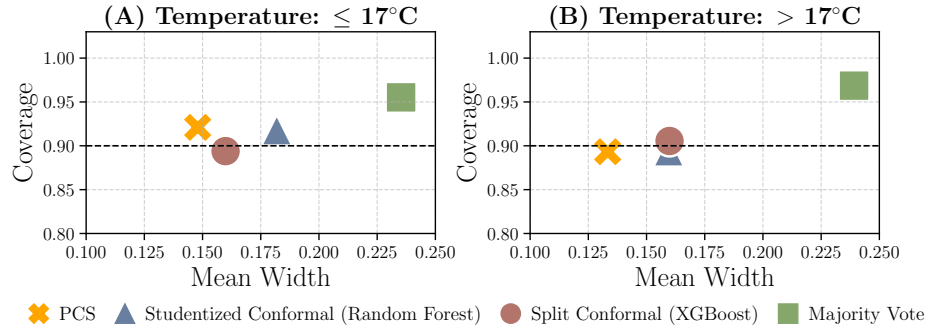
**California Housing (60)** Dataset consists of predicting median housing price in Census block groups. We form subgroups based on the median income of the block group. PCS adapts its width to maintain coverage across subgroups. Split Conformal (XGBoost) under-covers in the large-income subgroup. Other conformal methods maintain coverage while producing larger intervals.





**Fig. S5.** Coverage and width for PCS, and conformal regression approaches on subgroups in the CA Housing dataset (60). Panels (A) and (B) demonstrate performance on subgroups formed by median income of the block group. PCS adapts its width to maintain coverage, while other methods fail to achieve desired coverage or produce larger intervals.

**Powerplant (56)** Dataset consists of predicting electrical energy output of powerplants. We form subgroups based on the ambient temperature of the powerplant. PCS adapts its width to maintain coverage across subgroups. All conformal methods maintain coverage across subgroups while producing larger intervals.



**Fig. S6.** Coverage and width for PCS, and conformal regression approaches on subgroups in the Powerplant dataset (56). Panels (A) and (B) demonstrate performance on subgroups formed by ambient temperature of the powerplant. PCS adapts its width to maintain coverage, while other methods produce larger intervals.

**C. Comparison to PCS Ch.13 (20).** We report coverage and mean width of PCS and PCS Ch.13 across our 17 datasets. Both methods achieve desired coverage in all datasets. PCS produces equal or smaller interval width than PCS Ch.13.

Method	PCS		PCS Ch.13	
	Coverage	Mean Width	Coverage	Mean Width
Naval	0.906	0.001	0.906	0.001
Energy	0.919	<b>0.030</b>	0.918	0.035
Computer	0.899	<b>0.066</b>	0.898	0.067
Miami	0.904	<b>0.076</b>	0.901	0.080
Diamond	0.902	<b>0.093</b>	0.900	0.094
Insurance	0.907	<b>0.127</b>	0.912	0.179
Elevator	0.901	<b>0.127</b>	0.903	0.133
Airfoil	0.922	<b>0.134</b>	0.907	0.159
Powerplant	0.904	<b>0.139</b>	0.905	0.147
Concrete	0.915	<b>0.172</b>	0.917	0.209
Debutanizer	0.907	<b>0.187</b>	0.901	0.206
Superconductor	0.902	<b>0.193</b>	0.898	0.221
Parkinsons	0.904	<b>0.201</b>	0.898	0.206
CA	0.903	<b>0.271</b>	0.898	0.275
Kin8nm	0.905	<b>0.283</b>	0.899	0.286
QSAR	0.901	0.315	0.895	<b>0.311</b>
Protein	0.902	<b>0.571</b>	0.897	0.596

**Table S2.** Coverage and mean width of PCS and PCS Ch.13 across our 17 datasets. Both methods achieve desired coverage, while PCS produces equal or smaller interval width than PCS Ch.13.

### S3. Overview of Uncertainty Quantification Methods for Multi-Label Classification

**Additional Notations** Many conformal methods rely on ranks of predicted class probabilities, we introduce the following notation. For numbers  $a_1, a_2, \dots, a_n$ , let  $\pi$  be the permutation of the indices that sorts the numbers in descending order. That is,  $a_{\pi(1)} > a_{\pi(2)} > \dots > a_{\pi(n)}$ . We assume that  $\pi$  arbitrarily breaks ties. For classification algorithm  $\hat{f}(\cdot; \mathcal{D})$ , let  $\hat{f}^{(c)}(\cdot; \mathcal{D})$  denote the predicted probability for class  $c$ . Lastly, we adopt the convention that if a dataset consists of  $C$  classes, the classes are labeled  $1, 2, \dots, C$ .

**A. Top-K.** We describe the Top-K procedure from (41).

**Step 1: Data-Splitting and Model Training** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Fit algorithm  $f$  on training set to obtain fitted model  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$ .

**Step 2: Calibration** For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , make prediction using  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$  and obtain conformal score  $S_i = j$ , where  $Y_i = \pi(j)$  and  $\hat{f}^{(\pi(1))}(\mathbf{X}_i; \mathcal{D}_{\text{tr}}) > \hat{f}^{(\pi(2))}(\mathbf{X}_i; \mathcal{D}_{\text{tr}}) > \dots > \hat{f}^{(\pi(C))}(\mathbf{X}_i; \mathcal{D}_{\text{tr}})$ . That is,  $S_i$  is the rank of the predicted probability for the true class. Let  $q$  be the  $1 - \alpha$  quantile of the set  $\{S_i : i \in [|\mathcal{D}_{\text{val}}|]\}$ .

**Step 3: Generate Top-K Prediction Set** For a new test point  $\mathbf{X}$ , produce the prediction set

$$\mathcal{S} = \{\pi(1), \pi(2), \dots, \pi(q)\}$$

where  $\hat{f}^{(\pi(1))}(\mathbf{X}; \mathcal{D}_{\text{tr}}) > \hat{f}^{(\pi(2))}(\mathbf{X}; \mathcal{D}_{\text{tr}}) > \dots > \hat{f}^{(\pi(C))}(\mathbf{X}; \mathcal{D}_{\text{tr}})$ .

**B. Adaptive Prediction Sets.** We describe the Adaptive Prediction Sets (APS) procedure from (44).

**Step 1: Data-Splitting and Model Training** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Fit algorithm  $f$  on training set to obtain fitted model  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$ .

**Step 2: Calibration** For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , predict using  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$  and obtain conformal score  $S_i = \sum_{j=1}^t \hat{f}^{(\pi(j))}(\mathbf{X}_i; \mathcal{D}_{\text{tr}})$ , where  $\pi(t) = Y_i$ . Let  $q$  be the  $1 - \alpha$  quantile of the set  $\{S_i : i \in [|\mathcal{D}_{\text{val}}|]\}$ .

**Step 3: Generate APS Prediction Set** For a new test point  $\mathbf{X}$ , produce the prediction set

$$\mathcal{S} = \{\pi(1), \pi(2), \dots, \pi(v)\}, \text{ where } v = \min \left\{ t : \sum_{j=1}^t \hat{f}^{(\pi(j))}(\mathbf{X}; \mathcal{D}_{\text{tr}}) \geq q \right\}$$

**C. Regularized Adaptive Prediction Sets.** We describe the Regularized Adaptive Prediction Sets (RAPS) procedure from (41).

**Step 1: Data-Splitting and Model Training** Randomly split  $\mathcal{D}$  into a training set  $\mathcal{D}_{\text{tr}}$ , a tuning set  $\mathcal{D}_{\text{tune}}$ , and validation set  $\mathcal{D}_{\text{val}}$ . Fit algorithm  $f$  on training set to obtain fitted model  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$ .

**Step 2: Hyperparameter Tuning** The procedure has 2 hyperparameters:  $t_{\text{reg}}$  and  $\lambda$ . To tune  $t_{\text{reg}}$ , we perform Step 2 of the Top-K procedure (A) on  $\mathcal{D}_{\text{tune}}$  and set  $t_{\text{reg}} = q$ . To tune  $\lambda$ , we perform a grid search with candidate values: using the tuned  $t_{\text{reg}}$  and a candidate  $\lambda$ , we proceed to Steps 3 and 4 using  $\mathcal{D}_{\text{tune}}$ . We pick the  $\lambda$  that produce the smallest average prediction set size.

**Step 3: Calibration** For each  $(\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{val}}$ , predict using  $\hat{f}(\cdot; \mathcal{D}_{\text{tr}})$  and obtain conformal score

$$S_i = \sum_{j=1}^t \hat{f}^{(\pi(j))}(\mathbf{X}_i; \mathcal{D}_{\text{tr}}) + \lambda(t - t_{\text{reg}})^+$$

where  $\pi(t) = Y_i$  and  $(\cdot)^+$  denotes the positive part. Let  $q$  be the  $1 - \alpha$  quantile of the set  $\{S_i : i \in [|\mathcal{D}_{\text{val}}|]\}$ .

**Step 4: Generate RAPS Prediction Set** For a new test point  $\mathbf{X}$ , produce the prediction set

$$\mathcal{S} = \{\pi(1), \pi(2), \dots, \pi(v)\}, \text{ where } v = \min \left\{ k : \sum_{j=1}^k \hat{f}^{(\pi(j))}(\mathbf{X}; \mathcal{D}_{\text{tr}}) + \lambda(k - t_{\text{reg}})^+ \geq q \right\}$$

**D. Majority Vote.** We describe the Majority Vote procedure from (63).

**Step 1: Conformal Procedures Training and Calibration** For classification algorithms  $f_1, \dots, f_M$ , follow Procedure B to train and calibrate APS procedures  $\hat{C}_1, \dots, \hat{C}_M$  using  $\mathcal{D}$  at level  $1 - \alpha/2$ . For a point  $\mathbf{X}$ ,  $\hat{C}_i(\mathbf{X})$  would output the prediction set using the APS procedure with algorithm  $f_i$ .

**Step 2: Generate Majority Vote Prediction Interval** For a new test point  $\mathbf{X}$ , generate prediction sets  $\hat{C}_1(\mathbf{X}), \dots, \hat{C}_M(\mathbf{X})$ . Then produce the merged prediction set

$$\mathcal{I} = \left\{ k \in [n_{\text{cl}}] : \frac{1}{M} \sum_{i=1}^M \mathbf{1}\{k \in \hat{C}_i(\mathbf{X})\} > \frac{1}{2} \right\}$$

## S4. Additional Classification Results

We report coverage for the best-performing (as measured by average width) across our 6 tabular classification datasets. All methods achieve desired coverage. Interestingly, Majority Vote (63) often over-covers, possibly explaining its larger width.

Method	RAPS	Top-r	APS	PCS	Majority Vote
Dataset					
Chess	0.900313	0.909462	0.899733	0.897343	0.977245
Cover Type	0.900250	0.901700	0.898150	0.896850	0.988800
Dionis	NaN	0.893400	0.892467	0.904083	0.982150
Isolet	0.935064	0.924231	0.980449	0.960531	0.995321
Language	0.909500	0.912500	0.901500	0.909000	0.980000
Yeast	0.901347	0.920202	0.896296	0.900673	0.967340

**Table S1. Coverage for PCS, and best-performing conformal methods for our multi-label classification experiments.**

**A. Description of Classification Datasets.** We describe the context of the datasets used in our classification experiments.

**Language (71)** The dataset contains quantitative measurements of bodies of literature in English. Example features include frequency of first-person usage and frequency of past-tense usage. The goal is to predict the genre of the text, out of 30 potential genres such as fiction, memoir, and poetry.

**Yeast (72)** The dataset contains measurements on amino acid sequences of yeast proteins. Example features include discriminant analysis output of amino acid sequences, and nuclear localization consensus patterns. The goal is to predict the type of the yeast protein, out of 10 potential types such as cytoskeletal, nuclear and mitochondrial.

**Isolet (73)** The dataset contains characteristics of voice in recordings that each contain a single English letter. Example features include spectral coefficients, contour features, and sonorant features. The goal is to predict the letter pronounced, out of the 26 English letters.

**Cover Type (74)** The dataset contains cartographic variables of regions in the Roosevelt National Forest. Each observation represents a 30 by 30 meter cell. Example features include elevation, slope, and distance to nearest road. The goal is to predict the cover-soil type, out of the 100 potential types such as spruce-sand and pine-clay.

**Chess (75)** The dataset contains positions of both kings and the white rook in chess endgames. Example feature includes the row and column of the three pieces. The goal is to predict the number of optimal moves till white wins the game; if the move is more than 16, the game ends in a draw. Hence the potential classes are  $0, \dots, 16$  and “draw”.

**Dionis (76)** The dataset is anonymized from a machine learning challenge. The dataset has 60 numerical features and 355 classes. No further context is available.

## S5. Deep-Learning

We provide further details on the deep-learning experiments described in Section 7.

### A. Description of Datasets.

**CIFAR-100 (79)** This dataset consists of 60000  $32 \times 32$  natural colour images in 100 classes, each containing 600 images per class. There are 50000 training images and 10000 test images.

**Imagenet-Small (84)** ImageNet-Small contains 100000 natural images of 200 classes (500 for each class) downsized to  $64 \times 64$  colored images. Each class has 500 training images, 50 validation images and 50 test images.

**Caltech-UCSD Birds (73)** This is an image dataset with photos of 200 mostly North American bird species.

## S6. Theoretical Results

We describe the modified PCS procedure for regression, and then establish our formal theoretical results as follows.

**A. Modified PCS Procedure.** We use the notation established in Section 3.

1. Split  $\mathcal{D}$  into  $\mathcal{D}_{\text{tr}}$ ,  $\mathcal{D}_{\text{val}}$ , and  $\mathcal{D}_{\text{cal}}$ , and conduct the prediction check as described in step 1 of Section 3 on  $\mathcal{D}_{\text{val}}$  to obtain screened algorithms  $f_1, \dots, f_k$

2. Bootstrap the *training* dataset  $B$  times to obtain bootstrapped samples  $\mathcal{D}_{\text{tr}}^{(1)} \dots \mathcal{D}_{\text{tr}}^{(B)}$ . Fit all algorithms chosen in the previous step on every bootstrapped dataset  $\mathcal{D}_{\text{tr}}^{(b)}$  to obtain bootstrapped models  $\{\hat{f}_j(\cdot; \mathcal{D}_{\text{tr}}^{(b)}), j \in [k], b \in [B]\}$ .

To define the calibration procedure, we introduce some necessary notation. For a given point  $\mathbf{X}$ , form a prediction set  $\mathcal{P} = \{\hat{f}_j(\mathbf{X}; \mathcal{D}_{\text{tr}}^{(b)}); j \in [k], b \in T_i\}$ . Further, create  $[q_{\alpha/2}(\mathcal{P}), q_{1-\alpha/2}(\mathcal{P})]$ , where  $q_{\beta}(S)$  is the  $\beta$  quantile for a set  $S$  and let  $R_{\alpha}(\mathbf{X})$  denote the range of this interval. Next, we define score function for  $\mathbf{X}$  as follows

$$S(\mathbf{X}, Y) = \min\{\gamma : Y \in [-\frac{\gamma R_{\alpha}(\mathbf{X})}{2} + (\frac{q_{\alpha/2}(\mathcal{P}) + q_{1-\alpha/2}(\mathcal{P})}{2}), \frac{\gamma R_{\alpha}(\mathbf{X})}{2} + (\frac{q_{\alpha/2}(\mathcal{P}) + q_{1-\alpha/2}(\mathcal{P})}{2})\}. \quad [3]$$

Moreover, let  $S_{\text{cal}} = \{S(\mathbf{X}_i, Y_i) : i \in D_{\text{cal}}\}$ . Let  $\hat{\gamma}_{\alpha} = \lceil (1 - \alpha)(|D_{\text{cal}}| + 1) \rceil$  smallest of  $S_{\text{cal}}$

3. For a test point  $\mathbf{X}_*$ , define the PCS prediction set as

$$\hat{C}_{\text{PCS}}(\mathbf{X}_*) = \left[ \frac{\hat{\gamma}_{\alpha} R_*}{2} + \left( \frac{l(\mathbf{X}_*) + u(\mathbf{X}_*)}{2} \right), \frac{\hat{\gamma}_{\alpha} R_*}{2} + \left( \frac{l(\mathbf{X}_*) + u(\mathbf{X}_*)}{2} \right) \right], \quad [4]$$

Next, we provide our formal theoretical result and its proof.

**Theorem 1.** *For a test point  $(\mathbf{X}_*, Y)$ , assume  $\mathcal{D} \cup (\mathbf{X}_*, Y)$  is exchangeable. For given  $\alpha \in (0, 1)$ , the PCS prediction interval Eq. (4) satisfies*

$$\mathbb{P}(Y \in \hat{C}_{\text{PCS}}(\mathbf{X}_*) \mid (\mathbf{X}_i, Y_i) \in \mathcal{D}_{\text{tr}} \cup \mathcal{D}_{\text{val}}) \geq 1 - \alpha$$

*Proof.* The proof follows from the fact that we can rewrite the modified PCS prediction set Eq. (4) as follows

$$\hat{C}_{\text{PCS}}(\mathbf{X}_*) = \{y : S(\mathbf{X}_*, y) \leq \hat{\gamma}_{\alpha}\}.$$

Given this observation, we use the result that (9) that any a prediction set with a prefitted score function defined with form above is guaranteed to have  $1 - \alpha$  coverage.  $\square$