Conformalized Semi-supervised Random Forest for Classification and Abnormality Detection

Yujin Han†
Department of Computer Science,
The University of Hong Kong
Hong Kong, China

Mingwenchan Xu†
Department of IEMS
Northwestern University
Illinois, USA

Leying Guan
Department of Biostatistics
Yale University
New Haven, USA

Abstract

The Random Forests classifier, a widely utilized off-the-shelf classification tool, assumes training and test samples come from the same distribution as other standard classifiers. However, in safety-critical scenarios like medical diagnosis and network attack detection, discrepancies between the training and test sets, including the potential presence of novel outlier samples not appearing during training, can pose significant challenges. To address this problem, we introduce the Conformalized Semi-Supervised Random Forest (CSForest), which couples the conformalization technique Jackknife+aB with semi-supervised tree ensembles to construct a set-valued prediction $C(x)$. Instead of optimizing over the training distribution, CSForest employs unlabeled test samples to enhance accuracy and flag unseen outliers by generating an empty set. Theoretically, we establish CSForest to cover true labels for previously observed inlier classes under arbitrarily label-shift in the test data. We compare CSForest with state-of-the-art methods using synthetic examples and various real-world datasets, under different types of distribution changes in the test domain. Our results highlight CSForest's effective prediction of inliers and its ability to detect outlier samples unique to the test data. In addition, CSForest shows persistently good performance as the sizes of the training and test sets vary. Codes of CSForest are available at https://github.com/yujinhan98/CSForest.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

.

1 INTRODUCTION

A classifier typically generates predictions for a test sample by choosing the class label associated with the highest predicted probability. This approach proves inadequate for addressing the increasing demand for assessing prediction reliability in practical scenarios, such as medical diagnosis (Esteva et al., 2017; Kompa et al., 2021) and autonomous vehicles (Kalra and Paddock, 2016; Qayyum et al., 2020). One approach for addressing this challenge involves minimizing a combined cost associated with misclassification and rejection, permitting the avoidance of predictions for test samples exhibiting high uncertainty. For example, if the maximum estimated probability $\max_{k\in\{0,1\}} \hat{p}(k|x)$ for the binary response is low where $\hat{p}(k|x)$ is the estimated probability of beging in class k using the training data, we might choose not to predict a test observation x (Chow, 1970; Herbei and Wegkamp, 2006; Bartlett and Wegkamp, 2008). This idea has been implemented across various learning algorithms and expanded to address multiclass classification problems (Cortes et al., 2016; Ni et al., 2019; Charoenphakdee et al., 2021). The set-valued prediction via conformal prediction provides an alternative framework (Vovk et al., 2005; Papadopoulos et al., 2002; Lei and Wasserman, 2015; Gammerman et al., 2013), in which the classifier generates a set covering all possible labels for a given observation x based on the conformal score function $s(x, k)$ that measures the plausibility of label for x being k, e.g., $s(x, k) \leftarrow \hat{p}(k|x)$. For instance, one can form the calibrated set-valued prediction set $C(x) = \{k : s(x, k) \geq \tau_k\}$, with τ_k being class-dependent and calculated to ensure a desired coverage of the true label (Vovk et al., 2005).

Traditionally, classification uncertainty quantification assumes that the training and test samples are indepen-

†Equal contribution. This work was done at Yale University. Correspondence to: leying.guan@yale.edu.

dently and identically distributed (i.i.d.) from the same distribution. In reality, this assumption doesn't always hold. For instance, in medical applications, the test cohort may include samples representing novel pathologies that bear little similarity to the labeled training set (Lin et al., 2005). Similarly, network attackers may create new intrusions to evade existing detection systems (Marchette and Marchette, 2001). Therefore, it becomes essential to assess uncertainty under distributional changes and flag test samples where predictions should not rely solely on the model trained with the training data.

To address this challenge, we introduce CSForest (**C**onformalized **S**emi-Supervised Random **Forest**), an ensemble tree classifier that leverages both labeled training data and unlabeled test data to form calibrated set-valued prediction and flag outliers. The term "semi-supervision" here refers to the utilization of unlabeled test data. CSForest builds upon recent work on test-data optimized calibrated classification framework(Guan and Tibshirani, 2022). Guan and Tibshirani (2022) constructs a calibrated semi-supervised set-valued prediction via sample-splitting where one subset of samples is used for training the model while the remaining part is for calibration. In contrast, CSForest avoids the sample-splitting schema, constructing the random forest tree ensembles and calibrating the prediction using all samples. We summarize our contributions into three main aspects:

1. We present a novel classifier, CSForest, designed for classification with calibrated uncertainty quantification in the presence of distributional shifts between training and test datasets. It employs a novel semi-supervised random forest structure that differentiates between observed training classes and unlabeled test data, and adapts the conformalization technique Jackknife+aB (Kim et al., 2020) to handle the case of joint and asymmetric utilization of both training and test samples.

2. We provide a theoretical guarantee for true lable coverage using $C(x)$ constructed by CSForest, under arbitrarily shifted test distributions. This theoretically ensures the effectiveness of CSForest under varying degrees of data drift.

3. We conduct extensive experiments on simulated and publicly available datasets under various label shift settings to demonstrate CSForest's gain over existing state-of-the-art methods.

2 RELATED WORK

Distribution Shift. Regarding distributional changes, both the covariate and label shifts are commonly studied (Schölkopf et al., 2012). The former assumes the conditional density $p(y|x)$ to be fixed with $f(x)$, the marginal density of x, potentially changing (Shimodaira, 2000; Bickel et al., 2009; Gretton et al., 2009; Csurka, 2017); the latter treats $f_k(x)$, conditional density of x given the label $y = k$, as fixed, but the prevalence of different labels can vary among the observed training classes (Storkey, 2009; Lipton et al., 2018).

Recently, Guan and Tibshirani (2022) proposes BCOPS, a test-data optimized calibrated classifier, and the Generalized Label Shift (GLS) model defined in eq. (1), which extends the label shift model to include unseen classes to handle outliers. Suppose that the training data is a mixture of K different classes. For class k, its mixture proportion is π_k, and feature density is $f_k(x)$, with π_k satisfying $\sum_{k=1}^K \pi_k = 1$. The generalized label shift model assumes a target distribution accepting both label shift among training classes and the appearance of outlier component(s) and requires only $f_k(x)$ to remain the same for each observed class:

$$\mu(x) = \sum_{k=1}^{K} \tilde{\pi}_k f_k(x) + \delta \cdot f_R(x), \qquad (1)$$

where $\delta + \sum_{k=1}^K \tilde{\pi}_k = 1$, $\tilde{\pi}_k \geq 0$ represents the proportion of samples from class k in the target distribution, $\delta \geq 0$ represents the proportion of outlier samples not from the observed classes, and $f_R(x)$ represents the density for the outlier component. Under the GLS model, BCOPS utilizes both labeled training samples and unlabeled test samples to construct calibrated set-valued prediction. The crucial calibration step of BCOPS relies on sample-splitting, which results in low data-utilization efficiency, especially when training or test samples are limited.

Conformal Prediction. Conformal prediction (also known as conformal inference) (Vovk et al., 2005; Papadopoulos et al., 2002; Lei and Wasserman, 2015) aims to create statistically rigorous uncertainty sets/intervals for the predictions from classical machine learning models, aiming to cover the true label with a desired probability in the non-asymptotic regime, without model assumptions on how y depends on x. Consider $(x_i, y_i)_{i=1}^n$ as n (feature, label) pairs, and a new sample (x_{n+1}, y_{n+1}) where y_{n+1} is unobserved. Based on the previous n observations, the conformal prediction creates a prediction set $\hat{C}_n(x_{n+1})$ for the new instance x_{n+1} and ensure that $\mathbb{P}(y_{n+1} \in \hat{C}_n(x_{n+1})) \geq 1 - \alpha$, where $\alpha \in (0, 1)$ is the allowed miscoverage level. For example, in the classification setting, if $\alpha = 0.1$, then the probability that $\hat{C}_n(x_{n+1})$ contains the true label y_{n+1} is no smaller than 90%.

A key step in forming the conformal prediction is the choice of the conformal score function $s(x, y)$, which is used for evaluating how plausible of observing certain (x_i, y_i), followed by a calibration of the probability for observing (x_{n+1}, y_{n+1}) via comparing its score to those from labeled training samples. As some examples for classification problems, Romano et al. (2020) considers $s(x, k)$ as the probability (estimated) of observing labels with estimated conditional probability $p(.|x)$ no worse than that of class k, and $\hat{C}_n(x_{n+1}) = \{k : s(x_{n+1}, k) \geq \tau\}$ where τ is a threshold determined by the empirical distribution $\{s(x_i, y_i)\}_{i=1}^n$. Vovk et al. (2005) and Lei (2014) consider the conformal score $s(x, k)$ as the conditional probability of having label k or density of x in class k, and construct $\hat{C}_n(x_{n+1}) = \{k : s(x_{n+1}, k) \geq \tau_k\}$ using a class-specific threshold τ_k where τ_k is determined by the empirical distribution $\{s(x_i, k)\}_{i:y_i=k}$.

Different conformal prediction schemes have been developed in the literature. During the early times, split conformal prediction is usually adopted where we train the conformal score function $s(.)$ using one-fold of the data and perform calibration τ using the remaining (Vovk et al., 2005). In recent years, significant progress has been made in cross-conformal prediction to improve data utilization efficiency. Vovk et al. (2018) proposed splitting data into multiple folds, calculating scores for each fold using score functions learned from the remaining data, and aggregating all scores for calibration. Barber et al. (2021) developed Jacknife+ for regression problems, which combines Jacknife with conformal prediction and constructs the prediction interval as

$$\hat{C}(x) = \left\{ y : \frac{1}{n+1}\left(1 + \sum_{i=1}^{n} \mathbb{1}_{\hat{s}^i(x,y) \geq \hat{s}^i(x_i,y_i)}\right) \geq \alpha \right\},$$

where $\hat{s}^i(x, y) = |\hat{m}^i(x) - y|$ is the conformal score function using the mean-prediction function $\hat{m}^i(x)$ learned from training samples excluding (x_i, y_i). Although Jacknife+ can only provide a worst-case coverage guarantee at level $(1 - 2\alpha)$, the achieved empirical coverage is often well-calibrated. Kim et al. (2020) described Jacknife+aB to mediate the computational burden of Jacknife+, which ensembles and calibrates prediction using repeated Bootstraps rather than retraining the model after excluding each training sample.

3 CONFORMALIZED SEMI-SUPERVISED RANDOM FOREST

Despite the popularity of random forest and its variants, existing work implicitly assumes that training and test samples originate from the same distribution. This reliance makes them unreliable in the presence of distributional changes, which can be particularly problematic in safe-critical applications. Classification uncertainty quantification in this setting is also challenging. To address this issue, we introduce CSForest (**C**onformalized **S**emi-Supervised Random **Forest**), a tree-ensemble classifier that produces set-valued predictions designed to incorporate true labels while minimizing the inclusion of false labels, customized to match a target distribution $\mu(x)$.

$$\min \int_x |C(x)|\mu(x)dx,$$

s.t. P[k$\in C(X)|Y = k] \geq 1-\alpha$, for all $k = 1, \ldots, K$.(2)

Specifically, CSForest optimizes for a target distribution as a mixture of the training density $f_{tr}(x)$ and test feature density $f_{te}(x)$ and set $\mu(x) = f_{te}(x) + wf_{tr}(x)$, where $w \geq 0$. If $w = 0$, $\mu(x) = f_{te}(x)$ and the objective of CSForest coincides with the objective of BCOPS, which optimizes for the expected test cohort classification accuracy. On the other hand, when w is large, it has a similar objective as the CRF model and optimizes classification performance on samples generated the same way as the training cohort. When w is not excessively large, $f_{te}(x)$ is a significant contributor to $\mu(x)$, the constrained optimization objective in eq. (3) encourages $C(x) = \emptyset$ for unseen outliers even though we do not explicitly model outliers. In other words, if x is unlikely to belong to any of the observed training classes, we prefer to classify it as an outlier.

The following Proposition 3.1 further provides the oracle solution to eq. (3) under GLS model:

Proposition 3.1. *Set the conformal score function as $s(x, k; \mu) = [f_k(x)/\mu(x)]$. Under the GLS model, the solution to eq. (3) is $C(x) = \{k : \mathbb{E}_X[\mathbb{1}\{s(x, k; \mu) \geq s(X, k; \mu)\}|Y = k] \geq \alpha, k = 1, \ldots, K\}$.*

Proposition 3.1 rephrases Proposition 2 from Guan and Tibshirani (2022), and its proof is provided in Appendix A for completeness. CSForest estimates $C(x)$ in Proposition 3.1 using a semi-supervised random forest that utilizes labeled training samples and the unlabeled test cohort, coupled with the Jackknife+aB strategy for correct calibration to ensure coverage guarantee. Specifically, for a test sample x_i and a training sample $x_{i'}$ from class k with size n_k, CSForest constructs a conformal score function $\hat{s}^{ii'}(x, k; \mu)$ trained without x_i and $(x_{i'}, y_{i'})$ to measure how likely a sample is from class k. More specifically, CSForest replaces the oracle score $s(x, k; \mu)$ in Proposition 3.1 when comparing x_i and $x_{i'}$ and replaces expectation $\mathbb{E}_X[.]$ is replaced by an empirical version using corresponding $\hat{s}^{ii'}(x, k; \mu)$. In other words, let n_k be the training sample size for class k, the inclusion criterion in Proposition 3.1 is replaced by its empirical version below:

$$\hat{s}_{ik} = \frac{1 + \sum_{y_i=k} \mathbb{1}\{\hat{s}^{ii'}(x_i, k; \mu) \geq \hat{s}^{ii'}(x_{i'}, k; \mu)\}}{n_k + 1}. \quad (3)$$

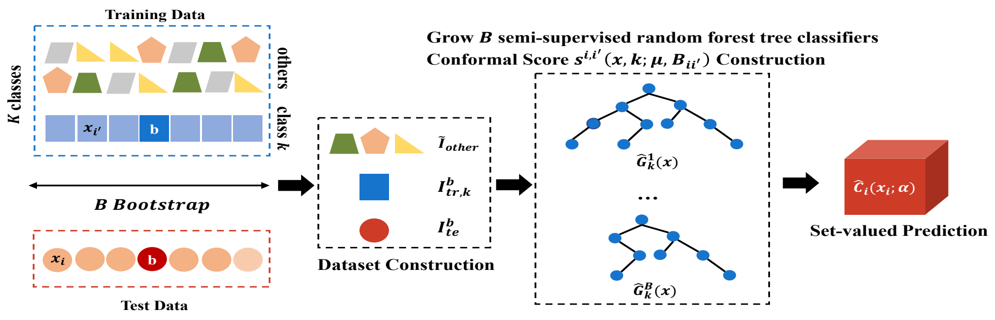


Figure 1: Overview of CSForest. For class k, let $\mathcal{I}_k^b$, $\mathcal{I}_{te}^b$ and $\tilde{\mathcal{I}}_{other}$ be Bootstrap samples from from training class k, test samples and training samples other than class k. We train a multi-class tree classifier with random feature selection as in the random forest using the Bootstrapped samples, where we maintain the labels all training samples and treat the test set as its own class. The resulting B random forest tree classifiers, $\{\hat{G}^1(x), ..., \hat{G}^B(x)\}$, are used to separate different labeled classes and the test samples. For the sample pair $x_i \in \mathcal{I}_{te}$ and $x_i' \in \mathcal{I}_k$, we aggregate trees that do not use x_i and x_i' (i.e., the data $\mathcal{B}_{ii'} = \{b : i \notin \mathcal{I}_{te}^b, i' \notin \mathcal{I}_k^b\}$) to form an ensemble classifier, and subsequently, an ensemble conformal score function $\hat{s}^{ii'}(x, k; \mu)$. Finally, we use the score function $\hat{s}^{ii'}(x, k; \mu)$ and compare $\hat{s}^{ii'}(x_i, k; \mu)$ to $\hat{s}^{ii'}(x_{i'}, k; \mu)$ for all $i' \in \mathcal{I}_k$ to form the calibrated evaluation $\hat{s}_{ik}$ for test sample x_i being in class k and include k in the prediction set $\hat{C}(x_i)$ if $\hat{s}_{ik}$ is no smaller than α.

The the estimated prediction set $\hat{C}(x_i)$ is :

$$\hat{C}(x_i) = \{k : \hat{s}_{ik} \geq \alpha\}. \tag{4}$$

Given a user-specified weight w, Figure 2 presents a graphical illustration of its model structure and Algorithm 1 delineates the ensemble tree constructions and prediction calibrations for CSForest. In Algorithm 1, we use $\mathcal{I}_{tr}$ and $\mathcal{I}_{te}$ to denote the training and test sets, respectively, and $\mathcal{I}_k$ to denote samples from the training class k. For each class k, lines 2-5 construct B Bootstrapped random forest tree classifiers to separate the training classes and the test samples. The random forest tree refers to a tree whose split is selected by the best split from L randomly selected candidate features, as constructed in the random forest, with $L = \lfloor\sqrt{p}\rfloor$ as the default split number in the R *range* package. Line 6 constructs the conformal score function $\hat{s}^{ii'}(x, k; \mu)$, using only the trees excluding test sample x_i and the training sample $x_{i'}$. In short, Algorithm 1 can be seen as estimating the oracle conformal score $s(x, k; \mu)$ under the target distribution with w being not excessively large, by utilizing trees from a weighted random forest classifier.

The estimated conformal score functions are then used to construct the calibrated score, $\hat{s}_{ik}$, and the prediction set, $\hat{C}(x)$, in lines 8-13. It is worth noting that, in line 3, to prevent redundant resampling in Bootstrap, we constrain $\tilde{\mathcal{I}}_{other}$ to be the Bootstrap sample of size $\min(\lceil n_{te}w\rceil, n - n_k)$ drawn from training samples, excluding class k. It is worth noting that the probability $Pr(B_{ii'} = \emptyset)$ decreases rapidly as B increases.

Algorithm 1 CSForest

```
Input  : Training Data $\{(x_i, y_i), i \in \mathcal{I}_{tr}\}$, Test Data
         $\{x_i, i \in \mathcal{I}_{te}\}$, $\tilde{B}$ and $w$ (1 by default.)
Output : Prediction sets $\hat{C}_i(x_i)$ for $i \in \mathcal{I}_{te}$.
1  for $k = 1, \ldots, K$ do
2      Sample $B$ from Binomial($\tilde{B}; (1 - \frac{1}{n_k+1})^{n_k}$).
       for $b = 1, \ldots, B$ do
3          Let $\mathcal{I}_k^b$, $\mathcal{I}_{te}^b$ be the Bootstraps of $\mathcal{I}_k$ (index of
            training class $k$) and $\mathcal{I}_{te}$. Let $\tilde{\mathcal{I}}_{other}$ be the
            Bootstrap of size $\min(\lceil n_{te}w\rceil, n - n_k)$ from
            the remaining training sample indices $\mathcal{I} \setminus \mathcal{I}_k$.

4          Grow a single random forest tree classifier $\hat{G}^b(x)$
            separating different labeled classes and the test
            samples using $\mathcal{I}_k^b \cup \mathcal{I}_{te}^b \cup \tilde{\mathcal{I}}_{other}$.
5      end
6      For sample pair $i \in \mathcal{I}_{te}, i' \in \mathcal{I}_k$, set $\mathcal{B}_{ii'} = \{b : i \notin$
        $\mathcal{I}_{te}^b, i' \notin \mathcal{I}_k^b\}$ and construct the conformal score
        function $\hat{s}^{ii'}(x, k; \mu) = \left(\sum_{b \in \mathcal{B}_{ii'}} \hat{G}_k^b(x)\right) / |\mathcal{B}_{ii'}|$.
7  end
8  for $i \in \mathcal{I}_{te}$ do
9      for $k = 1, \ldots, K$ do
10         Construct $\hat{s}_{ik}$ for sample $i$ and class $k$ via eq. (3).
11     end
12     Construct $\hat{C}(x_i) = \{k : \hat{s}_{ik} \geq \alpha\}$.
13 end
```

Remark 3.2. When $B_{ii'}$ is empty, $s^{ii'}(x, k; \mu)$ is not defined. In this case, We will exclude training sample $x_{i'} \in \mathcal{I}_k$ when calibrating for test sample x_i. Let n_{te} be the size of $\mathcal{I}_{te}$. The probability that $B_{ii'} = \emptyset$ is bounded by $\mathbb{P}[i' \in \mathcal{I}_k^b, \forall b \leq B] + \mathbb{P}[i \in \mathcal{I}_{te}^b, \forall b \leq B] =$

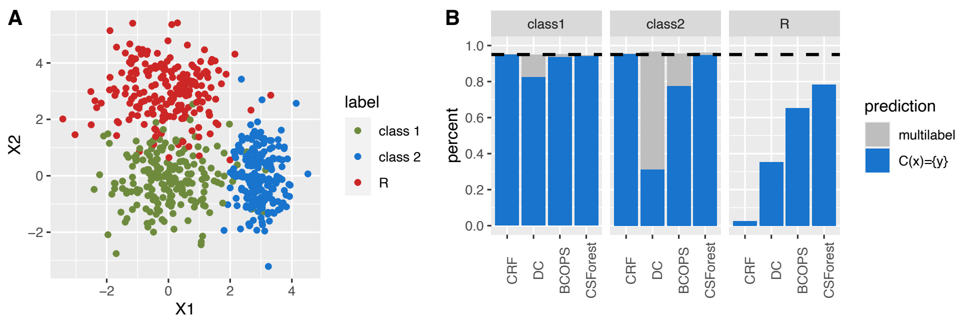


Figure 2: Panel A shows the first two dimensions of samples are generated from the three classes: green/blue/red points representing samples from class 1/2/R. Panel B shows the coverage rate which is defined by the proportion of samples with true labels included in their prediction sets. The horizontal dash line refers to the coverage level of 95%. Panel B is grouped by the actual labels in the testing data and colored based on if a prediction set contains only the correct label (blue) or more than the correct label (gray).

$[1-(1-\frac{1}{n_k})^{n_k}]^B + [1-(1-\frac{1}{n_{te}})^{n_{te}}]^B \approx 2(1-\frac{1}{e})^B$ for decently large n_{te} and n_k, which decreases fast with B.

Theorem 3.3 states that CSForest provides a worst-case coverage guarantee for the true response at the level $(1-2\alpha)$.

Theorem 3.3. *Suppose the generalized label shift model holds where features from class k are i.i.d generated from a distribution $\mathcal{P}_k$. For any fixed integers $\tilde{B} \geq 1$, the constructed $\hat{C}_i(x)$ from CSForest satisfies:*

$$\mathbb{P}\left[k \in \hat{C}(x_i) | y_i = k\right] \geq 1-2\alpha, \tag{5}$$

$$\textit{for all } i \in \mathcal{I}_{te} \ \textit{ and } \ k = 1, \ldots, K.$$

While the proof relies on the previous arguments used in Jacknife+aB for supervised regression problem Kim et al. (2020). However, new conditioning arguments are needed to estabilish exchangeability due to the paired sampling of both training and test samples. Please find the proof of Theorem 3.3 in Appendix A.

Theorem 3.3 ensures per-class coverage for observed training classes, which means we guarantee true label coverage for inlier classes even in arbitrarily shifted test distributions, e.g.,

$$\mathbb{P}\left[y \in C(x) | y \in \{1, \ldots, K\}, (x,y) \sim P_{te}\right] \geq 1-2\alpha,$$

for any test distribution P_{te} satisfying the generalized label shift model assumption. Although the theoretical guarantee for the worst-case coverage is at the level $(1-2\alpha)$, the empirical coverage using CSForest is usually close to or above the targeted level $(1-\alpha)$.

4 EXPERIMENTS

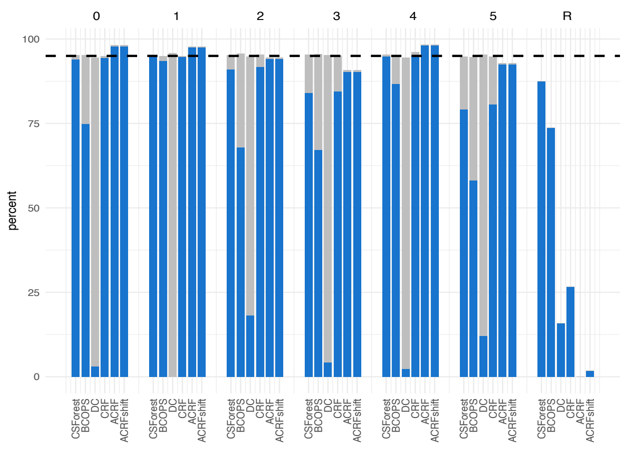
4.1 Synthetic Data

We begin with a simple illustrative 2D synthetic dataset and compare the performance using CSForest and three closely related set-valued conformal classifiers: BCOPS, which is a conformalized semi-supervised classifier that uses half of the samples for training while the other half for calibration (see Section 2); DC (density set classifier)(Cadre, 2006; Lei, 2014; Hechtlinger et al., 2018; Sadinle et al., 2019) and CRF (conformalized random forest), which follow the sample-splitting conformal prediction scheme while using the kernel estimate of the per-class density for $x|y$ and the estimated conditional probability of observing a label $y|x$ via a random forest on the training samples as the conformal score functions, respectively.

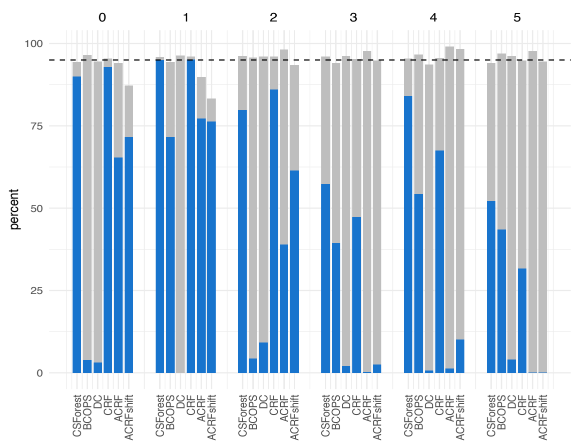
Example 1. *Let $X \in \mathbb{R}^{10}$ be the feature. We observe two classes $Y \in \{1,2\}$ in the training data, but the test data contains outliers labeled with $Y = R$. We generate $X_j \sim N(0,1)$ $(j = 3, \ldots, 10)$ as noise, with different classes separated by the first two dimensions:*

$$\begin{cases} X_1 \sim N(0,1),\ X_2 \sim N(0,1), & Y=1, \\ X_1 \sim N(3,0.5),\ X_2 \sim N(0,1), & Y=2, \\ X_1 \sim N(0,1),\ X_2 \sim N(3,1), & Y=R. \end{cases}$$

Figure 2(A) shows the first two dimensions of samples generated from the three classes $Y \in \{1,2,R\}$. We generated 200 samples from classes 1 and 2 to form the training set and 200 samples from each of the three classes to form the test set. In Figure 2(B), we evaluated the quality of the set-valued prediction $\hat{C}(x)$ using DC, CRF, BCOPS, and CSForest across 20 independent runs with a targeted miscoverage rate at $\alpha = 0.05$. All four methods achieve the desired 95% $(1-\alpha)$ coverage on true labels. However, both CSForest and BCOPS adapted to the test cohort and outperformed CRF and DC significantly in outlier detection. Additionally, compared to BCOPS, CSForest had fewer samples with multiple labels from classes 1 and 2 and exhibited a higher rejection rate for outliers.



(A) Per-class quality evaluation with outliers but no additional label shift among inlier digits, where the outliers are defined as $R = \{6, 7, 8, 9\}$.



(B) Per-class (class 0-5) quality evaluation with additional label shift among inlier digits but no outliers.

Figure 3: Per-class quality evaluation on MNIST. Panel A and B were grouped by the true labels in the testing data and colored based on whether a prediction set contains only the correct label (blue) or more than the correct label (gray). The horizontal dash line refers to the coverage level of 95%.

4.2 Real-World Data

In this section, our primary objective is to evaluate the effectiveness of CSForest on various real-world datasets, focusing on addressing the following three questions:

Q1. Can CSForest detect outliers efficiently while making accurate predictions for inliers in the presence of outliers but no additional label shift among inliers? (denoted as **outliers w/o shift**.)

Q2. Under the traditional label shift model without outliers, can CSForest achieve competitive performance compared to alternative classifiers? (denoted as **shift w/o outliers**.)

Q3. Does CSForest demonstrate stable performance as the training and test sample sizes vary?

Q1 and Q2 capture two extreme settings of the GLS model. We have set w at its default value for all experiments in the main paper with $w = 1$. In Appendix E, we include the sensitivity analysis of the crucial parameter w and shows CSForest performs well for different values of w, with $w = 1$ being a reasonable choice to balance performance for both inliers and outliers.

Datasets and Baselines. Our evaluation is conducted on three well-established image benchmarks: MNIST (LeCun and Cortes, 2010), FashionMNIST (Xiao et al., 2017), and CIFAR-10 (Krizhevsky et al., 2009). Additionally, we have included tabular data from the Network Intrusion domain and Chest X-ray data from the medical domain in the Appendix D.1 to demonstrate the effectiveness of CSForest in handling diverse datasets. To evaluate CSForest's performance, we compared it with BCOPS, CRF, DC along with two other approaches based on adaptive classification (Romano et al., 2020) and the covariate shift conformal prediction Tibshirani et al. (2019): ACRF (Adaptive classifier via random forest) and ACRFshift (Adaptive classifier via random forest under covariate shift). ACRF is a derandomized version of the existing conformalized adaptive random forest classifier (Romano et al., 2020), denoted as ACRFrandom, which aims for adaptive coverage across different feature regions. We will show results using ACRF instead of ACRFrandom in the main paper due to the latter's tendency to produce overly wide prediction sets, due to the attempt to achieve conditional coverage as indicated in the original paper (Romano et al., 2020). ACRFshift combines ACRF with the covariate shift conformal prediction. BCOPS, ACRF and ACRFshift all utilize random forest for constructing set-valued predictions, as a fair comparison to CSForest. More details of these baselines are provided in Appendix B.

Training Details and Evaluations. We set the number of trees $B = 3000$ for CSForest, and repeat all experiments ten times for performance evaluations. We evaluate the effectiveness of all methods using the type I error, type II error, and the average set length of $\hat{C}(x)$ at $\alpha = 0.05$. Type I error is the percentage of samples with the true label excluded from their associated set-valued prediction $\hat{C}(x)$ for observed classes. This error measure is directly linked to the coverage guarantee in Theorem 3.3. Type II error is calculated as the

Table 1: Achieved type I and type II errors at $\alpha = 0.05$ under different distributional shift settings. While most methods achieved desirable type I errors and true label coverage rates $(1-\alpha)$, only CSForest consistently achieved lower type II errors in both settings.

Dataset	Method	outliers w/o shift		shift w/o outliers	
		Type I Error	Type II Error	Type I Error	Type II Error
MNIST	CSForest	0.049±0.006	0.091 ± 0.008	0.048±0.016	**0.291±0.038**
	BCOPS	0.048±0.004	0.237±0.019	0.042±0.007	0.556±0.040
	DC	0.049±0.008	0.890±0.021	0.046±0.016	0.968±0.022
	CRF	0.048±0.007	0.338±0.035	0.046±0.018	0.428±0.082
	ACRF	0.046±0.006	0.430±0.003	0.025±0.011	0.884±0.012
	ACRFshift	0.046±0.006	0.432±0.009	0.055±0.013	0.828±0.015
CIFAR-10	CSForest	0.051±0.008	**0.000±0.000**	0.049±0.013	0.009±0.035
	BCOPS	0.049±0.006	0.001±0.000	0.042±0.009	0.029±0.006
	DC	0.046±0.007	0.048±0.091	0.039±0.010	0.071±0.115
	CRF	0.049±0.008	0.003±0.000	0.047±0.015	**0.000±0.000**
	ACRF	0.003±0.001	0.402±0.001	0.040±0.009	0.221±0.023
	ACRFshift	0.003±0.001	0.069±0.003	0.046±0.007	0.230±0.035
FashionMNIST	CSForest	0.050±0.005	**0.266±0.018**	0.038±0.009	**0.311±0.040**
	BCOPS	0.050±0.007	0.381±0.020	0.038±0.009	**0.311±0.040**
	DC	0.051±0.007	0.666±0.033	0.038±0.013	0.584±0.066
	CRF	0.051±0.006	0.514±0.021	0.038±0.014	0.804±0.080
	ACRF	0.051±0.006	0.537±0.013	0.054±0.009	0.835±0.020
	ACRFshift	0.046±0.005	0.481±0.019	0.072±0.021	0.814±0.039

percentage of samples with $\hat{C}(x)$ containing labels other than the true labels. The average length $\hat{C}(x)$ under the mixed distribution $\mu(x)$ is the optimization objective under the formulation described by eq. (3).

4.2.1 The Outliers w/o Shift Setting

In this section, the test set contains outlier labels relative to the training set but without any additional label shift. For each data set, we constructed the training set by including a subset of class labels and the test set with all labels, e.g., for the MINIST data, the training set had digit labels 0-5 of equal size and the test set included both digit labels 0-5 of equal size and digit labels 6-9. Details of the data subsampling schemes for Q1 can be found in Appendix C.

Table 1 displays the average type I and type II errors of all methods on the test data at $\alpha = 0.05$. All methods achieved the the targeted coverage rate at 95% when averaging inlier data. Figure 3(A) presents detailed classification results with different methods on the MNIST dataset. Although ACRF, ACRFshift, and CRF achieved slightly higher accuracy compared to CSForest among inlier digits under no additional label shift, CSForest demonstrated the strongest capability to detect outlier digits 6-9, even compared to BCOPS, with an outlier detection accuracy of approximately 90%. Similar results are observed on other datasets, as detailed in Appendix D.2.

4.2.2 The Shift w/o Outliers Setting

In the previous simulations, although we had outliers, the class ratios among inlier classes were balanced and remained the same for training and test data. To verify whether methods like CSForest, designed to achieve per-class coverage rather than marginal coverage, still maintain robustness when handling label shifts among inlier classes, we examine the predictive performance of all methods under the traditional label shift setting in the absence of outliers. For example, for the MNIST data set, the training and test sets contain digit labels 0-5 but with different class proportions. Details of the data subsampling and label shift schemes for Q2 can be found in Appendix C.

The achieved type I and type II errors using different methods in this standard label shift simulation can be found in Table 1. We observed that all methods achieved the desired coverage $(1-\alpha)$ with ACRFshift exhibiting high variability. However, CSForest is the only method that achieved consistently low type II errors: CRF has a type II error 10% and 50% more than CSForest on MNIST and FashionMNIST, respectively; BCOPS has a type II error 25% more than CSForest on MNIST; ACRF and ACRFshift both have type II errors more than two folds than those from

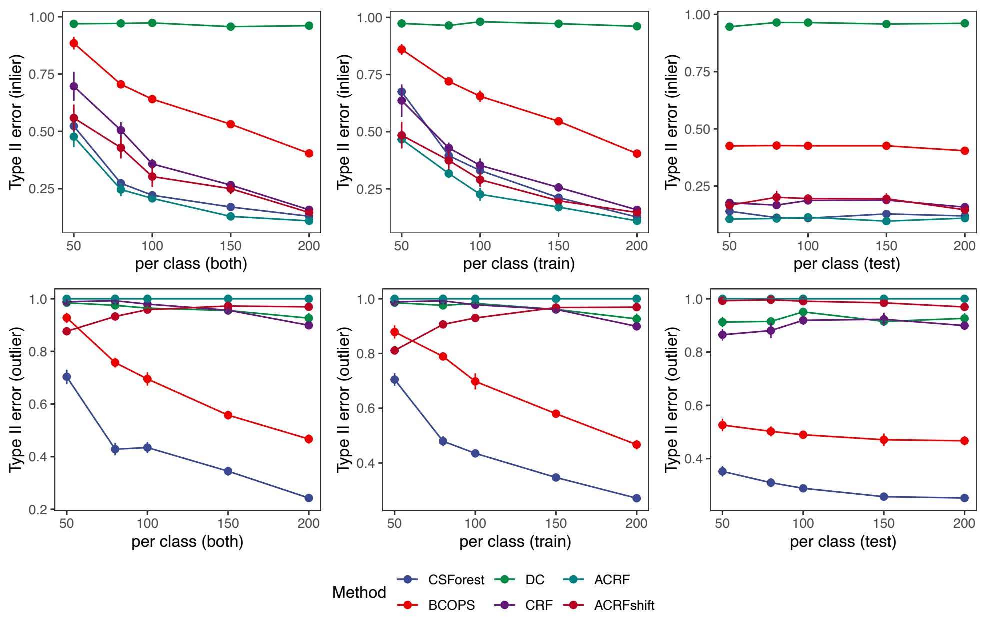


Figure 4: The type II error for inliers and outliers across different sample sizes on MNIST. Figure 4 demonstrates that CSForest outperforms the baselines by efficiently detecting outliers while maintaining lower inlier type II errors across various sample sizes. Note that error bars here are calculated based on repeated sample-splitting and can be smaller than the standard deviation due to sample dependence from different runs.

CSForest on all three data sets. Figure 3(B) presents a detailed view of the prediction results for each class on the MNIST dataset. CSForest contains a higher proportion of samples with only the correct labels in almost every class compared to the baseline models, which underscores the high-quality prediction sets $\hat{C}(x)$ generated by CSForest under Q2, consistent with the lower type II error reported in Table 1. Detailed results on other datasets can be found in Appendix D.2.

4.2.3 Comparisons with Varying Sample Sizes

We conducted a comparison of different methods under varying sample size settings. Specifically, we varied the number of training and test samples per class from 50 to 200. Figure 4 presents the type II errors for inliers and outliers across all models on the MNIST dataset. In Figure 4, it is evident that as training sample size increases, the type II error (inliers) decreases for all methods, while BCOPS and CSForest also benefit from increased test sample sizes. CSForest closely matches the CRF, the best-performing classifier in the inlier classification, for predicting inlier labels as we vary the training/test sample sizes from 50 to 200. CSForest and BCOPs outperformed other methods by a large margin for varying sample sizes for outlier detection, with CSForest significantly improving over BCOPS due to the enhanced data utilization efficiency. Of note, the ability for outlier detection (higher type II error for outliers) ACRFshift deteriorated as sample size increased. This surprising phenomenon is attributed to ACRFshift's decision rule for outliers, which strongly depends on the sample weights under the covariate shift model, denoted as $\gamma_{x_0}(x) = \frac{r(x)}{r(x_0)+\sum_{z_i \in \mathcal{I}_{cal}} r(x_i)}$. A sample is claimed an outlier $\gamma_{x_0}(x)$ is very large, and $\gamma_{x_0}(x)$ tends to increase with increased training sample sizes (Appendix B). Results on other datasets are consistent with those on MNIST (Appendix D.2).

5 DISCUSSION

We propose CSForest, which aims to construct a calibrated and narrow set-valued prediction set under distributional changes, as a powerful ensemble classifier for robust inlier classification and outlier detection. We theoretically justified its robustness for covering the true class label and confirmed its ability to construct high-quality prediction sets compared to alternative methods via extensive experiments.

Future Work. An interesting question is how much guidance from test samples is needed for effective outlier detection. Can CSForest still utilize the limited test samples for efficient outlier detection? As an exploratory experiment, we consider a challenging MNIST example, where we have 200 samples per class for labels 0-5 in the training set but only five samples per class for labels 0-9 in the test set. Figure 5 shows CSForest achieves an average type II error of approximately 42%

for inliers and 60% for outliers, whereas DC exhibits an average type II error of as high as 95%. This highlights the benefits of utilizing a small test set in CSForest and suggesting the potential of extending the framework of CSForest to settings with extremely limited or even single test samples for future exploration.

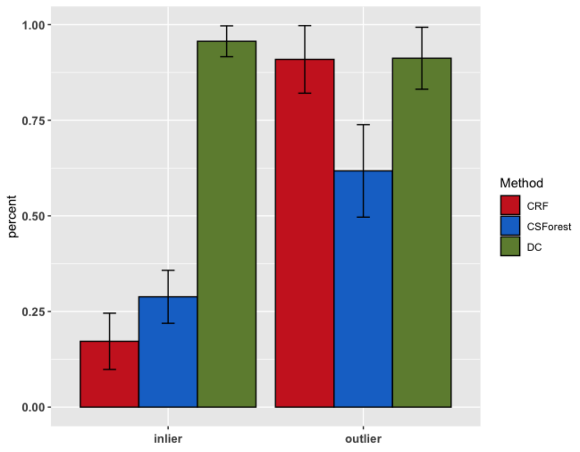


Figure 5: Achieved Type II errors for inliers and outliers across 100 repetitions at $\alpha = 0.05$ with merely 5 samples per-class in the test cohort.

Additionally, the GLS model assumes that the distribution of $x|y$ remains unchanged, which could be violated in practice. When both the distribution of y and the distribution of $x|y$ are allowed to change, the problem becomes much more challenging and less well-defined. One interesting future direction is to relax GLS model and assume bounded small changes in $x|y$, leading to the investigation of CSForest under an adversarial setting that allows adversarial yet small perturbations in $x|y$ during test time.

Acknowledgements

This work was supported by NSF award DMS2310836.

References

Barber, R. F., Candes, E. J., Ramdas, A., and Tibshirani, R. J. (2021). Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486–507.

Bartlett, P. L. and Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(8).

Bickel, S., Brückner, M., and Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10(9).

Cadre, B. (2006). Kernel estimation of density level sets. *Journal of multivariate analysis*, 97(4):999–1023.

Charoenphakdee, N., Cui, Z., Zhang, Y., and Sugiyama, M. (2021). Classification with rejection based on cost-sensitive classification. In *International Conference on Machine Learning*, pages 1507–1517. PMLR.

Chow, C. (1970). On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46.

Cortes, C., DeSalvo, G., and Mohri, M. (2016). Boosting with abstention. *Advances in Neural Information Processing Systems*, 29.

Csurka, G. (2017). Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118.

Gammerman, A., Vovk, V., and Vapnik, V. (2013). Learning by transduction. *arXiv preprint arXiv:1301.7375*.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

Guan, L. and Tibshirani, R. (2022). Prediction and outlier detection in classification problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(2):524–546.

Hechtlinger, Y., Póczos, B., and Wasserman, L. (2018). Cautious deep learning. *arXiv preprint arXiv:1805.09460*.

Herbei, R. and Wegkamp, M. H. (2006). Classification with reject option. *The Canadian Journal of Statistics/La Revue Canadienne de Statistique*, pages 709–721.

Kalra, N. and Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transporta-*

tion Research Part A: Policy and Practice, 94:182–193.

Kim, B., Xu, C., and Barber, R. (2020). Predictive inference is free with the jackknife+-after-bootstrap. *Advances in Neural Information Processing Systems*, 33:4138–4149.

Kompa, B., Snoek, J., and Beam, A. L. (2021). Second opinion needed: communicating uncertainty in medical machine learning. *NPJ Digital Medicine*, 4(1):1–6.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Lei, J. (2014). Classification with confidence. *Biometrika*, 101(4):755–769.

Lei, J. and Wasserman, L. (2015). Distribution-free prediction bands for nonparametric regression. *Quality control and applied statistics*, 60(1):109–110.

Lin, J., Keogh, E., Fu, A., and Van Herle, H. (2005). Approximations to magic: Finding unusual medical time series. In *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pages 329–334. IEEE.

Lipton, Z., Wang, Y.-X., and Smola, A. (2018). Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR.

Marchette, D. J. and Marchette, D. (2001). *Computer intrusion detection and network monitoring: a statistical viewpoint.* Springer.

Neyman, J. and Pearson, E. S. (1933). Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337.

Ni, C., Charoenphakdee, N., Honda, J., and Sugiyama, M. (2019). On the calibration of multiclass classification with rejection. *Advances in Neural Information Processing Systems*, 32.

Papadopoulos, H., Proedrou, K., Vovk, V., and Gammerman, A. (2002). Inductive confidence machines for regression. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*, pages 345–356. Springer.

Qayyum, A., Usama, M., Qadir, J., and Al-Fuqaha, A. (2020). Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. *IEEE Communications Surveys & Tutorials*, 22(2):998–1026.

Romano, Y., Sesia, M., and Candes, E. (2020). Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, 33:3581–3591.

Sadinle, M., Lei, J., and Wasserman, L. (2019). Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525):223–234.

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. (2012). On causal and anticausal learning. *arXiv preprint arXiv:1206.6471.*

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Storkey, A. (2009). When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, 30:3–28.

Tibshirani, R. J., Foygel Barber, R., Candes, E., and Ramdas, A. (2019). Conformal prediction under covariate shift. *Advances in neural information processing systems*, 32.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world.* Springer Science & Business Media.

Vovk, V., Nouretdinov, I., Manokhin, V., and Gammerman, A. (2018). Cross-conformal predictive distributions. In *Conformal and Probabilistic Prediction and Applications*, pages 37–51. PMLR.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747.*

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes/No/Not Applicable] Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes/No/Not Applicable] Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes/No/Not Applicable] Not Applicable

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes/No/Not Applicable] Yes
   (b) Complete proofs of all theoretical results. [Yes/No/Not Applicable] Yes
   (c) Clear explanations of any assumptions. [Yes/No/Not Applicable] Yes

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes/No/Not Applicable] Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes/No/Not Applicable] Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes/No/Not Applicable] Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes/No/Not Applicable] Not Applicable

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes/No/Not Applicable] Yes
   (b) The license information of the assets, if applicable. [Yes/No/Not Applicable] Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes/No/Not Applicable] Not Applicable
   (d) Information about consent from data providers/curators. [Yes/No/Not Applicable] Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Yes/No/Not Applicable] Not Applicable

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Yes/No/Not Applicable] Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Yes/No/Not Applicable] Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Yes/No/Not Applicable] Not Applicable

A PROOFS

A.1 Proof of Proposition 3.1

For the completeness, we provide a reproduction of the proof presented in Guan and Tibshirani (2022),

Proof. We first decompose the problem in eq. (3) into K independent problems for different classes, referred to as the problem P_k:

$$\min \int_x \mathbb{1}_{x \in \mathcal{A}_k} \mu(x) dx, \tag{6}$$

$$s.t.\ \mathbb{P}[x \in \mathcal{A}_k] \geq 1 - \alpha. \tag{7}$$

Let $\mathcal{A}_k$ be the solution to problem P_k, then the solution to problem eq. (3) is $C(x) = \{k : x \in \mathcal{A}_k\}$.

We then define $Q(\alpha, g; F)$ is the lower α percentile of a real-valued function $g(x)$ under distribution F, i.e.,

$$Q(\alpha, g; F) = sup\{t : \mathbb{P}_F(g(x) \leq t) \leq \alpha\}. \tag{8}$$

Following Guan and Tibshirani (2022), we regard the problem P_k as a hypothesis testing problem where the null hypothesis is $H_0 : x \sim f_k$ and the alternative is $H_1 : x \sim \mu_k$. The optimal solution of P_k is $\mathcal{A}_k$ which is actually the decision region of above hypothesis with the most powerful level α. Therefore, by Neyman–Pearson Lemma (Neyman and Pearson, 1933), we can construct the likelihood ratio statistic $s(x, k; \mu) = f_k(x)/\mu(x)$ and have the solution $\mathcal{A}_k = \{s(x, k; \mu) \leq Q(\alpha, s_k; F_k)\}$ where the s_k is the conformal score function of class k and F_k is the distribution of x from class k. Hence, the solution to eq. ((3) is $C(x) = \{k : \mathbb{E}_X[\mathbb{1}\{s(x, k; \mu) \geq s(X, k; \mu)\}|Y = k] \geq \alpha, k = 1, \ldots, K\}$.

□

A.2 Proof of Theorem 3.3

Proof. Here, we prove eq. (5) for any given class k and the test sample x_i. The original procedure for determining whether we should assign class k to sample x_i can also be described as following. First, generate two events, $\mathcal{E}_1$ and $\mathcal{E}_2$:

1. Event $\mathcal{E}1$: Training samples other than class k and bootstrap copies $\mathcal{I}^b_{other}$ for $b = 1, \ldots, \tilde{B}$.
2. Event $\mathcal{E}2$: Test samples other than x_i and bootstrap copies $\mathcal{I}^b_{test}$ for $b = 1, \ldots, \tilde{B}$.

Let $\mathcal{I}^b_k$ for $b = 1, \ldots, \tilde{B}$ represent $\tilde{B}$ bootstrap copies of training class k samples. We conduct our conformalized classification using only copies b with $v_b = 1$ for $b = 1, \ldots, \tilde{B}$, where $v_b \sim$ Bernoulli$((1-\frac{1}{n_k+1})^{n_k})$. The comparison between $x_{i'}$ from class k and the test sample x_i is performed by aggregating prediction functions using runs b with $v_b = 1$ and excluding both $x_{i'}$ and x_i: $\hat{G}^{ii'}(x; \mu) = \varphi(\hat{G}^b_k(x; \mu) : b$ satisfies $v_b = 1, i' \notin \mathcal{I}^b_k, i \notin \mathcal{I}^b_{te})$.

We will now condition on $\mathcal{E}_1$ and $\mathcal{E}_2$, and define $\mathcal{B}_i = \{b : i \notin \mathcal{I}^b_{te}\}$. The function $\hat{G}^{ii'}(x; \mu)$ can be rewritten as $\hat{G}^{ii'}(x; \mu) = \varphi(\hat{G}^b_k(x; \mu) : b \in \mathcal{B}_i$ satisfies $v_b = 1, i' \notin \mathcal{I}^b_k)$. A key observation is that this can be equivalently expressed as first sampling $B \sim$ Binomial$(|\mathcal{B}i|, (1 - \frac{1}{n_k+1})^{n_k})$ and constructing the ensemble prediction function comparing x_i and $x_{i'}$: $\hat{G}^{i'}(x; \mu) = \varphi(\hat{G}^b_k(x; \mu) : b$ satisfies $i' \notin \mathcal{I}^b_k, 1 \leq b \leq B)$. We have also dropped the superscript i since this dependence disappears after conditioning and restricting ourselves to $\mathcal{B}_i$.

Interestingly, under this new equivalent characterization, sampling $B \sim$ Binomial$(|\mathcal{B}_i|, (1 - \frac{1}{n_k+1})^{n_k})$ followed by bootstrap B copies of the n_k class k samples is equivalent to drawing $|\mathcal{B}i|$ bootstrap copies of the $n_k + 1$ samples, which include n_k samples from training class k and the test sample x_i, and then keeping only those bootstrap samples where x_i is not included. A similar equivalence was first noted by Kim et al. (2020) and utilized in the Jacknife+aB procedure for traditional supervised regression.

In summary, the decision rule in CSForest for whether to include label k in $\hat{C}(x_i)$ can be equivalently expressed with the following procedure after conditioning on $\mathcal{E}_1$ and $\mathcal{E}_2$:

1. Conduct $|\mathcal{B}i|$ bootstrap resamplings of the n_k+1 samples. Denote these samples as $x_1, \ldots, x_{n_k}$ (representing n_k class k training samples) and $x_{n_k+1} \leftarrow x_i$ as the test sample x_i. Let $\tilde{I}_b$ be the index of samples in the b^{th} bootstrap.

2. For each bootstrap, construct a random forest tree $\tilde{G}^b(x)$ separates $x_{\tilde{I}_b}$ from other samples (conditioned on).

3. For each pair (l, j) with $1 \leq l, j \leq n_k + 1$, construct $\hat{G}^{lj}(x) = \varphi(\{\tilde{G}_k^b(x) : l, j \notin \tilde{I}_b\})$ and include label k if and only if
$$1 + \sum_{j=1}^{n} \mathbb{1}\{\hat{G}^{n_k+1,j}(x_{n_k+1}) \geq \hat{G}^{j,n_k+1}(x_j)\} \geq (n_k+1)\alpha.$$

Note that $\hat{s}^{ii'}(x, k; \mu)$ is the same as $G^{n_k+1,i'}(x_{n_k+1})$.

Define $A_{lj} = \mathbb{1}\{\hat{G}^{l,j}(x_l) \geq \hat{G}^{j,l}(x_j)\}$ for all $1 \leq l, j \leq n_k + 1$. It is obvious that $A_{ii} = 1$ for all $i = 1, \ldots, n+1$. Define $A_{l\bullet} = \sum_{j=1}^{n_k+1} A_{lj}$ as the sum of the l^{th} row from the comparison matrix A. Then,
$$k \notin \hat{C}_l(x_{n_k+1}) \quad \text{if and only if} \qquad A_{n_k+1\bullet} \leq (n_k+1)\alpha - 1.$$

Hence, eq. (5) from Theorem 3.3 is equivalent to (9) below:
$$\mathbb{P}(A_{n_k+1\bullet} \leq (n_k+1)\alpha - 1 | y_{n_k+1} = k) \leq 2\alpha. \tag{9}$$

We now proceed to prove (9), which consists of two steps (1) $A_{j\bullet}$ are exchangeable with each other for $j = 1, \ldots, n+1$ when $y_{n_k+1} = k$, and (2) the strange set $S(A) = \{j : A_{j\bullet} \leq (n_k+1)\alpha - 1\}$ satisfies $|S(A)| \leq 2\alpha(n_k+1)$. Combining these two steps, we immediately have
$$\begin{aligned}&\mathbb{P}\left[A_{n_k+1\bullet} \leq (n_k+1)\alpha - 1 | y_{n_k+1} = k\right] \\ =&\mathbb{P}\left[(n_k+1) \in S(A) | y_{n_k+1} = k\right] = \frac{|S(A)|}{n_k+1} \leq 2\alpha.\end{aligned}$$

At this stage, proofs to above two steps (1) and (2) become identical to that used in the proofs to Theorem 1 in Barber et al. (2021) or Theorem 1 in Kim et al. (2020).

□

B MORE DETAILS ON BASELINES

In this section, we provide more details about the baseline models. We first introduce several existing methods for constructing set-valued predictions $C(x)$, including BCOPS, CRF, DC, and ACRFrandom.

B.1 BCOPS, CRF, DC and ACRFrandom

- BCOPS (Balanced Conformalized Optimal Prediction Sets) (Guan and Tibshirani, 2022) is a semi-supervised classifier that utilizes half of the training data to train a classifier, separating observed classes from unlabeled test samples. The remaining half of the training samples is used for calibration through conformal prediction and constructing a set-valued prediction set. In contrast, BCOPS focuses on optimizing model performance on the test set, setting $\mu(x) = f_{te}(x)$, which represents the marginal density for the test data. It constructs calibrated set-valued predictions by combining empirically estimated $v_k(x)$ with the sample-splitting conformal prediction method(Vovk et al., 2005). While BCOPS excels in abnormality detection, outperforming non-test-cohort-adaptive methods, it relies on having a large set of test data, and the sample-splitting scheme results in lower data utilization efficiency(Guan and Tibshirani, 2022).

- CRF (Comformalized Random Forest)(Vovk et al., 2005) constructs the set-valued prediction $\{k : \hat{p}_k(x) \geq \tau_k\}$ by including training labels k achieving high estimated probability from the random forest classifier, with the cut-offs τ_k chosen based on sample-splitting conformal prediction.

- DC (Density-set Classifier)(Hechtlinger et al., 2018) constructs the set-valued prediction similarly to CRF, except for replacing the estimated probability $\hat{p}_k(x)$ by an estimation of the density function for class k using the training data.

- ACRFrandom: We refer to the adaptive-coverage classification approach using random forest proposed in Romano et al. (2020) as ACRFrandom (Adaptive-coverage CRF with randomization) where the randomization is introduced via an additional uniform random variable U for tie-breaking. ACRFrandom constructs the prediction set $\hat{C}(x)$ by including labels with large estimated probabilities such that the total probability is greater than the upper-level quantile of the empirical distribution of $\{E_i\}_{i\in\mathcal{I}_{cal}} \cup \{\infty\}$ where $\mathcal{I}_{cal}$ is the calibration set in sample-splitting conformal prediction and E_i is the sum of estimated probabilities for all labels proceeding that for the true label.

B.2 ACRF and ACRFrandom

We further introduce the baseline ACRFrandom and its de-randomized version ACRF.

ACRFrandom

In the original proposal of Romano et al. (2020), the authors assume that training and test data to have the same distribution. ACRFrandom defines a function $\mathcal{S}$ with input x, $u \in [0,1]$, the conditional probability $\pi_y = P(Y=y|X=x)$, and the threshold τ. Define

$$\mathcal{S}(x,u;\pi,\tau) = \begin{cases} y \textit{ indices of the } L(x;\pi,\tau)-1 \textit{ largest } \pi_y(x), & u < V(x;\pi,\tau) \\ y \textit{ indices of the } L(x;\pi,\tau) \textit{ largest } \pi_y(x), & \textit{otherwise} \end{cases} \tag{10}$$

where

$$V(x;\pi,\tau) = \frac{\sum_{c=1}^{L(x;\pi,\tau)} \pi_{(c)}(x) - \tau}{\pi_{(L(x;\pi,\tau))}(x)} \tag{11}$$

$$L(x;\pi,\tau) = \min\{c \in \{1,\cdots,C\} : \pi_{(1)}(x) + \pi_{(2)}(x) + \cdots + \pi_{(c)}(x) \geq \tau\}. \tag{12}$$

and $\pi_{(i)}(x)$ is the *ith* largest conditional probability.

Further, ACRFrandom defines the generalized inverse quantile conformity score function E,

$$E(x,y,u;\hat{\pi}) = \min\{\tau \in [0,1] : y \in \mathcal{S}(x,u;\hat{\pi},\tau)\}. \tag{13}$$

And the empirical distribution is

$$V(x,y;E) = \frac{1}{|\mathcal{I}_{cal}|+1} \sum_{i\in\mathcal{I}_{cal}} \delta_{E_i} + \frac{1}{|\mathcal{I}_{cal}|+1}\delta_\infty \tag{14}$$

where E_i is constructed at the minimum τ for the calibration sample i such that y_i is included in $S(x_i,u_i;\hat{\pi},\tau)$ and δ_i denotes a point mass at i. The final prediction set is constructed as $\hat{C}(x) = \mathcal{S}(x,u;\pi,\hat{\tau}_\alpha)$, where $\hat{\tau}_\alpha$ is the upper level α quantile of the empirical distribution $\{E_i\}_{i\in\mathcal{I}_{cal}} \cup \{\infty\}$.

ACRF

We consider a derandomized version of ACRF without the uniform variable U in our experiment. For ACRF, we define

$$\mathcal{S}(x;\pi,\tau) = \{y \textit{ indices of the } L(x;\pi,\tau) \textit{ largest } \pi_y(x)\}, \tag{15}$$

where

$$L(x;\pi,\tau) = \min\{c \in \{1,\cdots,C\} : \pi_{(1)}(x) + \pi_{(2)}(x) + \cdots + \pi_{(c)}(x) > \tau\}. \tag{16}$$

We can similarly define a score function E,

$$E(x, y; \hat{\pi}) = \min\{\tau \in [0, 1] : y \in \mathcal{S}(x; \hat{\pi}, \tau)\}, \tag{17}$$

and construct E_i as the the minimum τ for the calibration sample i such that y_i is included in $S(x_i; \hat{\pi}, \tau)$. Same as in ACRFrandom, ACRF constructs as $\hat{C}(x) = \mathcal{S}(x; \pi, \hat{\tau}_\alpha)$, where $\hat{\tau}_\alpha$ is the upper level α quantile of the empirical distribution $\{E_i\}_{i\in\mathcal{I}_{cal}} \cup \{\infty\}$. Algorithm 2 shows the details of ACRF.

Algorithm 2 Implementation of ACRF

```
Input  : Training Data $\{z_i := (x_i, y_i)_{i=1}^n, i \in \mathcal{I}_{tr}\}$, Test Data $\{(x_i)_{i=1}^m, i \in \mathcal{I}_{te}\}$.
Output : Prediction sets $\hat{C}_i(x_i)$ for $i \in \mathcal{I}_{te}$.
1 Randomly split the training data into 2 subsets, the training set $\mathcal{I}_{tr}^1$, the calibration training set $\mathcal{I}_{tr}^2$.
2 Train random forest model $\mathcal{B}$ on all samples in $\mathcal{I}_{tr}^1$: $\hat{\pi}_1 \leftarrow \mathcal{B}((X_i, Y_i)_{i\in\mathcal{I}_{tr}^1})$.
3 Predict on $\mathcal{I}_{tr}^2$: $\hat{\pi}_2 \leftarrow \mathcal{B}((X_i)_{i\in\mathcal{I}_2})$ and $\mathcal{I}_{te}$: $\hat{\pi}_{te} \leftarrow \mathcal{B}((X_i)_{i\in\mathcal{I}_{te}})$.
4 Construct $\{E_i\}_{i\in\mathcal{I}_{tr}^2}$ treating $\mathcal{I}_{tr}^2$ as the calibration set.
5 Compute the level $(1-\alpha)$ quantile of the empirical distribution $\{E_i\}_{i\in\mathcal{I}_{tr}^2} \cup \{\infty\}$.
6 Use the function $\mathcal{S}$ defined in eq. (15) to construct the prediction set at $x_i \in \mathcal{I}_{te}$ as $\hat{C}_i(x_i) = \mathcal{S}(X_i; \hat{\pi}_{te}^i, \hat{\tau}_\alpha)$.
```

B.3 ACRFshift

Finally, we introduce ACRFshift, another baseline that explicitly accounts for distributional changes under covariate shift model. ACRFshift combines the covariate shift comformal prediction (Tibshirani et al., 2019) with ACRF. This has not been discussed in previous work, so we give details about how this is done in our paper. We split also the test samples into two sets $\mathcal{I}_{te}^1$, $\mathcal{I}_{te}^2$. Suppose that we now construct the prediction set for test samples in $\mathcal{I}_{te}^2$.

Instead of finding τ with eq. (17) and (14), we consider the following weighted calibration. The weighted function is $\gamma_x(x) = \frac{r(x)}{r(x)+\sum_{z_i\in\mathcal{I}_2} r(x_i)}$ and $r(x) = \frac{\mathbb{P}[W=1|x]}{\mathbb{P}[W=0|x]}$ is the conditional probability of being generated from the test data ($W = 1$ means from the test data, and $W = 0$ represent from the training data), learned from the classifier separating the test data $\mathcal{I}_{te}^1$ from the training data $\mathcal{I}_{tr}^1$. Instead of consider $\hat{\tau}_\alpha$ as the level $(1-\alpha)$ quantile of the empirical distribution $\{E_i\}_{i\in\mathcal{I}_{tr}^2} \cup \{\infty\}$, for any $x \in \mathcal{I}_{te}^2$, we consider the level $(1-\alpha)$ quantile of the weighted distribution below:

$$V_w(x, y; E) = \sum_{i\in\mathcal{I}_{tr}^2} \gamma_x(x_i)\delta_{E_i} + \gamma_x(x)\delta_\infty.$$

Similarly, whether to include the random variable U in ACRF will result in two versions: ACRFshiftrandom and ACRFshift.

Unlike CSForest, BCOPS, and even CRF, which naturally considers samples with $\emptyset$ as the ones not close to inlier classes, and thus, outliers, ACRF and ACRFshift both consider the conditional probability of $y|x$ and do not have such a feature encoded in their constructions. Hence, we adopt the rule where we reject a sample when $r_x(x)$ is very large compared to others with $\sum_{i\in I_{tr}^2} r_x(x_i) < \tau$. (Recall that $r_x(x_i) = \frac{1}{n+1}$ without covariate shift.)

B.4 Randomized ACRF/ACRFshift vs. derandomized ACRF/ACRFshift

In this section, we further demonstrate the difference between the randomized version and derandomized version to support our claim that removing the random variable U from ACRFrandom and ACRFshift helps achieve a desirable type II error.

Table 2 shows the type I error and type II error of ACRF/ACRFrandom and ACRFshift/ACRFshiftrandom (referred to as derandomized and randomized versions of ACRF/ACRFshift in Table 2). We observe that models with randomness and those without randomness achieve comparable type I errors when there is not shift in the inlier labels, while models with randomness tend to have worse type II errors and the final prediction set $\hat{C}$ is more likely to contain multiple labels. One potential explanation for this is that randomization can help with the control of conditional coverage and may lead to increased high type II error due to this more ambitious goal. Supporting this, the randomized version for both ACRF and ACRFshift controls the type I error while the derandomized version, especially ACRF, shows inflated type I error under the label shift model.

Table 2: Randomized version vs. derandomized version: Achieved Type I and Type II errors at $\alpha = 0.05$ with outlier components and no additional label shift among inlier digits and achieved Type I and Type II errors at $\alpha = 0.05$ with label shift among inlier digits but no outlier digits.

Version	Method	no additional label shift		additional label shift	
		Type I	Type II	Type I	Type II
Randomized	ACRF	0.049±0.007	0.702±0.017	0.025±0.007	0.884±0.014
	ACRFshift	0.053±0.007	0.681±0.014	0.055±0.013	0.828±0.015
derandomized	ACRF	0.047±0.006	0.431±0.003	0.171±0.024	0.313±0.067
	ACRFshift	0.036±0.009	0.439±0.009	0.080±0.026	0.630±0.127

C TRAINING DETAILS

In this section, we provide a detailed description of how we constructed datasets satisfying GLS and label shift using MNIST, CIFAR-10, and FashionMNIST for experimentation.

Q1 outliers w/o shift. For the datasets MNIST, CIFAR-10, and FashionMNIST, each consisting of 10 categories, we sampled 500 samples from each class (categories 0-5) to create a training set of 2500 samples. From the remaining samples in categories 0-5, we randomly selected 500 samples from each class, and similarly, we randomly selected 500 samples from each class in categories 6-9. These 5000 samples formed the test set. Notably, in the test set, categories 6-9 represent outliers that never appeared in the training set, while categories 0-5 are inlier samples.

Q2 shift w/o outliers. For the Label Shift setup, we sampled 500 samples from each class (categories 0-5) from MNIST, CIFAR-10, and FashionMNIST to create a training set of 3000 samples. From the remaining samples in categories 0-5, we randomly selected 100 samples from each class, and similarly, we randomly selected 500 samples from each class in categories 6-9. These 3000 samples formed the test set.

It is important to emphasize that for MNIST, CIFAR-10, and FashionMNIST, both CSForest and the baseline methods utilized representations extracted by a pre-trained ResNet-18 model as inputs, rather than the original images.

D ADDITIONAL EXPERIMENTS RESULTS

In the following section, we present additional experimental results to further substantiate the conclusions made in Section 4.

D.1 More Datastes

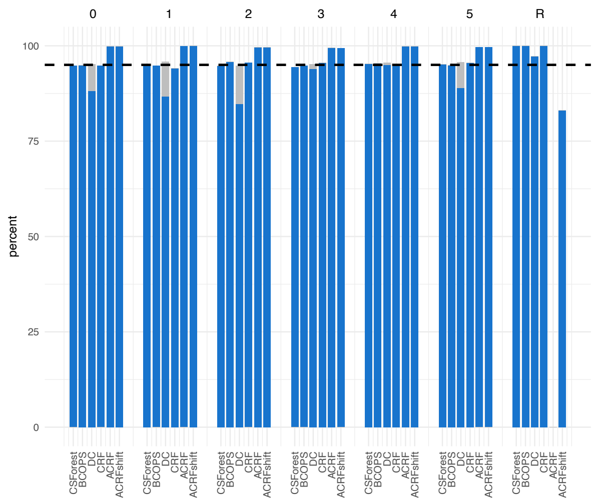
To further illustrate the effectiveness of CSForest under different tasks, we additionally include experiments on two new datasets: a cyber/network intrusion dataset from the KDD data competition and a chest X-ray dataset from the medical domain. Results in Tabel 3 shows CSForest achieves low type I errors and minimizes type II errors in both datasets, demonstrating its superior capability for outlier detection compared to the baseline.

Table 3: The achieved type I and type II errors at $\alpha = 0.05$ under no additional distribution shift but with outliers in the test set across 10 repetitions. For Network Intrusion, the test set includes additional 15 intrusion types as outliers; for Chest X-ray, the test set includes lung abnormalities caused by viruses as outliers.

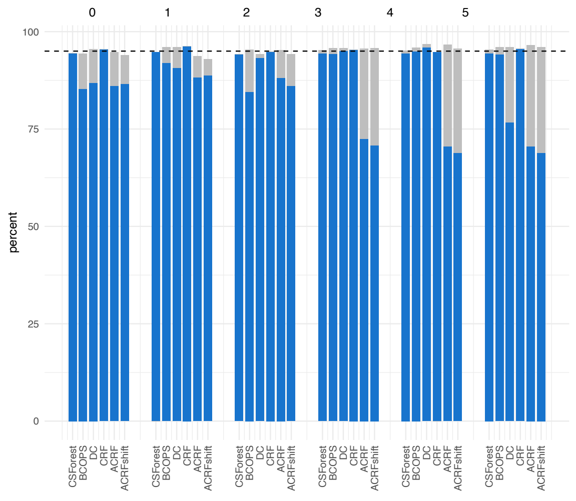
Method	Network Intrusion		Chest X-ray	
	Type I Error	Type II Error	Type I Error	Type II Error
CSForest	0.048 ± 0.013	**9.524e-5 ± 0.000**	0.056 ± 0.003	**0.566 ± 0.002**
BCOPS	0.047 ± 0.014	5.905e-4 ± 0.000	0.059 ± 0.005	0.576 ± 0.004
DC	0.049 ± 0.010	0.261± 0.049	0.062 ± 0.010	0.728 ± 0.013
CRF	0.030 ± 0.004	0.467± 0.104	**0.039 ± 0.033**	0.793 ± 0.178
ACRF	0.000 ± 0.000	0.857± 0.000	0.067 ± 0.046	0.922 ± 0.005
ACRFshift	0.001 ± 0.001	0.019 ± 0.004	0.039 ± 0.049	0.885 ± 0.057

D.2 Per-class Quality Evaluation

Figures 6 and 7 provide a detailed breakdown of the predictions made by all methods for each class in the CIFAR-10 and FashionMNIST datasets. Consistent with the results observed in MNIST, we find that CSForest is the top-performing method for outlier detection, and it avoids over-predicting by generating prediction sets that predominantly contain only the correct labels for each class.

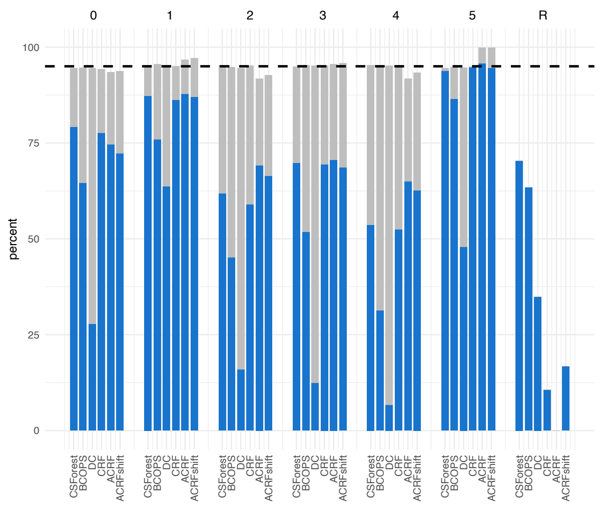


(a) Per-class quality evaluation with outliers but no additional label shift among inlier digits, where the outliers are defined as $R = \{6, 7, 8, 9\}$.

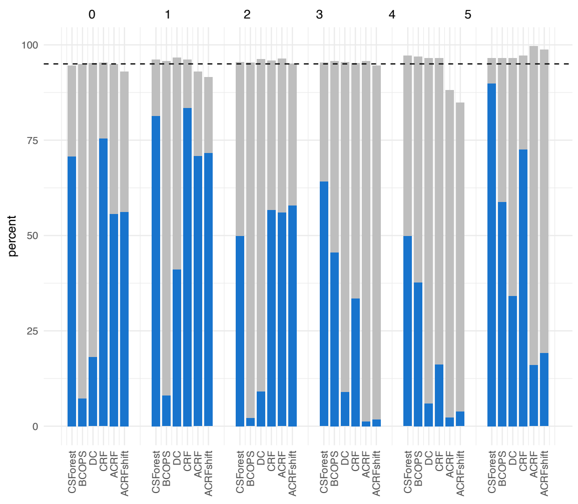


(b) Per-class (class 0-5) quality evaluation with additional label shift among inlier digits but no outliers.

Figure 6: Per-class quality evaluation on CIFAR-10. Figure 6 is grouped by the actual labels in the testing data and colored based on if a prediction set contains only the correct label (blue) or more than the correct label (gray). The horizontal dash line refers to the coverage level of 95%.



(a) Per-class quality evaluation with outliers but no additional label shift among inlier digits, where the outliers are defined as $R = \{6, 7, 8, 9\}$.



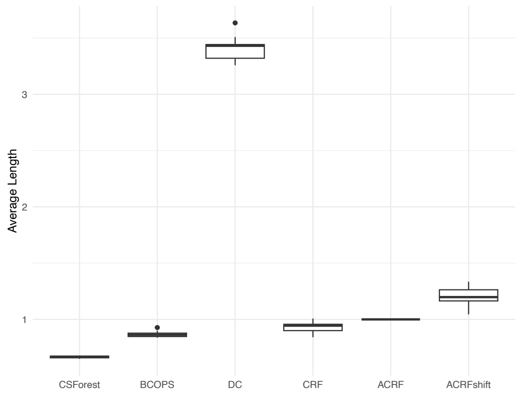
(b) Per-class (class 0-5) quality evaluation with additional label shift among inlier digits but no outliers.

Figure 7: Per-class quality evaluation on FashionMNIST. Figure 7 is grouped by the actual labels in the testing data and colored based on if a prediction set contains only the correct label (blue) or more than the correct label (gray). The horizontal dash line refers to the coverage level of 95%.

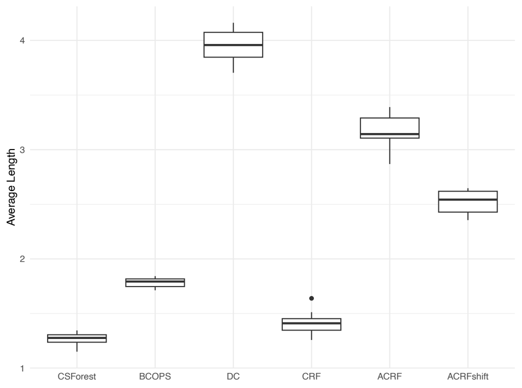
D.3 Average Length of the Prediction Set $\hat{C}$

We observe that across all datasets, whether in the setting with outliers and no additional label shift or without outliers and with additional label shift, CSForest consistently achieves the smallest average prediction set interval length. This indicates that CSForest's predictions do not contain a significant amount of redundant information, aligning with our previous observations of CSForest containing more "only correct labe" content for each class and exhibiting lower type II error.

To further illustrate that CSForest's prediction set $\hat{C}$ results in more accurate label predictions (i.e., predominantly containing only the correct labels) compared to other methods, Figure 8, 9 and 10 visualize the average interval length of prediction sets $\hat{C}$ for all methods.

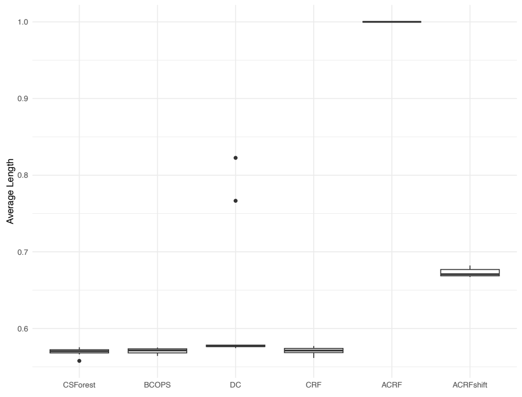


(a) Per-class quality evaluation of all methods with outlier components but no additional label shift among inlier digits.

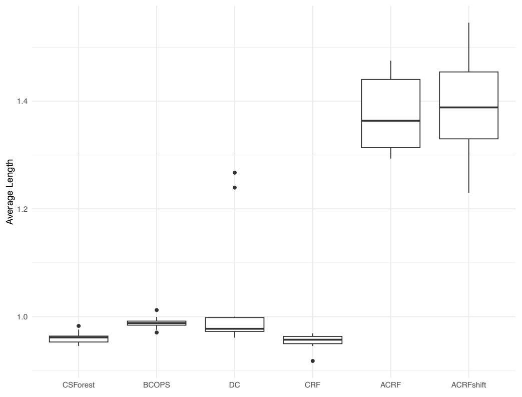


(b) Average length of the prediction set $\hat{C}$ of all methods with additional label shift among inlier digits but no outlier components.

Figure 8: Average length of the prediction set $\hat{C}$ on MNIST. For MNIST, in both settings, CSForest achieves the smallest average prediction set interval length, which aligns with the high "only correct labels" content demonstrated in Figure 3 for CSForest across all classes. This conclusion is further supported by the lower type II error exhibited by CSForest in Table 1.

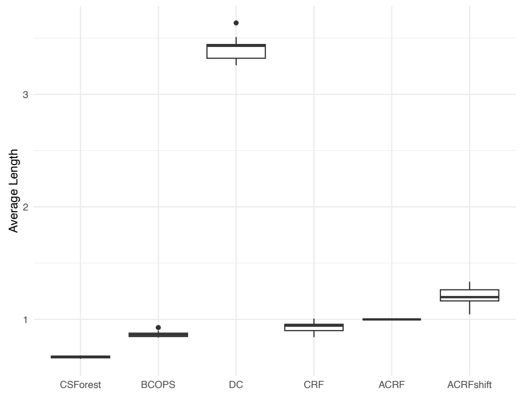


(a) Per-class quality evaluation of all methods with outlier components but no additional label shift among inlier digits.

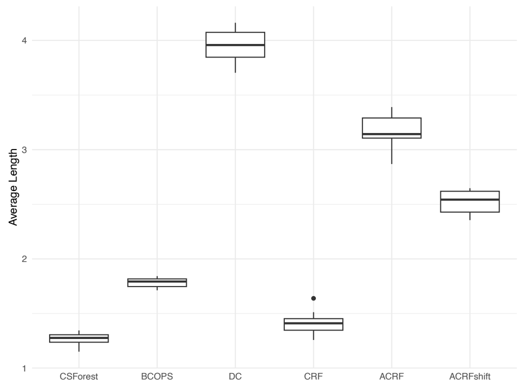


(b) Average length of the prediction set $\hat{C}$ of all methods with additional label shift among inlier digits but no outlier components.

Figure 9: Average length of the prediction set $\hat{C}$ on CIFAR-10. For CIFAR-10, in both settings, CSForest achieves the smallest average prediction set interval length, which aligns with the high "only correct labels" content demonstrated in Figure 6 for CSForest across all classes. This conclusion is further supported by the lower type II error exhibited by CSForest in Table 1.



(a) Per-class quality evaluation of all methods with outlier components but no additional label shift among inlier digits.



(b) Average length of the prediction set $\hat{C}$ of all methods with additional label shift among inlier digits but no outlier components.

Figure 10: Average length of the prediction set $\hat{C}$ on FashionMNIST. For FashionMNIST, in both settings, CSForest achieves the smallest average prediction set interval length, which aligns with the high "only correct labels" content demonstrated in Figure 7 for CSForest across all classes. This conclusion is further supported by the lower type II error exhibited by CSForest in Table 1.

D.4 CSForest's Performance with Varying Sample Size

In this section, we present the type II (inlier and outlier) error curves for all methods on CIFAR-10 and FashionMNIST as sample sizes vary. Consistent with the experimental results on MNIST, CSForest demonstrates superior outlier detection capabilities relative to the baseline as sample sizes change, and it also maintains lower inlier type II errors.

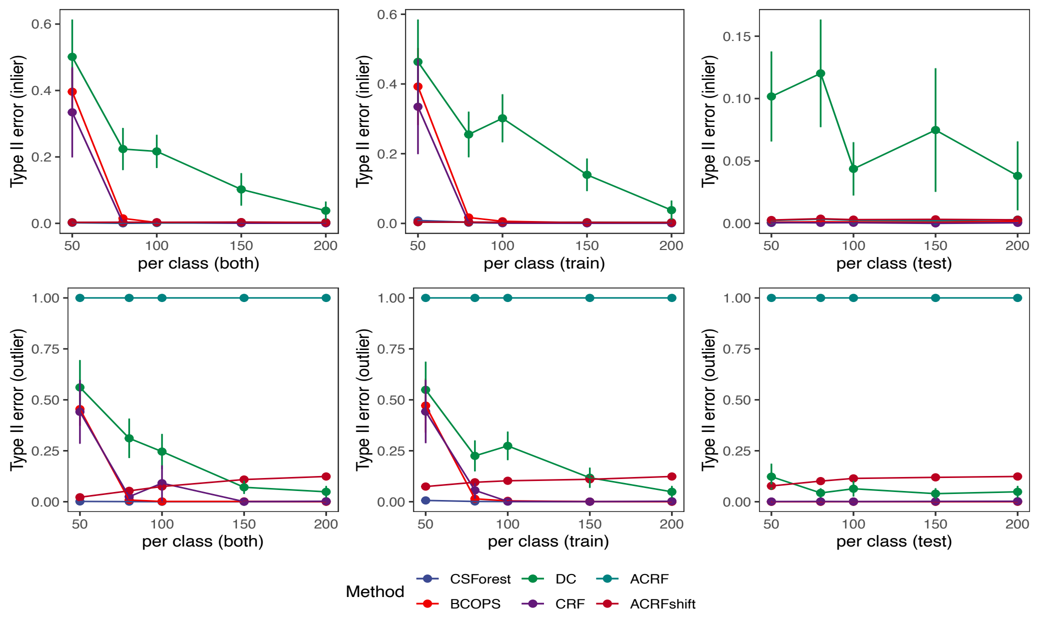


Figure 11: The type II error for inliers and outliers obtained under different sample sizes on CIFAR-10. Figure 11 illustrates that, compared to baselines, CSForest efficiently detects outliers while maintaining lower inlier type II errors across varying sample sizes.

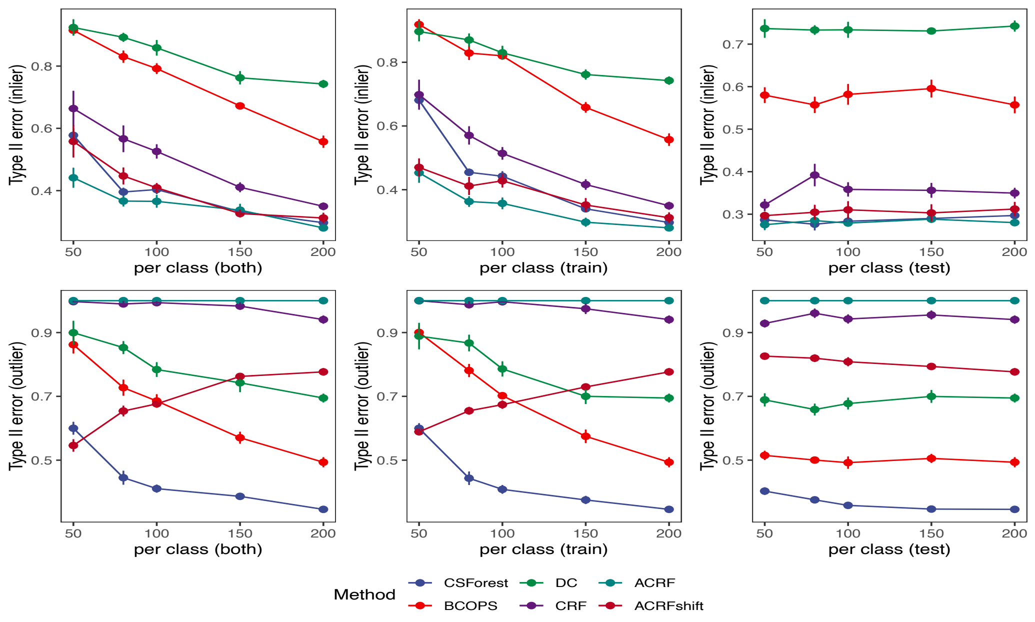


Figure 12: The type II error for inliers and outliers obtained under different sample sizes on FashionMNIST. Figure 12 illustrates that, compared to baselines, CSForest efficiently detects outliers while maintaining lower inlier type II errors across varying sample sizes.

E DISCUSSION ON w

To avoid oversampling, we impose constraints on $\tilde{\mathcal{I}}_{other}$, where the sample size for other classes is constrained to $\min(\lceil n_{te}w \rceil, n - n_k)$. That is, we weigh the training samples through sampling, and we will cap the influence of training samples if even taking all training samples becomes insufficient. So the threshold of satisfies $\lceil n_{te}w \rceil = n - n_k$. As a result, CSForest will yield the same Type I and Type II error once ω exceeds a certain threshold depending on the data.

Assuming a training dataset with K classes and T samples per class, and a test dataset with K' classes and T' samples per class. We have:

$$\begin{aligned} & n_{te}\omega + 1 \geq n - n_k \geq n_{te}\omega \\ \Longrightarrow\ & K'T'\omega + 1 \geq KT - T \geq K'T'\omega \\ \Longrightarrow\ & \frac{(K-1)T}{K'T'} \geq \omega \geq \frac{(K-1)T - 1}{K'T'} \end{aligned} \tag{18}$$

Based on eq. (18), we have the following conclusions:

1. If ω exceeds the threshold $\frac{(K-1)T}{K'T'}$, increasing ω will have no effect on CSForest.

2. If the test sample size $n_{te} = K'T' \gg KT = n_{tr}$ where n_{tr} is the training sample size, $\omega \to 1$. In this case, greater than 1 will achieve the same errors as $\omega = 1$.

In this section, we experiment with different choices of weights w from small 0 to large (exceeding the weight threshold) as opposed to fixing $w = 1$ at the default value. Although small w can sometimes lead to improved outlier detection and large w can sometimes improve the inlier classification, $w = 1$ tend to provides a good tradeoff between these two objectives on the three real data sets considered.

Table 4: Achieved Type I and Type II errors at $\alpha = 0.05$ and $\omega \geq 1$ on MNIST. We observed that when $\omega \geq 2$, CSForest achieves the same Type I and Type II error as $\omega \geq 2$.

ω	Type I Error	Type II Error (inlier)	Type II Error (outlier)
0	0.057±0.018	0.251±0.033	0.314±0.068
LOG	0.053±0.016	0.224±0.031	0.315±0.061
1	0.058±0.014	0.119±0.018	0.346±0.065
1.5	0.055±0.016	0.106±0.014	0.349±0.068
2	0.056±0.018	0.099±0.016	0.373±0.072
5	0.056±0.018	0.099±0.016	0.373±0.072
10	0.056±0.018	0.099±0.016	0.373±0.072
100	0.056±0.018	0.099±0.016	0.373±0.072

Table 5: Achieved Type I and Type II errors at $\alpha = 0.05$ and $\omega \geq 1$ on CIFAR-10. We observed that when $\omega \geq 2$, CSForest achieves the same Type I and Type II error as $\omega \geq 2$.

ω	Type I Error	Type II Error (inlier)	Type II Error (outlier)
0	0.045±0.015	0.001±0.001	0.000±0.000
LOG	0.043±0.016	0.000±0.001	0.000±0.000
1	0.043±0.016	0.000±0.001	0.000±0.000
1.5	0.043±0.014	0.000±0.001	0.000±0.000
2	0.043±0.016	0.000±0.001	0.000±0.000
5	0.043±0.016	0.000±0.001	0.000±0.000
10	0.043±0.016	0.000±0.001	0.000±0.000
100	0.043±0.016	0.000±0.001	0.000±0.000

E.1 MNIST

For the MNIST data, we consider $K = 6, T = 200, K' = 10, T' = 50$ and we can get the threshold $1.998 < w < 2.000$. For the MNIST dataset, we indeed observed in Table 4 that once $w \geq 2$, the type I error and type II error of CSForest no longer change.

E.2 CIFAR-10

For CIFAR-10, we consider $K = 6, T = 200, K' = 10, T' = 50$ and get the threshold $1.998 < w < 2.000$. Similarly to MNIST, for CIFAR-10, once w exceeds the threshold of 2, the performance of CSForest remains unchanged.

E.3 FashionMNIST

For the CIFAR-10 data, we consider $K = 6, T = 200, K' = 10, T' = 50$ and get the threshold $1.998 < w < 2.000$. For FashionMNIST as well, the performance of CSForest remains constant once w surpasses the threshold of 2.

Table 6: Achieved Type I and Type II errors at $\alpha = 0.05$ and $\omega \geq 1$ on FashionMNIST. We observed that when $\omega \geq 2$, CSForest achieves the same Type I and Type II error as $\omega \geq 2$.

ω	Type I Error	Type II Error (inlier)	Type II Error (outlier)
0	0.049±0.011	0.421±0.001	0.406±0.045
LOG	0.048±0.012	0.391±0.047	0.399±0.043
1	0.046±0.012	0.287±0.028	0.403±0.037
1.5	0.047±0.010	0.272±0.029	0.408±0.038
2	0.048±0.012	0.262±0.032	0.410±0.034
5	0.048±0.012	0.262±0.032	0.410±0.034
10	0.048±0.012	0.262±0.032	0.410±0.034
100	0.048±0.012	0.262±0.032	0.410±0.034