Conformal Anomaly Detection in Python: Moving Beyond Heuristic Thresholds with `nonconform`

Oliver Hennhöfer OLIVER.HENNHOEFER@H-KA.DE
Maximilian Kirsch MAXIMILIAN.KIRSCH@H-KA.DE
Christine Preisach CHRISTINE.PREISACH@H-KA.DE
Intelligent Systems Research Group,
Karlsruhe University of Applied Sciences, Germany

Editor: Ernst Ahlberg, Ulf Johansson, Henrik Boström, Alberto Carlevaro, Johan Hallberg Szabadváry and Lars Carlsson

Abstract

Most anomaly detection systems output scores rather than calibrated decisions, leaving practitioners to choose thresholds heuristically and without clear statistical interpretation. Conformal anomaly detection addresses this limitation by converting anomaly scores into calibrated p-values that are valid under the statistical assumption of data exchangeability, with a growing literature extending this idea beyond that setting. We present `nonconform`, a Python package for applying conformal anomaly detection within existing machine-learning workflows, and use it as the basis for an implementation-grounded introduction to the field. The package integrates with `scikit-learn`, `PyOD`, and custom anomaly detectors, and provides a unified interface for calibration, p-value generation, and false discovery rate control. It supports several conformalization strategies, ranging from simple split-conformal calibration to more data-efficient and shift-aware extensions. Through a progression from foundational concepts to advanced conformalization strategies, complemented by code examples, the paper connects the statistical ideas behind conformal anomaly detection to their practical use in `nonconform`. Empirical results demonstrate that the implemented methods enable statistically principled anomaly detection. Together, the package and exposition aim to make core conformal anomaly detection workflows more accessible and reproducible in experimental and production-oriented settings.

Keywords: Anomaly Detection, Conformal Inference, Software Implementation

1. Introduction

Anomaly detection is fundamentally an exercise in drawing a line between the "expected", which *conforms* to an anticipated state of normality, and the "exceptional". In practice, this task is often delegated to scoring functions such as the isolation path length in an *Isolation Forest* (Liu et al., 2008) or the distance to a hyperplane in a *One-Class Support Vector Machine* (Schölkopf et al., 2001). However, while these models are highly effective at ranking observations by their degree of anomaly, the resulting scores are typically heuristic and lack an intrinsic statistical interpretation. A practitioner who wants to move from an anomaly score to a decision must still choose a threshold — often using intuition, visual inspection, or more elaborate procedures that remain only loosely connected to the actual reliability of the resulting decisions. In other words, such ad hoc approaches offer no formal control of false alarms, leaving practitioners unable to answer a basic operational question: "*If I use this threshold, how often will I flag normal observations by mistake?*"

Without a calibrated link between threshold choice and error rates, deploying anomaly detection systems becomes difficult in settings where decisions must be operationally reliable and statistically interpretable. **Conformal Anomaly Detection (CAD)** (Laxhammar and Falkman, 2015; Bates et al., 2023) addresses this limitation by calibrating raw anomaly scores into valid p-values that quantify how unusual a new observation appears relative to a reference sample of normal data. In this way, CAD turns anomaly detection from a purely ranking-based task into a statistically interpretable testing problem, enabling decisions to be based on established statistical procedures with explicit control over false alarms.

1.1. Motivation: Statistical Error Control

In many real-world anomaly detection tasks, the cost of a false positive is a direct operational risk. Beyond the immediate cost of investigation, a high rate of false positives can trigger *alert fatigue* and erode trust in the underlying system. For an anomaly detection system to be operationally reliable, a practitioner should be able to specify a tolerated nominal error rate $\alpha \in (0, 1)$ (e.g. $\alpha = 0.05$, or 5%) and expect the resulting decision rule to respect this level in the long run. The appropriate choice of α depends on the application, since tolerance for false alarms is tied to the consequences of acting on them.

Standard anomaly detection methods do not usually produce outputs whose thresholds admit such a direct interpretation. As a result, the relationship between a chosen score threshold and the resulting false alarm rate is typically unclear until the system is deployed.

Under the standard conformal framework, this interpretability takes a concrete form. If a test observation is flagged whenever its conformal p-value is at most α, then under the null hypothesis $\mathcal{H}_0$ that the observation is an inlier, and under the assumptions required for conformal validity,

$$\mathbb{P}(\text{False Positive}) \leq \alpha.$$

This guarantee is largely independent of the particular anomaly scoring model used and therefore provides a robust statistical foundation for decision-making.

1.2. The Challenge of Scale: Multiple Testing

Controlling false alarms for a single observation is an important first step, but practical anomaly detection systems rarely evaluate observations in isolation. In monitoring, screening, or streaming settings, models may assess observations at scale, either in batches or sequentially. Once each observation is associated with its own statistical test, anomaly detection becomes a multiple-testing problem (Shaffer, 1995). A decision rule calibrated at $\alpha = 0.05$ per instance implies that, on average, at most one in twenty normal observations will be flagged purely by chance. In high-throughput settings, these individual risks accumulate, so that even when each decision is controlled in isolation, the overall number of false alarms can become substantial.

A natural reaction is to simply lower the tolerated error rate, making each individual test more conservative. While this reduces false alarms, it also suppresses genuine anomalies. More fundamentally, controlling errors *per observation* does not tell a practitioner much about the quality of the *set* of observations that is ultimately flagged. Operationally, the more relevant question is often: "*Among all alerts, what fraction turns out to be false alarms?*"

Informally, this is the quantity captured by the **False Discovery Rate (FDR)** (Benjamini and Hochberg, 1995), which can be thought of as

$$\text{FDR} \approx \frac{\text{False Alarms}}{\text{Total Alarms}} \quad \text{or, operationally,} \quad \frac{\text{Wasted Investigation Effort}}{\text{Total Investigation Effort}}.$$

Rather than constraining how often any one normal observation is mistakenly flagged, FDR controls the expected proportion of false positives among all flagged observations. In practice, this makes FDR control not merely a statistical safeguard but a resource-management principle: it allows some false alarms while ensuring that the selected set of anomalies remains informative overall.

Standard procedures for FDR control, most notably the **Benjamini–Hochberg Procedure (BH)** (Benjamini and Hochberg, 1995), operate on a collection of p-values and determine which observations to flag while controlling the FDR at a chosen target level. Their practical effect is to filter a potentially large set of candidate anomalies down to a smaller and more reliable discovery set. Observations that would pass a naive per-instance threshold may no longer survive this multiple testing correction, while sufficiently extreme cases remain selected. This selective behaviour is what makes FDR control well-suited for explorative tasks like anomaly detection at scale. While BH is defined for batch settings, analogous procedures also exist for sequential testing and other testing frameworks, e.g. Vovk et al. (2003), Vovk and Wang (2021) or Javanmard and Montanari (2018).

1.3. From Scores to Decisions

The multiple-testing perspective described above relies on the availability of valid p-values. Standard anomaly detectors do not produce them directly. CAD bridges this gap by comparing the anomaly score of a new observation with scores computed on reference data known to be normal. If the new score is more extreme than most reference scores, this provides evidence that the observation does not conform to the expected pattern. In that sense, CAD turns the raw output of a scoring model into a valid statistical p-value.

In the standard setting, the validity of this p-value construction rests on the statistical assumption of **Exchangeability** (Vovk et al., 2005). Informally, this means that the calibration observations and the new test observation can be regarded as being generated symmetrically under the null, so that none occupies a privileged position. A key consequence is that the statistical guarantee attaches to the calibration procedure rather than to the anomaly detector itself. CAD is therefore entirely agnostic to the underlying model.

That said, exchangeability is not always a realistic assumption in practice. Temporal data naturally exhibit serial dependence, and spatial data often show systematic spatial dependence. In both cases, the order or location of observations carries information, which violates the symmetry required by the standard conformal framework. This should not be understood as a limitation of conformal methods in general, but rather of their classical formulation. A growing body of research has developed extensions that relax the exchangeability assumption and recover approximate or exact validity guarantees in settings where the standard approach no longer applies directly. Where such extensions are applicable to anomaly detection, they are implemented in `nonconform` as the library evolves. The present work focuses on more classical conformal methods, with the aim of developing the foundational concepts, assumptions, and validity arguments that underpin the field.

1.4. Scope and Outline

As outlined, CAD serves two closely related purposes. It gives anomaly scores a statistical interpretation as calibrated p-values and consequently enables principled error control with statistical guarantees in downstream tasks. In this context, the paper develops different perspectives on workflows and outcomes while introducing `nonconform`.

Section 2 presents the theoretical and methodological foundations of CAD, beginning with the standard framework and then extending to more advanced strategies, including data-efficient resampling-based variants and methods designed for settings beyond the classical exchangeability assumption. Section 3 broadens the perspective by introducing conformal martingales as an advanced sequential use of conformal p-values, with emphasis on their interpretation as tools for change-point detection. Section 4 provides a technical introduction to `nonconform`, its design principles, and its integration with the broader Python ecosystem. Section 5 concludes with an outlook for the future direction of this project.

Demonstration. Each section is accompanied by empirical results to build intuition on the observed behaviour and potential failure modes. As practical demonstrations rather than a comprehensive empirical evaluation, all experiments use scikit-learn's Isolation Forest (Liu et al., 2008) on the UCI Statlog (Shuttle) dataset (Feng et al., 1993).[1]

2. Conformal Anomaly Detection

Section 1 motivated CAD as a way to turn heuristic anomaly scores into statistically interpretable decisions. We now formalize that idea in the standard setting. The basic construction is simple: a new observation is assigned an anomaly score; this score is compared with scores computed on reference calibration data, disjoint from the original training data, that is known to be normal. The test score's relative rank among the calibration scores is the test point's p-value (see Section 2.2). Under the exchangeability assumption introduced above, these p-values are valid in finite samples, regardless of the particular anomaly detector used to generate the scores.

2.1. Problem Setup and Notation

We begin by fixing notation for the standard setting. Let $X_1, \ldots, X_n \in \mathcal{X}$ denote observations representing normal behaviour, and let $X_{n+1}, \ldots, X_{n+m} \in \mathcal{X}$ denote new observations to be assessed. An anomaly detector is represented by a scoring function

$$s : \mathcal{X} \to \mathbb{R},$$

where larger values indicate greater deviation from the reference pattern. The role of CAD is to calibrate these scores so that they admit a statistical interpretation.

To this end, the scores of the test observations are compared with the scores of the reference sample, yielding a p-value for each X_{n+j}, $j = 1, \ldots, m$. These p-values quantify how extreme the test scores are relative to the normal reference observations and can then be used either directly for thresholding or as input to multiple-testing procedures. The case $m = 1$ corresponds to testing a single new observation, as described in Section 1.1.

1. All results are reproducible using the code available at github.com/OliverHennhoefer/nonconform-paper.

2.2. Inductive Conformal Anomaly Detection

We now make this construction concrete in the standard inductive, or split-conformal, setting: The reference sample $X_1, \ldots, X_n$ is partitioned into a training set $\mathcal{D}_{\text{train}}$ and a calibration set $\mathcal{D}_{\text{cal}}$, with

$$\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{cal}} = \varnothing, \qquad \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{cal}} = \{X_1, \ldots, X_n\}.$$

The scoring function s is fitted using only $\mathcal{D}_{\text{train}}$. It is then evaluated on the calibration observations to obtain calibration scores

$$S_i := s(X_i), \qquad X_i \in \mathcal{D}_{\text{cal}}.$$

Consider first a single new observation X_{n+1}, with corresponding score

$$S_{n+1} := s(X_{n+1}).$$

Under the convention that larger scores indicate greater deviation from normality, the split-conformal p-value is defined as

$$\hat{p}(X_{n+1}) = \frac{\sum_{X_i \in \mathcal{D}_{\text{cal}}} \mathbf{1}\{S_i \geq S_{n+1}\} + 1}{|\mathcal{D}_{\text{cal}}| + 1}.$$

This quantity is the fraction of calibration scores that are at least as large as the test score, including a finite-sample correction. Small values of $\hat{p}(X_{n+1})$ indicate that the test score is unusually extreme relative to the calibration set and therefore provides evidence against the null hypothesis that X_{n+1} follows the same distribution as the normal reference observations.

Validity. The importance of this construction is that $\hat{p}(X_{n+1})$ is a valid p-value under the standard conformal assumptions. More precisely, if the calibration observations and the test point are exchangeable, and if the score function s is fitted using only $\mathcal{D}_{\text{train}}$, then

$$\mathbb{P}(\hat{p}(X_{n+1}) \leq t) \leq t, \qquad t \in [0, 1].$$

Thus, under the null, the conformal p-value is super-uniform, i.e. *at worst* biased towards conservativeness. In particular, for any significance level $\alpha \in (0, 1)$,

$$\mathbb{P}(\hat{p}(X_{n+1}) \leq \alpha) \leq \alpha,$$

so the decision rule that flags X_{n+1} whenever $\hat{p}(X_{n+1}) \leq \alpha$ controls the type-I error in finite samples. This is the formal version of the operational guarantee discussed in the introduction: once anomaly scores have been conformalized, thresholding at level α yields a false-alarm probability of at most α under the null hypothesis.

This guarantee relies on the sample split. Once s has been fitted on $\mathcal{D}_{\text{train}}$, the calibration and test scores are computed using the same fixed scoring rule and are therefore comparable under the null hypothesis. The $+1$ correction in numerator and denominator ensures exact finite-sample validity and prevents p-values equal to zero. Since conformal p-values lie on a discrete grid, they are generally not exactly uniform, but rather super-uniform.

Accordingly, the standard inductive conformal decision rule flags X_{n+1} whenever

$$\hat{p}(X_{n+1}) \leq \alpha.$$

The discriminative performance of s is not part of the validity argument, but it directly affects statistical power and determines whether and how many discoveries can be made.

2.3. From Individual Decisions to Multiple Testing

The single-observation construction extends directly to the batch setting. For test observations $X_{n+1}, \ldots, X_{n+m}$, we obtain conformal p-values

$$\hat{p}_j := \hat{p}(X_{n+j}), \qquad j = 1, \ldots, m.$$

Each of these p-values is marginally valid under its corresponding null hypothesis. At the same time, because they are all computed using the same random calibration sample, they exhibit dependence stemming from the common calibration set inducing a shared source of randomness across all test points.

When many observations are screened simultaneously, marginal validity of the individual $\hat{p}_j$ values is no longer sufficient if one seeks a batch-level error guarantee. In that setting, thresholding the $\hat{p}_j$ values separately at an unadjusted level is generally inadequate, and the problem is more naturally formulated as one of multiple testing. One therefore applies a multiple-testing procedure to $(\hat{p}_1, \ldots, \hat{p}_m)$ to obtain binary decisions $(\delta_1, \ldots, \delta_m) \in \{0, 1\}^m$, where $\delta_j = 1$ indicates that X_{n+j} is flagged as anomalous. The mentioned dependence among the conformal p-values then becomes important, because it affects which multiple-testing procedures can be justified under the results of Bates et al. (2023).

A standard choice is the BH procedure, which targets FDR control at a prescribed level $\alpha \in (0, 1)$, even under this dependence. Let

$$\hat{p}_{(1)} \leq \hat{p}_{(2)} \leq \cdots \leq \hat{p}_{(m)}$$

denote the ordered conformal p-values, and define

$$k^* := \max \left\{ k \in \{1, \ldots, m\} : \hat{p}_{(k)} \leq \frac{k}{m} \alpha \right\},$$

with $k^* = 0$ if the set is empty. If $k^* > 0$, BH flags all observations satisfying

$$\hat{p}_j \leq \hat{p}_{(k^*)}.$$

Whether this step enjoys rigorous finite-sample validity depends on the dependence structure of the conformal p-values, a point returned to in Section 2.7.

This separates the CAD pipeline into two conceptually distinct steps: first, conformal calibration turns raw anomaly scores into valid per-observation p-values. Second, a multiple-testing procedure turns those p-values into a coherent set of anomaly decisions at the desired error level. In this way, the operational perspective from Section 1 carries over to the formal setting: conformalization provides calibrated evidence for each individual observation, while multiple testing determines how that evidence is aggregated when anomaly detection is performed at scale.

When observations arrive sequentially, conformal p-values can be tested with online testing variants of the BH procedure. However, in practice these methods are more conservative, since decisions must be made without access to future p-values, see Javanmard and Montanari (2018).

Implementation: Inductive Conformal Anomaly Detection

```python
from oddball import Dataset, load  # Optional dependency
from pyod.models.iforest import IForest

from nonconform import ConformalDetector, Split
from nonconform.metrics import false_discovery_rate, statistical_power

x_train, x_test, y_test = load(Dataset.SHUTTLE, setup=True, seed=42)

detector = ConformalDetector(
    detector=IForest(),
    strategy=Split(1_000),  # Splits calibration set from training set
    seed=42
)

detector.fit(x_train)  # Trains and calibrates

decisions = detector.select(x_test, alpha=0.2)  # Applies BH Procedure

print(f"Empirical FDR: {false_discovery_rate(y_test, decisions)}")
print(f"Statistical Power: {statistical_power(y_test, decisions)}")
```

```
Empirical FDR: 0.18
Statistical Power:  0.99
```

Detached Calibration. While the standard conformal pipeline fits the model and calibrates it in a single coordinated sequence, this is not always feasible in production environments where base models are computationally expensive or already deployed. To accommodate this, `nonconform` supports a *detached* calibration workflow. Practitioners can wrap an already-fitted model and bypass the training phase entirely, applying `.calibrate(X_calib)` directly to a held-out inlier set, allowing for retroactive detector conformalization.

2.4. Resampling-based Extensions

The split-conformal construction offers the clearest route to finite-sample validity, but it requires a calibration set that is disjoint from the data used to fit the anomaly detector. In small samples, this can be costly as fewer observations remain for fitting, and the resulting conformal p-values become coarse because they lie on the grid

$$\left\{\frac{1}{|\mathcal{D}_{\text{cal}}|+1}, \frac{2}{|\mathcal{D}_{\text{cal}}|+1}, \ldots, 1\right\}.$$

In CAD, this discreteness matters directly, since a small calibration set limits the smallest attainable p-value and can make downstream testing procedures less powerful.

This motivates resampling-based extensions (Hennhöfer and Preisach, 2024) that reuse the available inlier data more efficiently while preserving the symmetry structure required for conformal validity approximately (Vovk, 2015).

Two important families are cross-conformal and bootstrap-based methods. The former partitions the data into K folds, fits the detector repeatedly on $K-1$ folds, and uses the held-out fold for scoring, so that every observation receives an out-of-sample score while more data are used for fitting than in split conformal.

Implementation: Cross-Conformal Calibration (CV/CV+)

```
detector = ConformalDetector(
    detector=IForest(),
    strategy=CrossValidation(k=10, mode="plus")
)
```

Leave-one-out (also called *Jackknife*) is the special case with $K = n$.

Implementation: Leave-One-Out-Conformal Calibration

```
detector = ConformalDetector(
    detector=IForest(),
    strategy=CrossValidation.jackknife(mode="plus")
)
```

Bootstrap-based variants instead replace deterministic folds by repeated resampling with replacement and aggregate out-of-bag scores across bootstrap models. In the implemented "+" modes, fold- or bootstrap-specific models are retained and their test scores are aggregated, while calibration uses out-of-fold or out-of-bag scores.

Implementation: Bootstrap-Conformal Calibration (JaB/JaB+)

```
detector = ConformalDetector(
    detector=IForest(),
    strategy=JackknifeBootstrap(n_bootstraps=100, mode="plus")
)
```

In general, these methods trade additional computation for better data efficiency and higher p-value resolution than split conformal. Their validity guarantees are method-specific and are not equivalent to the exact split-conformal guarantee. The non-"+" variants additionally rely on stability-type approximations.

Figure 1 compares recall and FDR across resampling-based strategies, which reuse the same N reference observations for model fitting and calibration, and the split procedure, which divides the available data between training and calibration. It highlights how the effective calibration-set size determines the minimum attainable p-value and statistical power.

Probabilistic Approximation. With `Probabilistic()`, the package provides an approximate estimation that replaces empirical ranks with a kernel density estimates. This yields non-discrete and arbitrarily small p-values, avoiding the empirical resolution floor. However, the variant is strictly not conformal: finite-sample validity and FDR control are not guaranteed, and any asymptotic justification requires additional regularity assumptions.

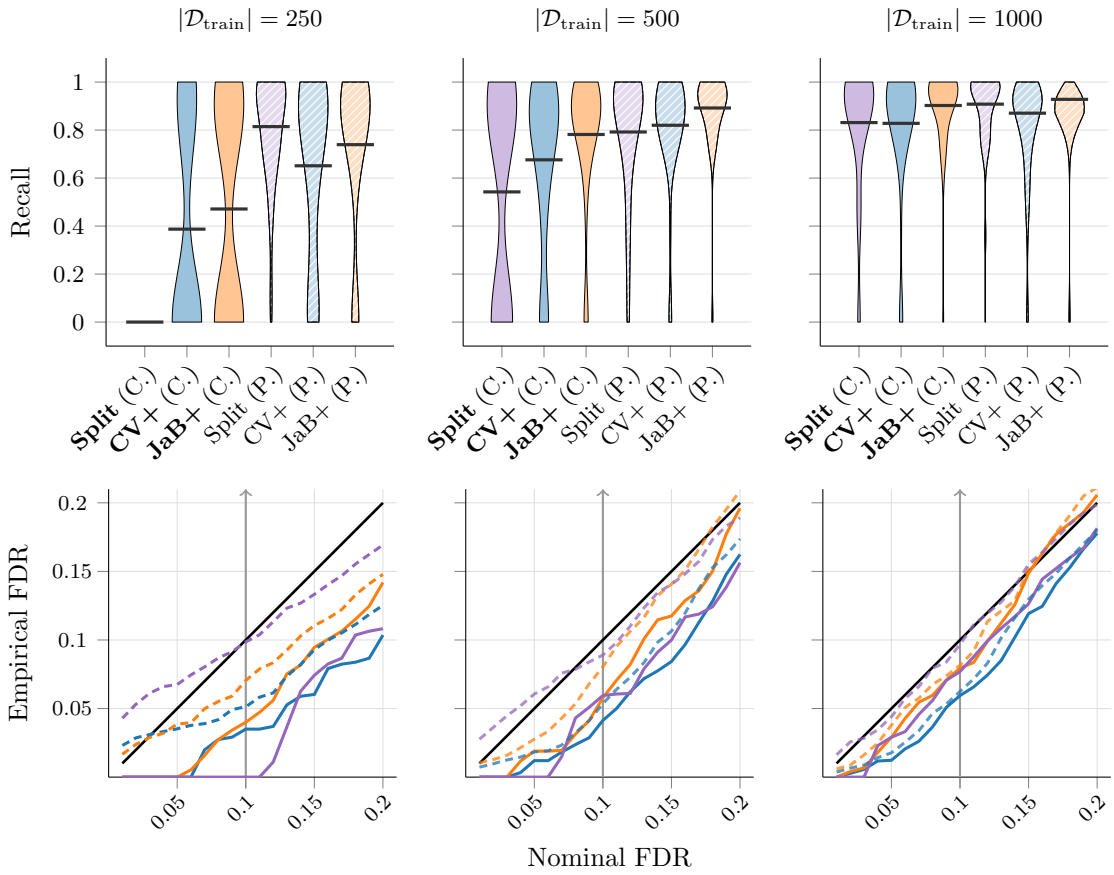

Figure 1: **Recall and FDR depend on the calibration-set size.** Top: distribution of recall at nominal FDR level $\alpha = 0.1$ for training-set sizes $|\mathcal{D}_{\text{train}}| \in \{250, 500, 1000\}$. Bottom: empirical FDR across corresponding nominal FDR levels. Depicted are the standard conformal and the probabilistic (P.) approach. The **Split** variant is calibrated on $\mathcal{D}_{\text{train}}/2$. Results are averaged over 50 randomized trials. **JaB+** uses `n_bootstraps`=100, and **CV+** uses `k`=10.

2.5. Marginal and Calibration-Conditional Error Control

A useful distinction, already implicit in the discussion above, is that between *marginal* and *calibration-conditional* error control (Bates et al., 2023). Marginal error control averages over both the random calibration set and the random test sample. In this sense, it guarantees that conformal p-values behave correctly across repeated applications of the full data-generating process. This is the standard form of validity attached to classical conformal methods and the one underlying the split-conformal guarantees introduced earlier.

By contrast, calibration-conditional control is stronger: it requires validity to hold after conditioning on the realized calibration set, rather than only on average over repeated calibration samples. This distinction is especially relevant in applications where one fixed reference dataset is collected once and then reused to screen many future observations.

In such settings, marginal validity may be formally correct yet still offer limited reassurance to the practitioner who must rely on one particular calibration set in deployment.

Implementation: Marginal Calibration

```
detector = ConformalDetector(
    detector=IForest(),
    strategy=Split(1_000),
    estimation=Empirical()  # Controls marginally (default)
)
```

Operationally, the calibration-conditional approach starts from the same empirical conformal construction but then applies an additional simultaneous correction to the resulting p-values. The parameter `delta` specifies the tolerated failure probability of this stronger guarantee: for example, `delta=0.1` means that, with probability at least 0.9 over the draw of the calibration set, the corrected p-values are valid for that realized calibration sample. The argument `method="simes"` specifies the particular correction used to build this uniform bound, here a Simes-type adjustment following Bates et al. (2023). Smaller values of `delta` give stronger assurance but typically produce more conservative p-values.

Implementation: Calibration-conditional Calibration

```
detector = ConformalDetector(
    detector=IForest(),
    strategy=Split(1_000),
    estimation=ConditionalEmpirical(method="simes", delta=0.1)
)
```

The trade-off is familiar: stronger conditioning generally requires more conservative procedures. Calibration-conditional guarantees therefore tend to come at some cost in power, since they adjust the empirical p-values to remain reliable for the realized calibration sample rather than only on average across repeated samples. For CAD, this distinction becomes especially important once one moves beyond single-observation decisions and begins to study repeated or large-scale use of the same calibrated system. Figure 2 compares marginal and calibration-conditional error control and its impact on testing power.

2.6. Assumptions for Standard Conformal Validity

In the standard, unweighted setting, conformal validity rests on exchangeability between the calibration observations and the test observation under the null hypothesis. In most applications, this is ensured by assuming that these observations are sampled i.i.d. from a common inlier distribution, although exchangeability is formally slightly weaker than full independence and identical distribution. What matters is that, under the null, the calibration scores and the test score are generated symmetrically, so that none occupies a privileged role in the conformal comparison.

A second requirement concerns the separation between model fitting and calibration. The anomaly detector may otherwise be arbitrary, but after it is trained on $\mathcal{D}_{\text{train}}$, including

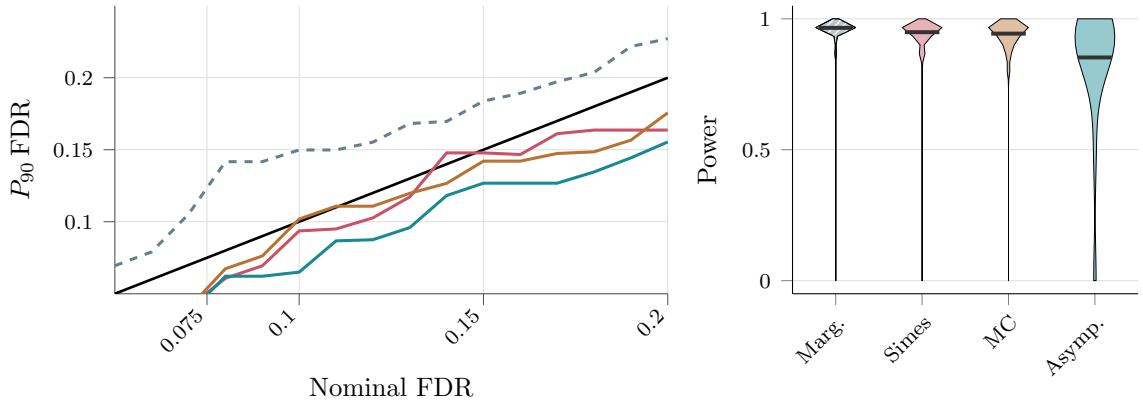

Figure 2: **Conditional-calibration corrections reduce upper-tail empirical FDR at a cost in power.** Left: 90th-percentile empirical FDR across 20 randomized trials as a function of the nominal FDR target, illustrating tail behavior under conditional calibration with `delta=0.1`, with marginal control as reference. Right: distribution of recall across trials at nominal level $\alpha = 0.1$. Methods correspond to JaB+ with marginal control and calibration-conditional control via `method="simes"`, `"mc"`, and `"asymptotic"`.

any independent algorithmic randomness, the resulting scoring rule must remain fixed while calibration and test observations are scored; it must not adapt to either set. This is why validity is largely model agnostic. Nevertheless, the detector remains crucial for statistical power: a weak or unstable scoring rule may still yield valid p-values, but they can be uninformative and lead to few true discoveries.

Under these conditions, the resulting conformal p-values are marginally valid. That is, their type-I error guarantee holds after averaging over both the random calibration sample and the random test point. As discussed above, stronger calibration-conditional guarantees are possible, but they require additional adjustment and typically come with a corresponding loss in power. The classical exchangeability framework is therefore the natural starting point for CAD. The next section considers how this picture changes once exchangeability is no longer tenable (due to covariate/feature shift) and the calibration and test distributions must be related through a more general weighting scheme.

2.7. Weighted Conformal Anomaly Detection

The standard conformal construction relies on exchangeability between calibration and test observations. This assumption can be too restrictive when the notion of normality remains unchanged, but the distribution of inlier covariates shifts between calibration and deployment. In this case, the calibration sample is no longer representative of the test environment, and conformal p-values may lose their validity, see Figure 3.

Weighted CAD addresses this covariate-shift setting by replacing the unweighted empirical comparison with a weighted one. Let calibration inliers be sampled from a source distribution P, while test inliers follow a target distribution Q. The shift is assumed to be

captured by a covariate-dependent likelihood ratio (i.e. a *Radon–Nikodým derivative*)

$$w(x) = \frac{\mathrm{d}Q_X}{\mathrm{d}P_X}(x),$$

or a suitable estimate thereof (Shimodaira, 2000), for example via probabilistic Random Forest classification. A support condition is required: Q_X must be absolutely continuous with respect to P_X with $\mathrm{supp}(Q_X) \subseteq \mathrm{supp}(P_X)$, so that covariate values occurring at test time have positive probability under the calibration distribution. Regions outside calibration support cannot be empirically corrected by reweighting.

Using the weights $w(X_i)$, calibration scores that are more representative of the target environment contribute more strongly to the conformal reference distribution, while less representative scores contribute less. The resulting weighted conformal p-values therefore compare a test observation to a target-adjusted inlier reference distribution rather than to the source calibration sample alone, so rank counts are replaced by weighted rank counts.

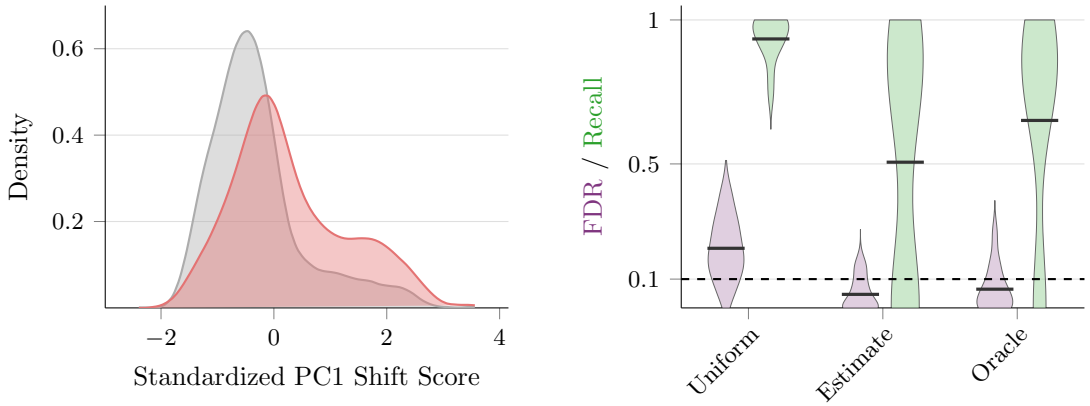

Figure 3: **Importance weighting restores empirical marginal FDR control in this covariate shift scenario.**[3] Left: density of the standardized first principal-component shift score demonstrating covariate shift between the calibration and test distribution. Right: distribution of empirical marginal FDR and recall at nominal level $\alpha = 0.1$ for uniform weighting, estimated weighting (logistic), and oracle weighting with weighted conformal selection.

2.8. Weighted Conformal Validity and Multiple Testing

Weighted conformal validity replaces ordinary exchangeability with weighted exchangeability induced by the covariate-shift assumption. Calibration and test observations may have different covariate distributions, but their difference must be captured by the density ratio $w(x)$, while the relevant conditional notion of normality given X remains stable. Thus, weighted conformal inference addresses covariate shift with overlap, not arbitrary concept shifts between disconnected distributions.

3. Covariate shift induced by rejection-sampling along 1st inlier principal component Jin and Candès (2026).

Under these assumptions, and with the correct weights, weighted conformal p-values are marginally valid for a single test under the target distribution: thresholding one such p-value at level α controls the type-I error at level α. If w is unknown, it is commonly replaced by a density-ratio estimate learned from calibration and target covariates, but exact validity is then no longer automatic and depends on the quality of this estimate. Practically, this is why sufficient overlap is important. If the target distribution concentrates in regions poorly represented by calibration data, a few observations receive large weights and the calibration may become unstable or ineffective for discovery (Hennhöfer and Preisach, 2026).

This single-test validity does not automatically imply finite-sample multiple-testing guarantees. In the unweighted setting, the shared calibration sample induces dependence across test p-values, but this dependence is structured enough to support e.g. standard BH. Under weighting, the resulting p-values need not satisfy positive regression dependence on a subset (PRDS), the positive-dependence property used in the usual BH argument, because the covariate-dependent weights also enter the shared calibration comparison.

Consequently, BH may remain useful in practice, and FDR control may hold asymptotically, but exact finite-sample FDR control generally requires procedures tailored to the weighted setting, such as Weighted Conformalized Selection (WCS) (Jin and Candès, 2026). Weighted CAD should therefore be viewed not only as an alternative calibration formula, but also as a shift-aware extension that changes the downstream multiple-testing problem.

Implementation: Weighted Conformal Anomaly Detection

```
detector = ConformalDetector(
    detector=IForest(),
    strategy=Split(1_000),
    weight_estimator=forest_weight_estimator()
).fit(x_train)

# Dispatches to WCS if a 'weight_estimator' (above) is defined
decisions = detector.select(
    x_test,
    alpha=0.1
)
```

3. Conformal Martingales and Change-Point Detection

The CAD framework developed so far is primarily pointwise: each new observation receives a conformal p-value and is assessed either individually or as part of a multiple-testing problem. In sequential settings, however, the central question is often different. Rather than asking whether a single observation is anomalous, one may want to detect whether the data-generating mechanism itself has changed over time. This is where *conformal martingales* become natural. Within anomaly detection, they are best viewed not as a replacement for batch outlier screening, but as a process-level method for temporal anomaly detection, with change-point detection as the main use case.

The basic idea is to monitor a stream of conformal p-values over time. Under the null hypothesis of exchangeability, these p-values are valid; in the ideal randomized online conformal construction, they behave like independent draws from the uniform distribution on $[0, 1]$. A conformal martingale transforms the sequence $p_1, p_2, \ldots$ into a nonnegative evidence process $M_0, M_1, M_2, \ldots$ that is fair, in a betting sense, under the null hypothesis. Consequently, if M_n grows to a large value, this indicates accumulated evidence against the assumption that the stream is still behaving as before. A common way to turn this evidence process into an alarm is to use a *Ville threshold*: one raises an alarm once M_n crosses a fixed level λ. For a nonnegative martingale started at one, Ville's inequality gives the anytime bound

$$\mathbb{P}\left(\sup_{t\geq 0} M_t \geq \lambda\right) \leq \frac{1}{\lambda},$$

so the probability of crossing the threshold under the null is at most $1/\lambda$. For example, $\lambda = 100$ corresponds to an anytime false-alarm bound of at most 1% for a monitored stream.

Implementation: Exchangeability Martingale

```python
from sklearn.ensemble import IsolationForest

from nonconform import ConformalDetector, Split
from nonconform.martingales import AlarmConfig, PowerMartingale

detector = ConformalDetector(
    detector=IsolationForest(random_state=42),
    strategy=Split(n_calib=0.2)
).fit(x_train)

martingale = PowerMartingale(
    epsilon=0.5,
    alarm_config=AlarmConfig(ville_threshold=100)
)

for x_t in data_stream:
    p_t = detector.compute_p_value(x_t)
    state = martingale.update(p_t)

    if "ville" in state.triggered_alarms:
        print("Alert")
```

A classical example is the power martingale (Vovk et al., 2003)

$$M_n^{(\varepsilon)} = \prod_{i=1}^{n} \varepsilon p_i^{\varepsilon-1}, \qquad \varepsilon \in (0, 1],$$

which grows when unusually many small conformal p-values are observed. Since the performance of a fixed value of ε can depend strongly on the form of the departure from

exchangeability, one often works instead with mixtures over different betting parameters (Vovk et al., 2003). Adaptive plug-in variants estimate the betting strategy from previous p-values (Fedorova et al., 2012). In this way, conformal martingales provide an online mechanism for accumulating evidence against the null, rather than issuing single-step decisions.

This perspective connects naturally to change-point detection. An isolated small p-value may correspond only to a transient or local anomaly. By contrast, a sustained run of smaller p-values leads to systematic martingale growth and is therefore more indicative of a structural shift in the underlying process. Conformal martingales are useful when the goal is to determine whether a previously calibrated detector or predictive model is no longer operating under the assumptions on which its validity was based. They therefore provide a statistically principled bridge between CAD and online change-point detection.

The current `nonconform` implementation follows this interpretation directly: it operates on sequential conformal p-values rather than on raw anomaly scores, and provides power, simple-mixture, and simple-jumper martingales. In addition to the cumulative martingale evidence process, the implementation supports Ville thresholding of the standard martingale and a restart-mixture e-process, which improves sensitivity to later changes while preserving anytime false-alarm control when the input sequence is conditionally valid. The package deliberately keeps downstream decision logic outside the martingale classes: a martingale alarm signals accumulated evidence of distributional change, while the response—such as retraining, or recalibration—remains a separate design choice (Vovk et al., 2021).

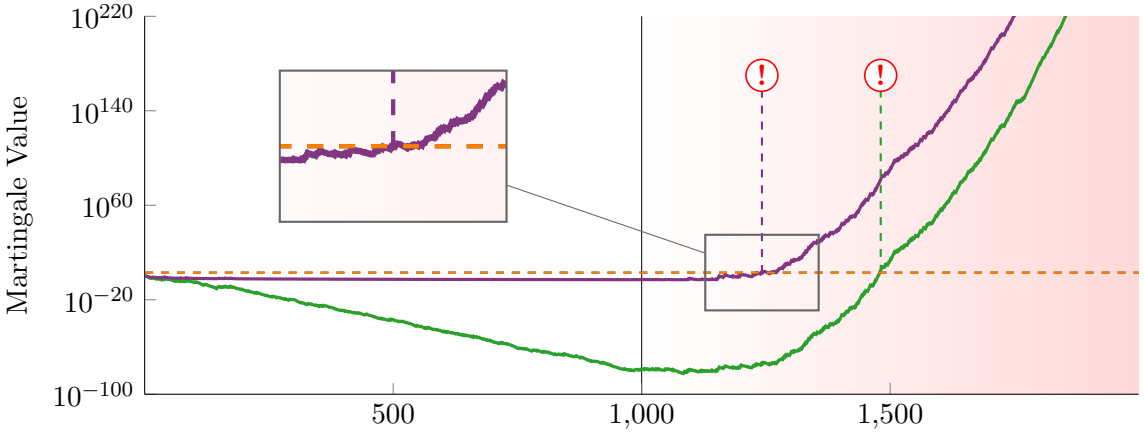

Figure 4: **Exchangeability martingales detect distributional changes in data streams.** A stream of 2,000 observations is processed via Isolation Forest, producing p-values that are approximately uniform before the change point and become non-uniform as the anomaly rate increases linearly from 0% to 100% after $t = 1000$. The **Standard Martingale** and **Restart-mixture Martingale** evaluate p-values sequentially and trigger alarms when crossing the threshold $1/\alpha$, with $\alpha = 0.001$. For each Ville-valid statistic, this controls the anytime false-alarm probability at no more than 0.1% under an exchangeability null. The stated Ville bound applies only when sequential p-values generate conditionally e-valid betting factors. The fixed-split paths shown here are illustrative processes.

Conformal martingales should be understood as a complementary extension of CAD. Standard conformal methods answer the question "*which observations look unusual relative to a reference sample?*" Conformal martingales instead address the sequential question "*is the stream, as a process, still behaving as if it were generated under the same exchangeable regime?*" This makes them particularly useful in monitoring and deployment settings, where the onset of a new regime may matter more than any single anomalous point observation.

4. The `nonconform` Package

CAD in practice requires more than a scoring model: it also needs calibrated score routing, valid p-value construction, and multiplicity-aware decision rules. While Python provides mature conformal prediction libraries, including `MAPIE` (Cordier et al., 2023), `crepes` (Boström, 2024), and `puncc` (Mendil et al., 2023), these tools are largely designed around regression and classification workflows rather than anomaly detection. The `nonconform` package was designed to address this gap:

`https://github.com/OliverHennhoefer/nonconform`[4]

4.1. Positioning and Scope

To the best of our knowledge, `nonconform` is currently the only Python package that treats CAD as a primary, package-level objective: anomaly detector adaptation, conformal p-value computation, error control procedures, covariate-shift-aware weighting, and online-capable exchangeability martingales in one coherent API.

4.2. Core API and Modular Architecture

The central entry point is the `ConformalDetector` meta-estimator, built in a scikit-learn-compatible style (Buitinck et al., 2013). Its core lifecycle is

$$\texttt{.fit()} \rightarrow \texttt{.select()},$$

with optional `.compute_p_values()` and `.score_samples()` for custom analysis workflows and `.calibrate()` for detached calibration of wrapped, already-fitted models.

4.2.1. Core Modules and Responsibility Boundaries

The architecture separates concerns across modules:

- `nonconform.adapters`: detector adaptation and score-direction normalization.
- `nonconform.resampling`: conformalization strategies.
- `nonconform.scoring`: p-value computation.
- `nonconform.weighting`: estimator-based importance weighting.

4. All descriptions in this paper are tied to the release version `v1.0.2`. Install via `pip install nonconform` or `uv add nonconform`, optional dependencies can be added depending on workflow specifics.

- `nonconform.fdr`: weighted FDR selection primitives.
- `nonconform.martingales`: sequential exchangeability-evidence processes.
- `nonconform.structures`: `AnomalyDetector` protocol and result container.

4.2.2. Conformalization Strategy

The package provides three strategy families in `nonconform.resampling`:

- `Split()`
- `CrossValidation()`
- `JackknifeBootstrap()`

For resampling methods, `mode` (`plus` vs. `single_model`) controls the model retention–validity tradeoff. `CrossValidation.jackknife()` implements leave-one-out resampling.

4.2.3. Estimation Strategy

On top of calibrated scores, the package offers three estimator classes:

- `Empirical()`
- `ConditionalEmpirical()`
- `Probabilistic()`[5]

These are exposed in `nonconform.scoring` and integrated into `ConformalDetector` through the `estimation` argument. Detached calibration is only implemented for `Split()`.

4.3. Compatibility & Interoperability

The polarity of processed anomaly scores is critical for compatibility over different detector implementations. Internally, `nonconform` standardizes to

$$\text{higher score} \rightarrow \text{more anomalous}.$$

The `score_polarity` interface supports explicit conventions, strict `"auto"` inference for recognized detector families, and implicit defaults when omitted.

PyOD. `nonconform` supports `PyOD` Zhao et al. (2019) detectors through adapter-based compatibility, allowing direct use of established one-class detectors (e.g., `IForest`, `LOF`).

Warning

Some `PyOD` detectors (e.g., `COPOD`, `ECOD`, `LOCI`, `SOS`, `COF`) are not compatible with conformal workflows, as their scores are not generated by a fixed function post-training. As a result, conformal p-values and FDR guarantees are not valid.

Scikit-learn. `nonconform` follows scikit-learn estimator conventions (`fit`, `get_params`, `set_params`), enabling straightforward integration into existing model-selection and pipeline code. In particular, `ConformalDetector` supports standard estimator semantics while adding conformal calibration and decision control through `select()`.

5. `Probabilistic()` uses [weighted] KDE rather than empirical ranks. It is not conformal and has no finite-sample guarantee; asymptotic validity requires additional regularity assumptions.

Custom. Custom detector implementations can be integrated via the `AnomalyDetector` protocol. This keeps the detector layer replaceable while preserving a fixed conformal interface for calibration, p-value computation, and downstream FDR-controlled selection.

4.4. Error Control Under Batch and Shift

For standard batch usage, the recommended one-step API is `ConformalDetector.select()`, which computes conformal p-values and applies BH-style control in the unweighted mode. This avoids brittle manual thresholding and keeps the decision pipeline statistically explicit.

For covariate shift, `nonconform` uses estimator-based weighting via `weight_estimator`, for example `logistic_weight_estimator()`. In weighted mode, `select()` dispatches to weighted conformalized selection via `weighted_false_discovery_control()`.

4.5. Sequential Monitoring

Beyond pointwise decisions, `nonconform` supports streaming evidence monitoring through `PowerMartingale`, `SimpleMixtureMartingale`, and `SimpleJumperMartingale`. Alarm behavior is configured through `AlarmConfig` thresholds (Ville, CUSUM, Shiryaev–Roberts) and exposed in `MartingaleState.triggered_alarms`. This design keeps sequential evidence processes separate from cross-hypothesis FDR selection logic.

4.6. Reliability, Reproducibility, and Documentation

The repository follows a layered validation design across unit, integration, and end-to-end tests, and CI workflows for multi-version testing coverage, strict documentation builds, and wheel/post-publish smoke checks. Documentation includes API references plus executable examples for `scikit-learn`, `PyOD`, custom detectors, weighted workflows, detached calibration, conditional calibration, and martingales in greater detail:

https://oliverhennhoefer.github.io/nonconform/

Overall, `nonconform` contributes a robust implementation of CAD by making statistical assumptions explicit, API boundaries modular, and evaluation workflows reproducible.

5. Conclusion

The `nonconform` package implements a statistical decision layer for anomaly detection. By separating scoring from calibration, selection, weighting, and sequential evidence accumulation, it makes the assumptions behind each decision explicit. The resulting workflow replaces heuristic thresholds with conformal p-values that support FDR-controlled discovery sets under the corresponding assumptions, or martingale-based evidence measures for shifts in data streams that, e.g., inform retraining. The central message is simple: reliable anomaly detection is not obtained from better scores and more complex models or workflows alone, but from decision rules whose statistical guarantees enable model-free and finite-sample valid quantification of uncertainty.

We plan to gradually expand `nonconform`'s use cases and invite open-source maintainers, CAD enthusiasts, and authors of relevant works to implement new methods, making applications and research in this field more accessible.

Acknowledgments

This work was conducted as part of a research project funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under grant number 01MV23020A.

References

Stephen Bates, Emmanuel Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. Testing for outliers with conformal p-values. *The Annals of Statistics*, 51(1):149–178, 2023. doi: 10.1214/22-AOS2244.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. doi: 10.1111/j.2517-6161.1995.tb02031.x.

Henrik Boström. Conformal prediction in python with crepes. In Simone Vantini, Matteo Fontana, Aldo Solari, Henrik Boström, and Lars Carlsson, editors, *Proceedings of the Thirteenth Symposium on Conformal and Probabilistic Prediction with Applications*, volume 230 of *Proceedings of Machine Learning Research*, pages 236–249. PMLR, 2024.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Thibault Cordier, Vincent Blot, Louis Lacombe, Thomas Morzadec, Arnaud Capitaine, and Nicolas Brunel. Flexible and Systematic Uncertainty Estimation with Conformal Prediction via the MAPIE library. In *Conformal and Probabilistic Prediction with Applications*, volume 204, pages 549–581, 2023.

Valentina Fedorova, Alex Gammerman, Ilia Nouretdinov, and Vladimir Vovk. Plug-in martingales for testing exchangeability on-line. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1639–1646. Omnipress, 2012.

C. Feng, A. Sutherland, S. King, S. Muggleton, and R. Henery. Statlog Project. UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5XS3B.

Oliver Hennhöfer and Christine Preisach. Leave-one-out-, bootstrap- and cross-conformal anomaly detectors. In *2024 IEEE International Conference on Knowledge Graph (ICKG)*, pages 110–119. IEEE, December 2024. doi: 10.1109/ickg63256.2024.00022.

Oliver Hennhöfer and Christine Preisach. Between resolution collapse and variance inflation: Weighted conformal anomaly detection in low-data regimes, March 2026.

Adel Javanmard and Andrea Montanari. Online rules for control of false discovery rate and false discovery exceedance. *The Annals of Statistics*, 46(2), April 2018. ISSN 0090-5364. doi: 10.1214/17-aos1559.

Ying Jin and Emmanuel J Candès. Model-free selective inference under covariate shift via weighted conformal p-values. *Biometrika*, 113(1), 02 2026. ISSN 1464-3510. doi: 10.1093/biomet/asaf066.

Rikard Laxhammar and Göran Falkman. Inductive conformal anomaly detection for sequential detection of anomalous sub-trajectories. *Annals of Mathematics and Artificial Intelligence*, 74(1–2):67–94, June 2015.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008. doi: 10.1109/ICDM.2008.17.

Mouhcine Mendil, Luca Mossina, and David Vigouroux. PUNCC: a python library for predictive uncertainty calibration and conformalization. In *Conformal and Probabilistic Prediction with Applications*, volume 204, pages 582–601. PMLR, 2023.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001. doi: 10.1162/089976601750264965.

Juliet Popper Shaffer. Multiple hypothesis testing. *Annual Review of Psychology*, 46(1): 561–584, 1995. doi: 10.1146/annurev.ps.46.020195.003021.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000. doi: 10.1016/S0378-3758(00)00115-4.

Vladimir Vovk. Cross-conformal predictors. *Annals of Mathematics and Artificial Intelligence*, 74(1–2):9–28, 2015. doi: 10.1007/s10472-013-9368-4.

Vladimir Vovk and Ruodu Wang. E-values: Calibration, combination and applications. *Ann. Stat.*, 49(3):1736–1754, June 2021.

Vladimir Vovk, Ilia Nouretdinov, and Alex Gammerman. Testing exchangeability on-line. In *Proceedings of the Twentieth International Conference on Machine Learning*, ICML'03, pages 768–775. AAAI Press, 2003. ISBN 1577351894.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, NY, 1 edition, 2005. doi: 10.1007/b106715.

Vladimir Vovk, Ivan Petej, Ilia Nouretdinov, Ernst Ahlberg, Lars Carlsson, and Alex Gammerman. Retrain or not retrain: conformal test martingales for change-point detection. In *Proceedings of the Tenth Symposium on Conformal and Probabilistic Prediction and Applications*, volume 152 of *Proceedings of Machine Learning Research*, pages 191–210. PMLR, 2021.

Yue Zhao, Zain Nasrullah, and Zheng Li. PyOD: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20(96):1–7, 2019.