# Extended K-Means Isolation Forest: A Hybrid Approach Combining Random Projections and Clustering for Anomaly Detection

Vlad-Ioan Bîrsan

Computer Science Department, University of Bucharest, Romania

## Abstract

Unsupervised anomaly detection is a fundamental problem in data mining, with applications spanning across many fields. While isolation-based methods like Isolation Forest (IF) are highly effective, their axis-parallel partitioning strategy can struggle with complex geometric distributions. In this paper, we conduct a comprehensive comparative study of efficiency and robustness across existing isolation forest variants, including Standard IF, Extended IF (EIF), Generalized IF (GIF), and K-Means IF (K-Means IF). Furthermore, we introduce two novel hybrid algorithms that extend the density-aware partitioning of K-Means IF: (1) Subspace K-Means IF, which projects data into random axis-parallel subspaces before clustering, and (2) Extended K-Means Isolation Forest (EKM-IF), which projects data onto random oblique hyperplanes prior to clustering. We evaluate these six methods on 13 benchmark datasets using ROC-AUC, PR-AUC, and training time metrics. Our results indicate that while the proposed hybrid approaches incur higher computational costs, they offer distinct advantages in capturing anomalies within complex manifolds where standard methods fail. Ultimately, we demonstrate that there is no single "winner" algorithm; rather, the optimal choice depends on the specific geometric structure and dimensionality of the dataset.

## 1 Introduction

Unsupervised anomaly detection is a fundamental problem in data mining, with applications spanning across many fields including fraud detection, fault diagnosis in industrial systems, and network intrusion detection. The diversity of these applications has led to a wide spectrum of algorithmic approaches, primarily categorized into distance-based methods, which assume anomalies are far from their nearest neighbors; density-based methods, which assume that anomalies occur in low-density regions; and model-based methods, which identify deviations from a learned normal profile.

Among these, isolation-based methods have been recognized as a highly effective category, particularly for high-dimensional data. One of the representative algorithms in this class is the Isolation Forest (IF) [4], which operates on the principle that anomalies are "few and different." IF constructs an ensemble of binary trees by recursively splitting the data using randomly selected features and split values. Because anomalies are numerically distant from the majority of data, they are susceptible to isolation closer to the root of the tree, resulting in shorter average path lengths compared to normal instances. This purely randomized approach offers significant computational efficiency. However, Standard IF suffers from "axis-parallel" bias; because it splits data only along coordinate axes, it struggles to capture correlations between features or detect anomalies in more complex distributions.

Several variants have been proposed to address these limitations. The Extended Isolation Forest (EIF) [1] generalizes the splitting condition by using random hyperplanes with arbitrary slopes rather than axis-parallel cuts. This allows the algorithm to capture more complex dependencies and eliminates the "ghost regions" often seen in Standard IF score maps. Building on this, the Generalized Isolation Forest (GIF) [3] further refines the splitting process to avoid "empty branches"—a common inefficiency in EIF where random cuts may separate no data—thereby improving computational speed without sacrificing detection accuracy. More recently, the K-Means Isolation Forest (K-Means IF) [2] introduced a density-aware partitioning strategy. Instead of purely random splits, K-Means IF utilizes K-Means clustering at each tree node to define branches, allowing the tree structure to adapt naturally to the local data density and producing a multi-branch tree rather than a strictly binary one.

In this paper, we conduct a comprehensive comparative study of efficiency and robustness across these existing isolation forest variants. Furthermore, we introduce two novel hybrid algorithms that extend the density-aware partitioning of K-Means IF to address its own limitations in high-dimensional spaces. The first approach is *Subspace K-Means IF*, which projects data into random axis-parallel subspaces before clustering, allowing the algorithm to focus on different feature subsets dynamically. The second approach is *Extended K-Means Isolation Forest (EKM-IF)*, which projects data onto random oblique hyperplanes prior to clustering, combining the geometric flexibility of EIF with the density adaptability of K-Means IF. While these hybrid methods incur higher training and inference costs due to the complexity of projections and clustering, they aim to offer distinct robustness in scenarios where standard methods fail. We emphasize that there is no single "winner" algorithm; rather, our objective is to present the specific trade-offs between various isolation forest algorithms in terms of computational cost and accuracy.

The remainder of this paper is organized as follows: Section 2 recalls the K-Means clustering algorithm and details the Standard IF, EIF, GIF, and K-Means IF algorithms.

Section 3 introduces our novel variations: Subspace K-Means IF and Extended K-Means IF. Section 4 presents the experiments and results, evaluating all six methods on 13 benchmark datasets, as well as on a sum of synthetic datasets generated from various distributions. Finally, Section 5 offers concluding remarks.

# 2 Technical description of the problem

In this section, we review the Standard Isolation Forest (IF) algorithm and some of its variants: Extended IF (EIF), Generalized IF (GIF), and K-Means IF.

## 2.1 Isolation Forest

The core premise of the Isolation Forest algorithm is that anomalous data points are more prone to isolation than normal points. Fundamentally, an isolation forest functions as an ensemble of isolation trees. Let $t$ denote the number of trees in the forest. Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$, where $X \subset \mathbb{R}^d$ and each point $x_i = \left[x_i^{(1)} x_i^{(2)} \ldots x_i^{(d)}\right]$, for $i = \overline{1, \ldots, n}$, is a $d$-dimensional vector, the construction of an isolation tree proceeds recursively. First, a component $q$ is randomly selected for projection. Next, the minimum and maximum values of the points along this dimension are identified, and a split value is uniformly sampled within this range. Specifically, a value $p$ is chosen randomly from the interval $\left[\min_{i=1,\ldots,n} x_i^{(q)}, \max_{i=1,\ldots,n} x_i^{(q)}\right]$. This split creates two branches: one containing points where the $k$-th component is less than or equal to $p$, formally $\{x_i \mid x_i^{(q)} \leq p\}$, and the other containing points where it is greater than $p$, $\{x_i \mid x_i^{(q)} > p\}$. This process repeats recursively on the new nodes until either a pre-defined maximum tree depth is reached or a node contains a single data point. This procedure is repeated $t$ times to generate the ensemble of trees.

To determine the anomaly score of a data point, we compute the depth of the leaf node it reaches. This is done by traversing the tree from the root, applying the split criteria at each node. Consequently, the anomaly score provided by a single tree is directly tied to the depth at which the point terminates. Since normal points typically reside deep in the tree while anomalies are isolated near the root, full tree construction is often unnecessary; instead, a maximum depth limit is imposed. In this scenario, leaf nodes may contain one or multiple points. For nodes with multiple points, the depth of a specific point is approximated. The outlier score is defined by the following formula:

$$s(x, n) = 2^{-\frac{E[h(x)]}{c(n)}} \tag{1}$$

where:

$$c(n) = 2 \cdot H(n-1) - \frac{2 \cdot (n-1)}{n} \tag{2}$$

Here, $c(n)$ represents the average path length $h(x)$ for a dataset of size $n$, and $H(n) \approx \ln(n) + \gamma$ is the $n$-th harmonic number, with $\gamma \approx 0.577215$ being the Euler-Mascheroni constant. Analyzing the formula, if $E[h(x)] \approx c(n)$, the average depth of a point equals the average path length, yielding a score of 0.5, which indicates no clear anomaly. If $E[h(x)]$ is significantly larger than $c(n)$, the score approaches 0, suggesting the point is likely normal. Conversely, if $E[h(x)]$ is much smaller than $c(n)$, the score approaches 1, indicating a likely anomaly. Thus, $s(x) \in [0, 1]$; values near 0 suggest inliers, while values near 1 suggest outliers. The exponent effectively compares the actual depth of the point to the expected average depth in a random tree of size $n$. A small ratio implies the point is isolated shallowly (anomalous), whereas a large ratio implies it is deep (normal).

**Algorithm 1:** Create Isolation Forest

**Data:** $X$ - dataset, $t$ - number of trees, $\psi$ - subsampling size
**Result:** iForest - a list of $ITree$

```
Function createIForest(X, t, ψ)
    iForest ← ∅
    l ← log2(ψ)
    for i = 1 : t do
        X' ← sample(X, ψ)
        iTree ← createITree(X', l, 0)
        iForest ← iForest ∪ {iTree}
    return iForest
```

Given that datasets can be large, trees can become computationally expensive. Liu et al. demonstrated in [4] that isolation forests perform efficiently with subsampling. Rather than using the entire dataset, trees can be constructed from smaller, random batches. Table 1 presents the optimal hyperparameters identified by Liu et al. in [4] regarding ensemble size, subsample size, and tree depth:

| Param | Meaning | Value |
|---|---|---|
| $\psi$ | Subsampling size | 256 |
| $t$ | Number of trees | 100 |
| $l$ | Maximum depth of a tree | $\log_2(\psi)$ |

Table 1: Selection of isolation forest hyper-parameters

In our experiments, we adopted the values specified in Table 1. A skeletal overview of the isolation forest algorithm is provided in 1, 2, 3, as introduced in [4].

## 2.2 Extended Isolation Forest

As illustrated in Figure 1, the Standard Isolation Forest generates "ghost artifacts"—regions assigned a low anomaly score despite containing little to no data. Furthermore, line patterns parallel to the coordinate axes are frequently observed, resulting in a data distribution approximation that is far from perfect. For instance, consider data at the edge of the distribution: in its immediate vicinity, along a diagonal direction, the anomaly score correctly increases. However, along directions parallel to

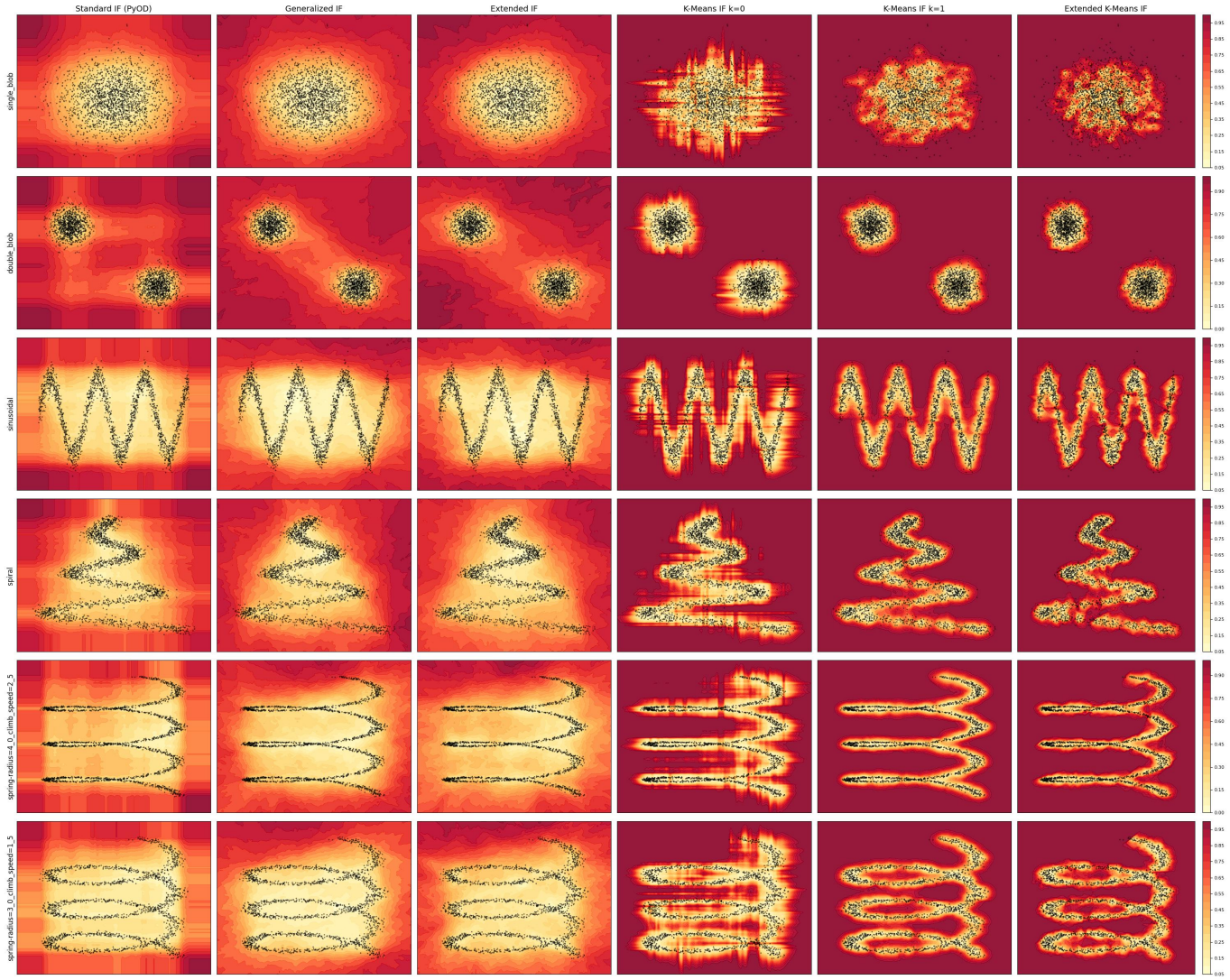


Figure 1: Anomaly detection score maps across different algorithms and synthetic datasets.

the axes, the anomaly score remains low, incorrectly indicating "normal regions." This phenomenon, first identified by Hariri et al. in [1], occurs because Standard IF employs orthogonal hyperplanes parallel to the system's axes to separate data. To address this shortcoming, Hariri et al. proposed the Extended Isolation Forest (EIF) in [1], a variation that utilizes hyperplanes with randomly chosen slopes. Formally, when building a tree node, a random vector $u \in \mathbb{R}^d$ is sampled from $\mathcal{N}(0, I_d)$ and subsequently normalized to $w = \frac{u}{\|u\|_2}$. Additionally, an intercept vector $p \in \mathbb{R}^d$ is randomly chosen within the smallest hypercube enclosing all the datapoints at the node. The separation is then determined by $(x - p)^T \cdot w > 0$ for points going to the right branch, and $(x - p)^T \cdot w \leq 0$ for points going to the left branch. A limitation of this approach is that it can generate branches leading to nodes with no data, resulting in computational inefficiency. This occurs because the intercept vector is chosen within the enclosing hypercube, which does not guarantee the existence of data on both sides of the hyperplane. As the tree grows deeper and the number of points per node decreases, the probability of generating empty nodes increases significantly, related to the ratio between the volume of the convex hull and the volume of the enclosing hypercube. Nevertheless, Figure 1 illustrates how EIF effectively mitigates "ghost artifacts" and eliminates axis-parallel patterns from the anomaly score map. A skeleton overview of the algorithm is provided in 1, 4, and 5.

## 2.3 Generalized Isolation Forest

As discussed in the previous subsection, a significant limitation of EIF is its intercept selection strategy, which can result in branches leading to empty nodes. As the tree depth increases, the probability of generating such empty branches rises, incurring additional computational overhead. To address this issue, Lesouple et al. introduced the Generalized Isolation Forest (GIF) in [3], a variation of EIF. The fundamental difference between GIF and EIF lies in the selection of the separation hyperplane. Instead of randomly selecting a hyperplane within the smallest hypercube enclosing the data, GIF selects a hyperplane that passes through the convex hull of the data. Because the separation hyperplane intersects the convex hull, it

**Algorithm 2:** Create Isolation Tree

**Data:** $X$ - data points, $l$ - max. height, $lvl$ - current height
**Result:** iTree - a node in the isolation tree

**Function** *createITree*$(X, l, lvl)$
  **if** $|X| \leq 1$ *or* $lvl = l$ **then**
    **return** $ITree\{size \leftarrow |X|\}$
  draw $q \sim \mathcal{U}(0, 1)$
  $q \leftarrow \lfloor d \cdot q \rfloor$
  $p_{min} \leftarrow \min_{x \in X} x^{(q)}$
  $p_{max} \leftarrow \max_{x \in X} x^{(q)}$
  draw $p \sim \mathcal{U}(p_{min}, p_{max})$
  $X_l \leftarrow \{x \mid x \in X, x^{(q)} \leq p\}$
  $X_r \leftarrow \{x \mid x \in X, x^{(q)} > p\}$
  **return**
   $ITree\{size \leftarrow |X|,$
    $splitVal \leftarrow p,$
    $splitAtt \leftarrow q,$
    $left \leftarrow \text{createITree}(X_l, l, lvl + 1),$
    $right \leftarrow \text{createITree}(X_r, l, lvl + 1)\}$

**Algorithm 3:** Compute Anomaly Score

**Data:** $x$ - data point, $iTree$ - isolation tree, $lvl$ - height of current node, $l$ - max. height
**Result:** score - anomaly score of data point

**Function** *score*$(x, iTree, lvl, l)$
  **if** $iTree.size \leq 1$ *or* $lvl = 1$ **then**
    **return** $1 + c(iTree.size)$
  $p \leftarrow iTree.splitVal$
  $q \leftarrow iTree.splitAtt$
  **if** $x^{(q)} \leq p$ **then**
    **return** $1 + \text{score}(x, iTree.left, lvl + 1, l)$
  **return** $1 + \text{score}(x, iTree.right, lvl + 1, l)$

is guaranteed to partition the data points into two non-empty subsets. Formally, GIF randomly selects a normal unit vector $w = \frac{u}{\|u\|_2}$, where $u \in \mathbb{R}^d, u \sim \mathcal{N}(0, I_d)$, and subsequently projects the data points onto it via $x^T \cdot w$. It then computes the minimum and maximum projection values and samples the separation value $p$ uniformly within this interval, i.e., $p \sim \mathcal{U}(p_{min}, p_{max})$, where $p_{min} = \min\{x^T \cdot w \mid x \in X\}$ and $p_{max} = \max\{x^T \cdot w \mid x \in X\}$. Although the overall anomaly score distribution does not shift significantly compared to the EIF algorithm, as depicted in 1, the primary advantage lies in the improved computational performance derived from the elimination of empty branches. Algorithms 1, 6, and 7 provide a skeletal overview of GIF.

## 2.4 K-Means Isolation Forest

A distinct variation of the Isolation Forest algorithm is the K-Means Isolation Forest (K-Means IF), introduced by Karczmarek et al. in [2], which represents a hybrid approach combining isolation and density-based anomaly detection methods. The primary innovation of this algorithm lies in its strategy for selecting separation hyperplanes. Unlike the Standard IF, which randomly selects a separation value on a random component (effectively choosing a hyperplane orthogonal to a random axis), this algorithm randomly selects a component, projects all data onto it, and then applies the K-Means clustering algorithm to determine the partition boundaries. Consequently, a node will have $k$ child nodes, where $k$ is the number of identified clusters. The value of $k$ is determined using the "elbow-rule," which we define after a brief review of the K-Means algorithm. In essence, K-Means IF constructs a tree node by first randomly selecting a component, projecting the data onto it, and then assigning each data point to the cluster it most likely belongs to. This assignment is based on the distance from the point to the centroid of the cluster. Since clusters are formed in a 1-dimensional space, the assignment boundaries are dictated by hyperplanes perpendicular to the randomly chosen component. The resulting structure is no longer a binary tree, but a tree with an arbitrary number of child nodes. We next recall the K-Means algorithm and the "elbow-rule" before detailing the K-Means IF algorithm and its anomaly scoring methodology.

Given $k$, the number of clusters, the K-Means algorithm begins by randomly selecting $k$ distinct data points from the dataset (these points serve as the initial centroids of the clusters). It then iterates through the remaining data points, assigning each to the cluster with the closest centroid based on Euclidean distance. Once all points are assigned, new centroids are computed by calculating the mean of the points within each cluster. The assignment process repeats with the new centroids until convergence is achieved—either a maximum number of iterations is reached or the variation in centroids becomes negligible. Since the algorithm's outcome depends on the initial selection of the $k$ centers, it is typically run multiple times, and the best clustering (in terms of minimal intra-cluster variation) is selected.

The "elbow-rule" is a heuristic method for determining a suitable number of clusters to partition a dataset. Given that the worst clustering occurs at $k = 1$ (variance equal to the total dataset variance) and the best at $k = |X|$ (where each point forms its own cluster, resulting in zero variance), we observe that the Sum of Squared Errors, $SSE = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$, decreases as $k$ increases, where $\mu_i$ is the centroid of the $i$-th cluster, and $C_i = \{x \in X \mid |x - \mu_i|_2 \leq |x - \mu_j|_2, \forall j = \overline{1, \ldots, k}\}$. Knowing that increasing $k$ generally improves clustering, the goal is to find a $k$ that provides diminishing returns. Using the "elbow-rule," we iterate through different values of $k$ starting from 1 until the improvement in $SSE$ becomes marginal. If $SSE$ is plotted against $k$, a sharp decrease is initially observed, followed by a flattening of the curve; the "elbow" point marks where the rate of decrease slows

**Algorithm 4:** Create Extended ITree

**Data:** $X$ - data points, $l$ - max. height, $lvl$ - current height
**Result:** iTree - a node in the isolation tree

**Function** *createITree*$(X, l, lvl)$
  **if** $|X| \leq 1$ *or* $lvl = l$ **then**
    **return** $iTree\{size \leftarrow |X|\}$
  draw $u \sim \mathcal{N}(0, I_d)$
  $w \leftarrow \frac{u}{\|u\|_2}$
  $p_{min} \leftarrow \min X$, $p_{min} \in \mathbb{R}^d$
  $p_{max} \leftarrow \max X$, $p_{max} \in \mathbb{R}^d$
  draw $p \sim \mathcal{U}(p_{min}, p_{max})$
  $X_l \leftarrow \{x \mid x \in X, (x-p)^T \cdot w \leq 0\}$
  $X_r \leftarrow \{x \mid x \in X, (x-p)^T \cdot w > 0\}$
  **return**
    $iTree\{size \leftarrow |X|,$
      $intercept \leftarrow p,$
      $normal \leftarrow w,$
      $left \leftarrow \text{createITree}(X_l, l, lvl+1),$
      $right \leftarrow \text{createITree}(X_r, l, lvl+1)\}$

**Algorithm 5:** Compute Anomaly Score EIF

**Data:** $x$ - data point, $iTree$ - extended isolation tree, $lvl$ - height of current node, $l$ - max. height
**Result:** score - anomaly score of data point

**Function** *score*$(x, iTree, lvl, l)$
  **if** $iTree.size \leq 1$ *or* $lvl = 1$ **then**
    **return** $1 + c(iTree.size)$
  $w \leftarrow iTree.normal$
  $p \leftarrow iTree.intercept$
  **if** $(x-p)^T \cdot w \leq 0$ **then**
    **return** $1 + \text{score}(x, iTree.left, lvl+1, l)$
  **return** $1 + \text{score}(x, iTree.right, lvl+1, l)$

significantly.

Returning to the K-Means based Isolation Forest, at every node, we compute the clustering of the resident data points projected onto a randomly chosen component using the elbow rule (typically finding $k \in \{4, 5\}$). The number of branches originating from that node corresponds to the number of clusters found, with each branch receiving the points closest to the centroid of its respective cluster. The score obtained at each split is computed by the formula:

$$s(x) = 1 - \frac{d(x, \mu)}{r} \tag{3}$$

where $\mu$ is the cluster center and $r$ is the cluster radius. We emphasize that if $x$ belongs to the cluster, its distance to the center will be smaller than the radius (i.e., $\frac{d(x,\mu)}{r} \leq 1$), thus $s(x) \geq 0$. If $x$ is very close to the cluster center, $s(x)$ approaches 1, indicating confidence that the point is normal. Conversely, if $x$ is far from the cluster, then $\frac{d(x,\mu)}{r} \gg 1$, resulting in a highly negative $s(x)$, which indicates an anomaly. Note that a split is chosen with respect to the closest cluster to the input data point. To compute the anomaly score at the forest level, we use the formula:

$$a(x) = 1 - \frac{1}{t} \sum_{i=1}^{t} \sum_{j=1}^{M_i} s_j(x) \tag{4}$$

where $M_i, i = \overline{1, \ldots, t}$ is the sequence of splits taken by an input point for tree $i$. If a point is anomalous, $a(x)$ will be very large. On the other hand, a normal point will have a score close to 0. Since $s(x) \in (-\infty, 1]$, this implies $a(x) \in [0, \infty)$. In our experiments, to ensure consistent anomaly scores across all evaluated methods, we applied the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ to the final anomaly score. With this transformation, normal points receive scores close to 0, while anomalous ones receive scores close to 1. A skeletal overview of the K-Means IF algorithm can be found in 1, 8 and 9.

# 3 Alternative solutions

In this section, we introduce two novel variations of the Isolation Forest algorithm that combine the random projection strategies of Extended IF (EIF) with the clustering-based partitioning of K-Means IF. We refer to these newly introduced methods as Subspace K-Means IF and Extended K-Means IF.

## 3.1 Subspace K-Means Isolation Forest

The Subspace K-Means Isolation Forest is a variation of the Isolation Forest algorithm that brings together the random selection of a subspace (parallel to the coordinate axes) with the clustering-based partitioning mechanism introduced by K-Means IF. More precisely, an isolation node is recursively constructed by first selecting $k$ random components, where $1 \leq k \leq d$, and then projecting the dataset into the subspace defined by these selected components. Subsequently, the K-Means clustering algorithm is applied, utilizing the "elbow" rule to cluster the points and generate child nodes. The assignment of a point to a cluster follows the same logic as in K-Means IF, specifically determining membership based on the Euclidean distance to the nearest centroid. It is important to note that this algorithm represents a generalization of K-Means IF, as the standard K-Means IF projects data onto a single dimension before clustering and partitioning. In fact, the Subspace K-Means Isolation Forest with $k = 1$ is mathematically equivalent to the K-Means IF introduced by Karczmarek et al. in [2]. However, the distinction between these two versions lies in the complexity of the partition boundaries. Recall that in the original K-Means IF, the separation boundaries are effectively hyperplanes orthogonal to the coordinate axes. In contrast, in the subspace version, the resulting separation boundaries form polytopes that enclose the data clusters within the projected dimensions, with the dimensionality defined by the hyperparameter $k$. For instance, if $X \subset \mathbb{R}^2$ and

**Algorithm 6:** Create Generalized ITree

**Data:** $X$ - data points, $l$ - max. height, $lvl$ - current height
**Result:** iTree - a node in the isolation tree

**Function** *createITree*$(X, l, lvl)$
  **if** $|X| \leq 1$ *or* $lvl = l$ **then**
    **return** $iTree\{size \leftarrow |X|\}$
  draw $u \sim \mathcal{N}(0, I_d)$
  $w \leftarrow \frac{u}{\|u\|_2}$
  $p_{min} \leftarrow \min\{x^T \cdot w \mid x \in X\}$
  $p_{max} \leftarrow \max\{x^T \cdot w \mid x \in X\}$
  draw $p \sim \mathcal{U}(p_{min}, p_{max})$
  $X_l \leftarrow \{x \mid x \in X, x^T \cdot w \leq p\}$
  $X_r \leftarrow \{x \mid x \in X, x^T \cdot w > p\}$
  **return**
   $iTree\{size \leftarrow |X|,$
    $thresh \leftarrow p,$
    $normal \leftarrow w,$
    $left \leftarrow$ createITree$(X_l, l, lvl + 1),$
    $right \leftarrow$ createITree$(X_r, l, lvl + 1)\}$

$k = 2$ (i.e., clustering is performed in the original space at each node), the resulting partitions resemble Voronoi cells. As the depth of the tree increases, these cells become increasingly granular, tightly encapsulating the data. This characteristic allows the method to capture correlations between attributes more effectively than the original K-Means IF, which treats dimensions independently during clustering. The anomaly scoring for a point is calculated using the same rules as in the original K-Means IF: at every node level, we calculate the membership score for the assigned cluster and average these scores across all trees. Given the substantial algorithmic similarity to the original method—differing primarily in the multi-dimensional projection step—we omit a redundant explicit overview of the algorithm skeleton.

**Algorithm 7:** Compute Anomaly Score GIF

**Data:** $x$ - data point, $iTree$ - generalized isolation tree, $lvl$ - height of current node, $l$ - max. height
**Result:** score - anomaly score of data point

**Function** *score*$(x, iTree, lvl, l)$
  **if** $iTree.size \leq 1$ *or* $lvl = 1$ **then**
    **return** $1 + c(iTree.size)$
  $w \leftarrow iTree.normal$
  $p \leftarrow iTree.thresh$
  **if** $x^T \cdot w \leq p$ **then**
    **return** $1 +$ score$(x, iTree.left, lvl + 1, l)$
  **return** $1 +$ score$(x, iTree.right, lvl + 1, l)$

**Algorithm 8:** Create K-Means ITree

**Data:** $X$ - data points, $l$ - max. height, $lvl$ - current height
**Result:** iTree - a node in the isolation tree

**Function** *createITree*$(X, l, lvl)$
  **if** $|X| \leq 1$ *or* $lvl = l$ **then**
    **return** $iTree\{size \leftarrow |X|\}$
  draw $u \sim \mathcal{U}(0, 1)$
  $q \leftarrow \lfloor d \cdot q \rfloor$
  $X' = \{x^{(q)} \mid x \in X\}$
  Let $C$ be a set of sets representing the partitioning of dataset $X$ into clusters with respect to $X'$ using "elbow" rule and K-Means algorithm
  Let $\mu$ be the set of centroids of the resulted clusters
  Let $radii$ be the set of radii of the resulted clusters
  $child\_nodes = \emptyset$
  **for** $i = 1 : k$ **do**
    $child\_nodes \leftarrow$
     $child\_nodes \cup \{createITree(C_i, l, lvl + 1)\}$
  **return** $iTree\{size \leftarrow |X|,$
    $projAtt \leftarrow q,$
    $centroids \leftarrow \mu,$
    $radii \leftarrow radii,$
    $children \leftarrow child\_nodes\}$

## 3.2 Extended K-Means Isolation Forest

The Extended K-Means Isolation Forest algorithm represents another variation of the Isolation Forest framework, offering an alternative generalization to the K-Means IF. The primary distinction of this method lies in its projection strategy: rather than projecting data onto a single axis (as in the original K-Means IF) or a subset of axes (as in Subspace K-Means IF), this method provides the flexibility to project data into a general subspace via a random normal projection matrix. Following this projection, the data points are clustered and scored using the same mechanisms employed in the previously discussed K-Means IF variants.

Formally, when constructing a node, we initialize a random matrix $W \in \mathbb{R}^{d \times k}$ (where elements are drawn from a standard normal distribution), with $d$ denoting the dimensionality of the input data and $k$ the dimensionality of the target subspace. We then compute the dot product $X' = X \cdot W \in \mathbb{R}^{n \times k}$, which yields the projection of the initial data into the new space. In this context, $W$ acts as a weight matrix, quantifying the contribution of each component in the initial space to the components in the target space; effectively, $W$ captures correlations between features in the original space.

To better visualize this methodology, consider the following cases. If $k = 1$, the initial dataset is projected onto a line. Furthermore, if $W = [e_i]$, where $e_i \in \mathbb{R}^d$ is a

**Algorithm 9:** Compute Anomaly Score K-Means IF

**Data:** $x$ - data point,
$iTree$ - K-Means isolation tree,
$lvl$ - height of current node,
$l$ - max. height
**Result:** score - anomaly score of data point

**Function** $score(x, iTree, lvl, l)$
  **if** $iTree.size \leq 1$ *or* $lvl = 1$ **then**
    **return** 0
  $child\_nodes \leftarrow iTree.children$
  $q \leftarrow iTree.projAtt$
  $radii \leftarrow iTree.radii$
  $\mu \leftarrow iTree.centroids$
  $best\_score \leftarrow -\infty$
  $best\_child \leftarrow \emptyset$
  **for** $i = 1 : length(child\_nodes)$ **do**
    $s \leftarrow 1 - \frac{x^{(q)} - \mu_i}{radii_i}$
    **if** $s > best\_score$ **then**
      $best\_score \leftarrow s$
      $best\_child \leftarrow child\_nodes_i$
  **return**
   $best\_score + score(x, best\_child, lvl + 1, l)$

standard basis vector, the algorithm simply projects the data onto the $i$-th component, rendering the Extended K-Means IF equivalent to the original K-Means IF. If $W$ is composed of selected basis vectors $W = [e_{i_1}, e_{i_2}, \ldots, e_{i_k}]$, with distinct indices $i_1, \ldots, i_k \leq d$, the algorithm projects the data into the subspace spanned by these specific components, making it equivalent to the Subspace K-Means IF introduced earlier. Finally, if $W$ is chosen at random and $k = d$, the newly projected data represents a rotation of the data in the original space. Once the data is projected, we simply apply the K-Means clustering mechanics (utilizing the elbow rule) and the same scoring function defined for the K-Means IF methods. Given the structural similarity to the prior algorithms—differing only in the selection of the projection matrix and the data transformation step—we omit a redundant algorithmic skeleton, as the generalization is straightforward for the reader.

# 4 Experiments and Results

In this section, we present the experimental results obtained using Standard Isolation Forest (IF), Extended IF (EIF), Generalized IF (GIF), K-Means IF, and our proposed variations: Subspace K-Means IF and Extended K-Means IF. We evaluated these methods on 13 benchmark datasets from the ODDS library, which are detailed in Table 2. The performance metrics considered include ROC-AUC, PR-AUC, and training time. For the Standard IF, EIF, GIF, and K-Means IF algorithms, we adopted the parameter settings listed in Table 1, specifically a subsampling size of 256, a maximum tree height of 8, and a forest size of 100 trees. In contrast, for experiments involving

| Dataset | Samples | Features | Cont. |
|---|---|---|---|
| Aloi | 50000 | 27 | 3.02 |
| ANN Thyroid | 6916 | 21 | 3.61 |
| Breast Cancer | 367 | 30 | 2.72 |
| Cardio | 1831 | 21 | 9.61 |
| Forest Cover | 286048 | 10 | 0.96 |
| Ionosphere | 351 | 33 | 35.90 |
| Letter | 1600 | 32 | 6.25 |
| Mammography | 11183 | 6 | 2.32 |
| Pen Gloabl | 809 | 16 | 11.12 |
| Pen Local | 6724 | 16 | 0.15 |
| Satellite | 6435 | 36 | 31.64 |
| Shuttle | 46464 | 9 | 1.89 |
| Speech | 3686 | 400 | 1.65 |

Table 2: ODDS benchmark datasets stats

K-Means Based Methods, the ensemble size was reduced to 10 estimators due to the significantly higher computational cost associated with both training and inference. However, the subsampling size and maximum tree depth remained consistent with the other methods. The maximum number of clusters allowed at a tree node was set to 5, implying that a tree in-node could have a maximum of 5 children and a minimum of 2. Given the stochastic nature of tree construction, each experiment was repeated

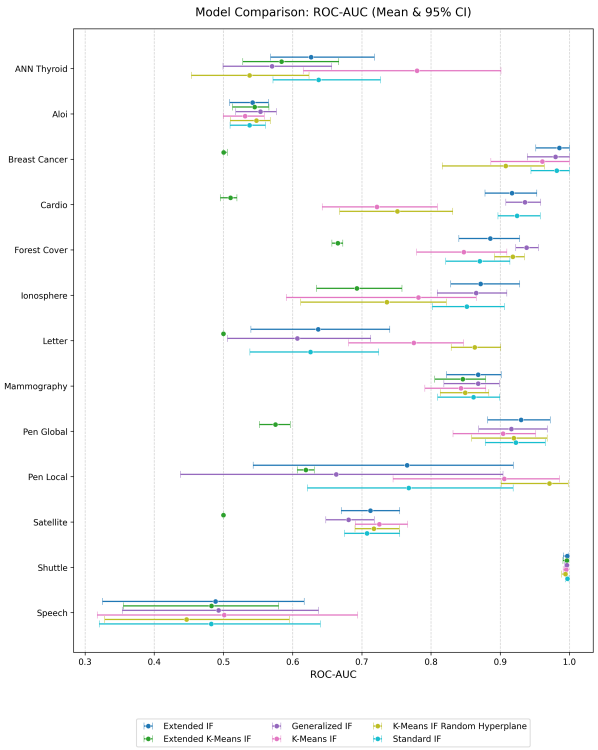


Figure 2: Comparative analysis of ROC-AUC performance across ODDS benchmark datasets. The plot displays the mean ROC-AUC scores obtained by Standard IF, EIF, GIF, K-Means IF, and the proposed Subspace and Extended K-Means IF variants. Error bars represent the 95% confidence intervals derived from 50 independent experimental runs using different random seeds.

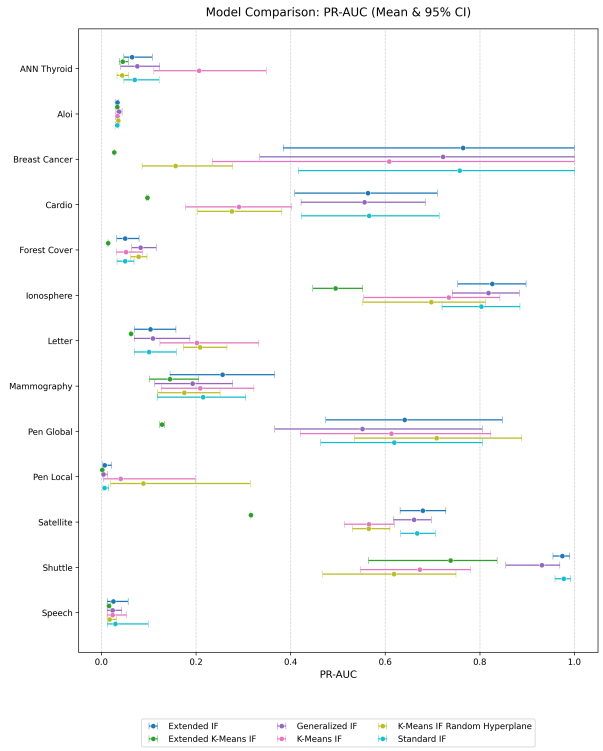


Figure 3: Comparative analysis of PR-AUC performance across ODDS benchmark datasets. The plot displays the mean PR-AUC scores obtained by Standard IF, EIF, GIF, K-Means IF, and the proposed Subspace and Extended K-Means IF variants. Error bars represent the 95% confidence intervals derived from 50 independent experimental runs using different random seeds.

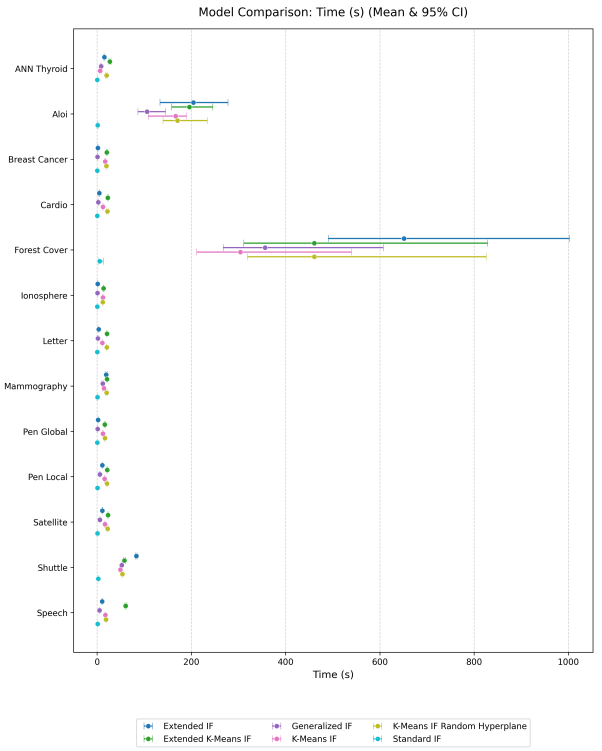


Figure 4: Comparative analysis of training time duration across ODDS benchmark datasets. The plot displays the mean training times obtained by Standard IF, EIF, GIF, K-Means IF, and the proposed Subspace and Extended K-Means IF variants. Error bars represent the 95% confidence intervals derived from 50 independent experimental runs using different random seeds.

50 times with different random seeds. We report the mean performance along with the 95% confidence interval (reflecting the 2.5% to 97.5% quantile range). The results are illustrated in Figures 2, 3, and 4.

We further evaluated the methods against diverse synthetic datasets drawn from multiple distributions, including single-cluster blobs, double-cluster blobs, sinusoidal shapes, spirals, and helicoidal structures with varying parameters. The resulting anomaly score heatmaps are presented in Figure 1. As observed, the K-Means IF variants generally exhibit a superior fit to the underlying data distributions, with Subspace K-Means IF demonstrating the most robust performance, followed closely by Extended K-Means IF. Notably, because the original K-Means IF applies clustering strictly along single dimensions, linear artifacts remain visible in its score maps, though these are less pronounced than the axis-parallel artifacts characteristic of the Standard IF.

Figures 5, 6, 7, 8, 9, and 10 provide detailed visualizations for each method across the dataset distributions. These figures depict the structure of a sample tree, the isolation path taken by a selected inlier and outlier, the constructed separation hyperplanes, and a radial view of the forest highlighting the top 100 inlier and outlier scores. A distinct structural difference is observable in the K-Means-based methods, which tend to generate shorter and wider trees compared to their random projection counterparts. Furthermore, for IF, EIF, and GIF, the radial plots reveal a visible clustering of anomalies near the center, confirming that anomalies are isolated closer to the root. Regarding the separation boundaries, the Subspace K-Means IF yields particularly interesting results, isolating data clusters via complex Voronoi cell structures. While the separation boundaries for Extended K-Means IF are not explicitly plotted due to the complexity of visualizing high-dimensional projections, they are analytically similar to those of Subspace K-Means IF, differing only in that the Voronoi cells are formed in the projected space rather than the original feature space. To ensure reproducibility, the complete code used for these experiments is available in our GitHub repository here.

# 5 Conclusion

In this work, we presented a comparative study of isolation-based anomaly detection methods and introduced two hybrid variants: *Subspace K-Means IF* and *Extended K-Means Isolation Forest*. These new methods integrate random projections with clustering to better capture complex, non-linear data distributions. Our experiments show that while the proposed algorithms effectively mitigate the axis-parallel artifacts of Standard IF, this robustness comes with increased computational cost. Ultimately, the choice of algorithm depends on the

specific trade-off between speed and geometric sensitivity required by the application; Standard IF, EIF and GIF remain ideal for efficiency, while K-Means IF and our hybrid approaches are superior for detecting anomalies in complex manifolds.

# References


[1] Sahand Hariri, Matias Carrasco Kind, and Robert J Brunner. Extended isolation forest. *IEEE transactions on knowledge and data engineering*, 33(4):1479–1489, 2019.

[2] Paweł Karczmarek, Adam Kiersztyn, Witold Pedrycz, and Ebru Al. K-means-based isolation forest. *Knowledge-based systems*, 195:105659, 2020.

[3] Julien Lesouple, Cédric Baudoin, Marc Spigai, and Jean-Yves Tourneret. Generalized isolation forest for anomaly detection. *Pattern Recognition Letters*, 149:109–119, 2021.

[4] 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.

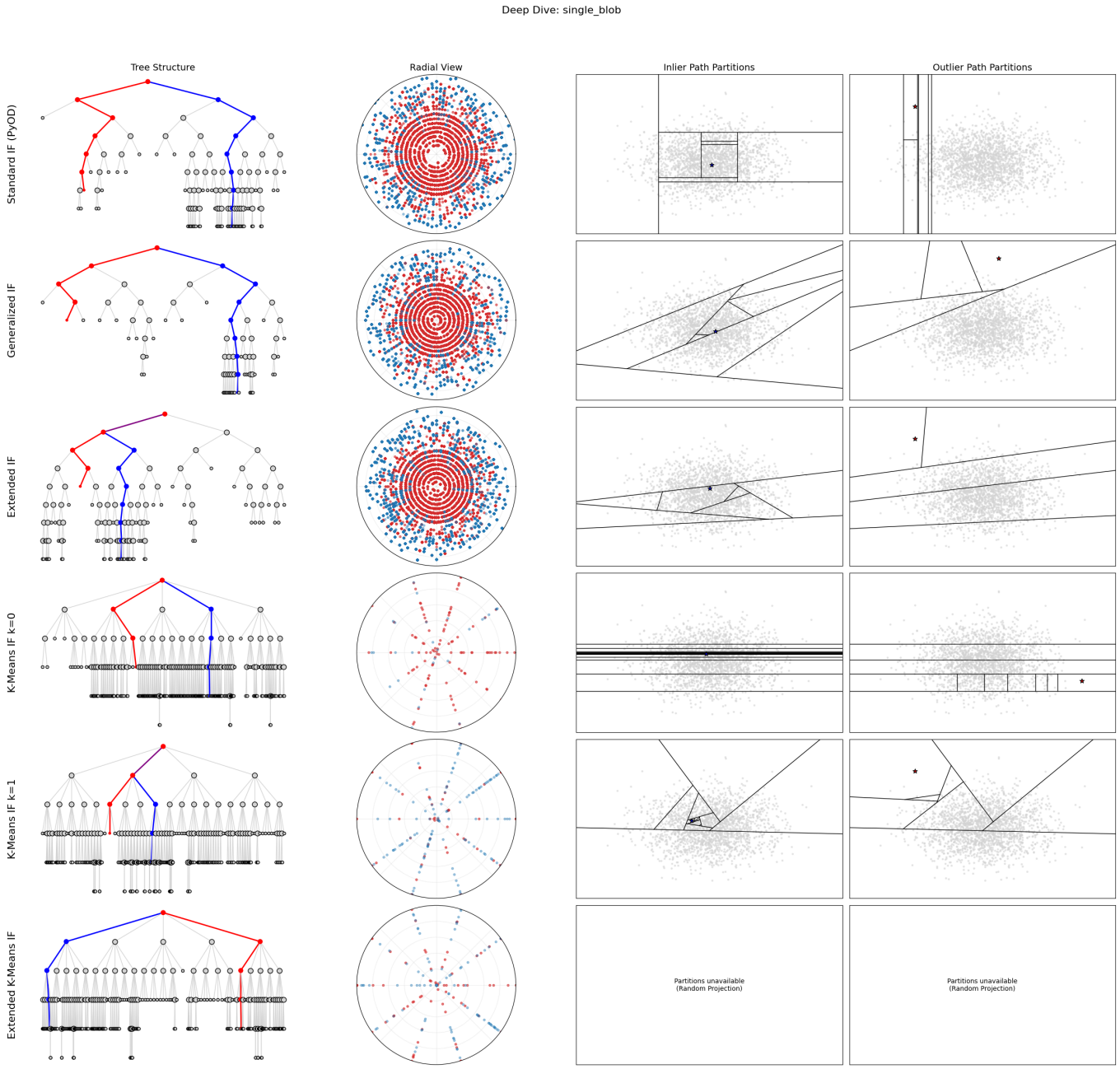


Figure 5: Detailed visualization of the internal mechanisms for each anomaly detection method, exemplified on the 'single_blob' dataset. The rows represent different algorithms. The columns display (from left to right): the structure of a representative tree; a radial view of the forest scores; the partition boundaries (separation hyperplanes) along the isolation path of a selected inlier; and the partition boundaries along the isolation path of a selected outlier.

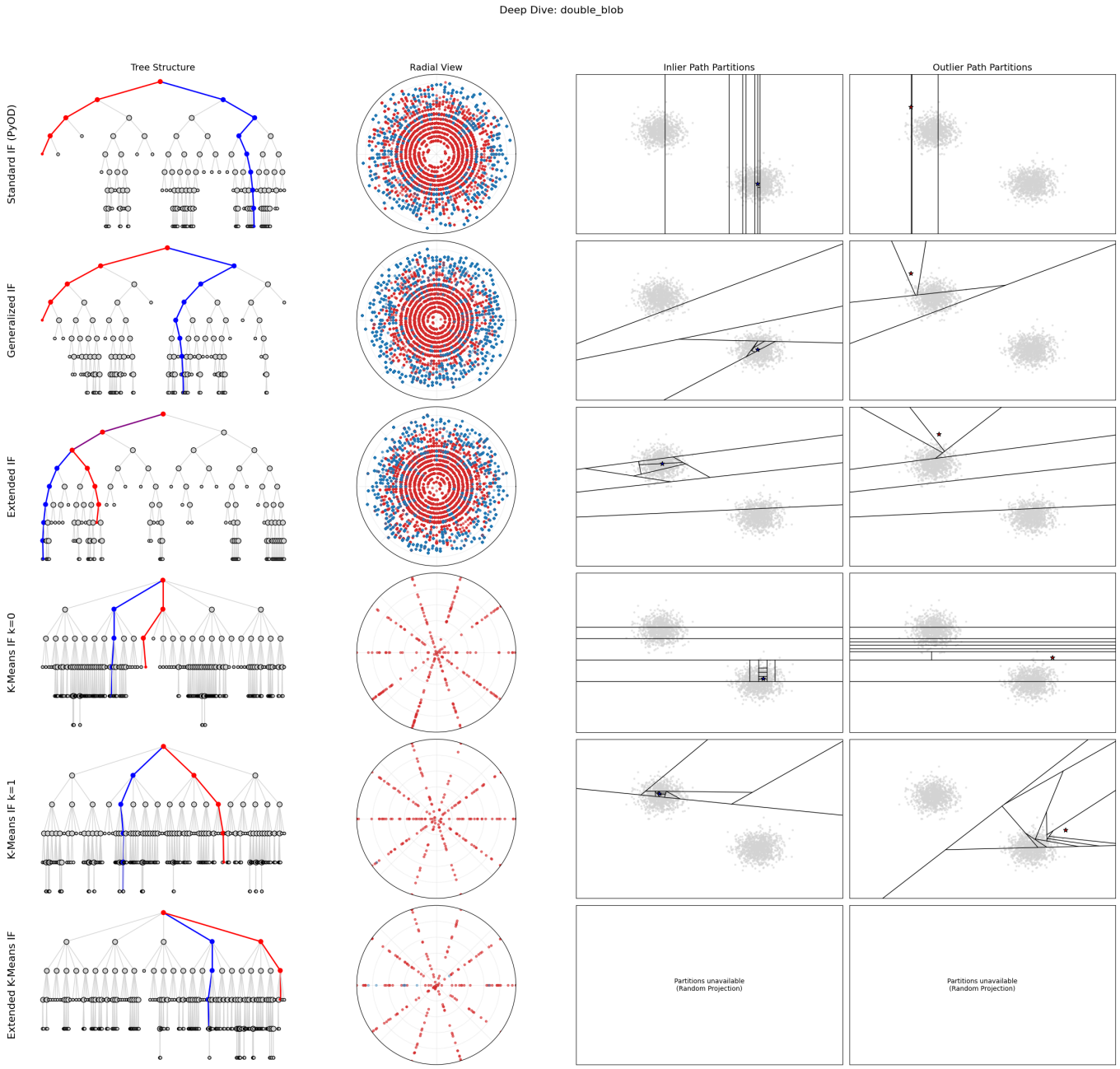


Figure 6: Detailed visualization of the internal mechanisms for each anomaly detection method, exemplified on the 'double_blob' dataset. The rows represent different algorithms. The columns display (from left to right): the structure of a representative tree; a radial view of the forest scores; the partition boundaries (separation hyperplanes) along the isolation path of a selected inlier; and the partition boundaries along the isolation path of a selected outlier.

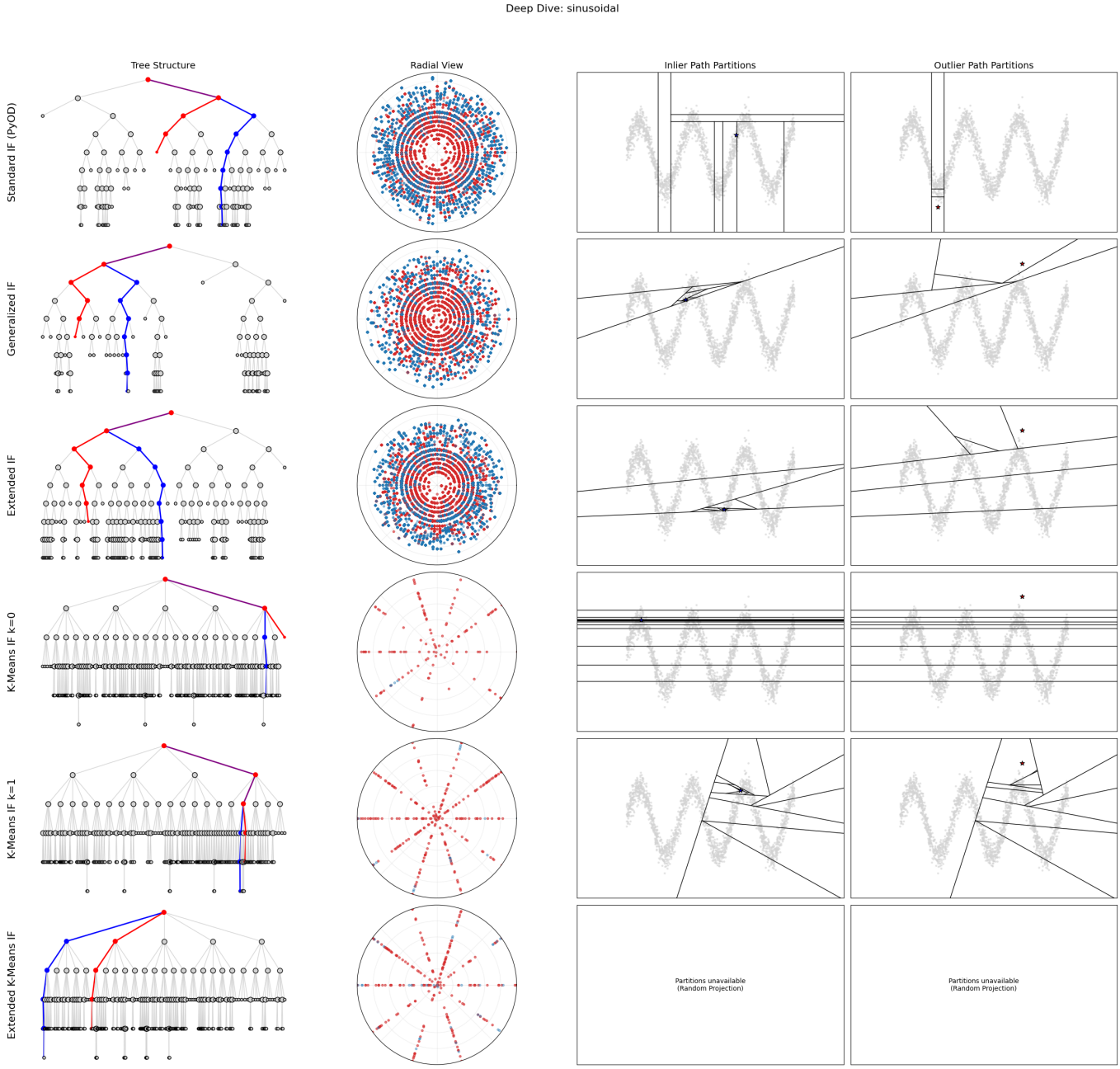


Figure 7: Detailed visualization of the internal mechanisms for each anomaly detection method, exemplified on the 'sinusoidal' dataset. The rows represent different algorithms. The columns display (from left to right): the structure of a representative tree; a radial view of the forest scores; the partition boundaries (separation hyperplanes) along the isolation path of a selected inlier; and the partition boundaries along the isolation path of a selected outlier.

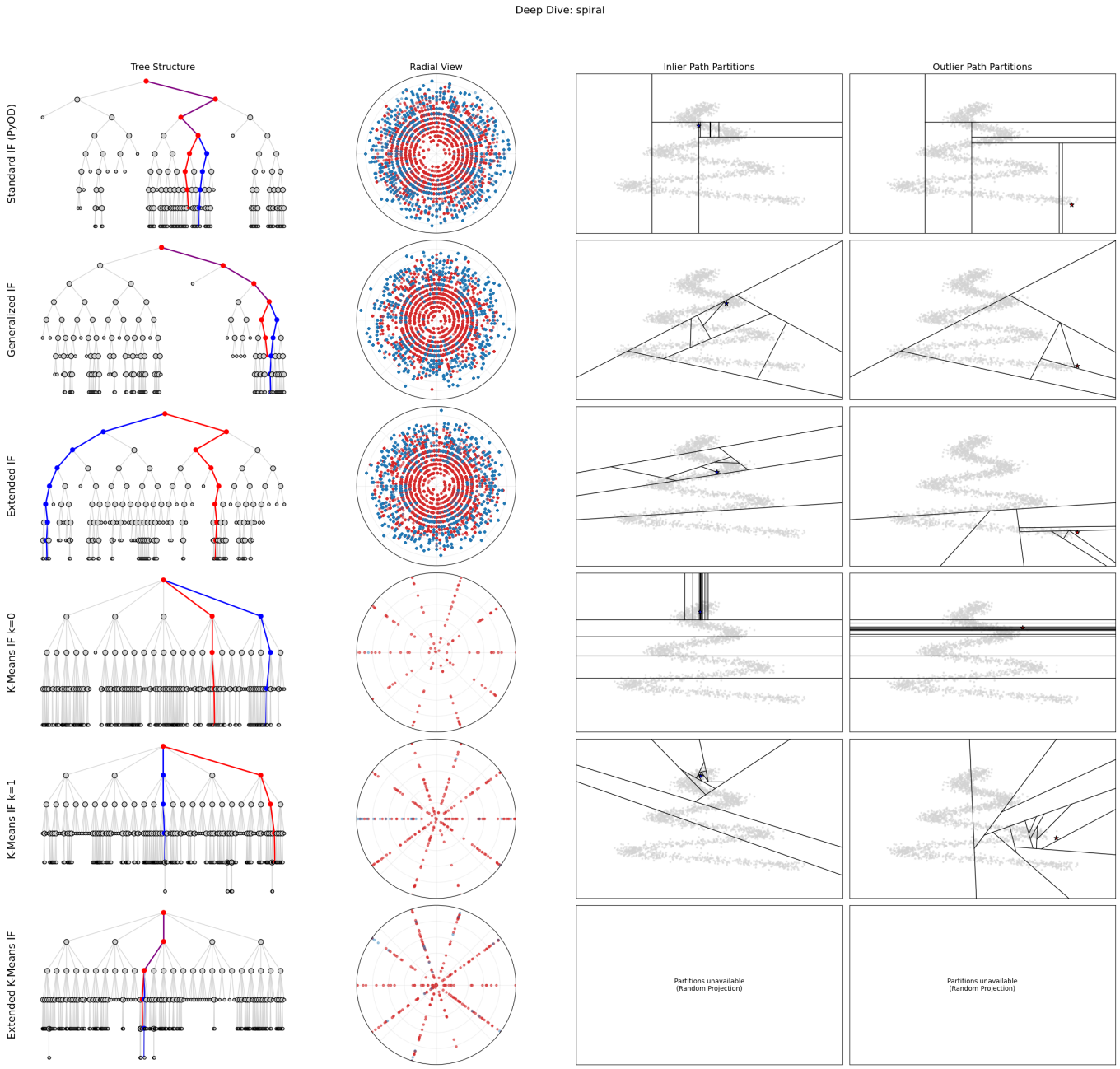


Figure 8: Detailed visualization of the internal mechanisms for each anomaly detection method, exemplified on the 'spiral' dataset. The rows represent different algorithms. The columns display (from left to right): the structure of a representative tree; a radial view of the forest scores; the partition boundaries (separation hyperplanes) along the isolation path of a selected inlier; and the partition boundaries along the isolation path of a selected outlier.

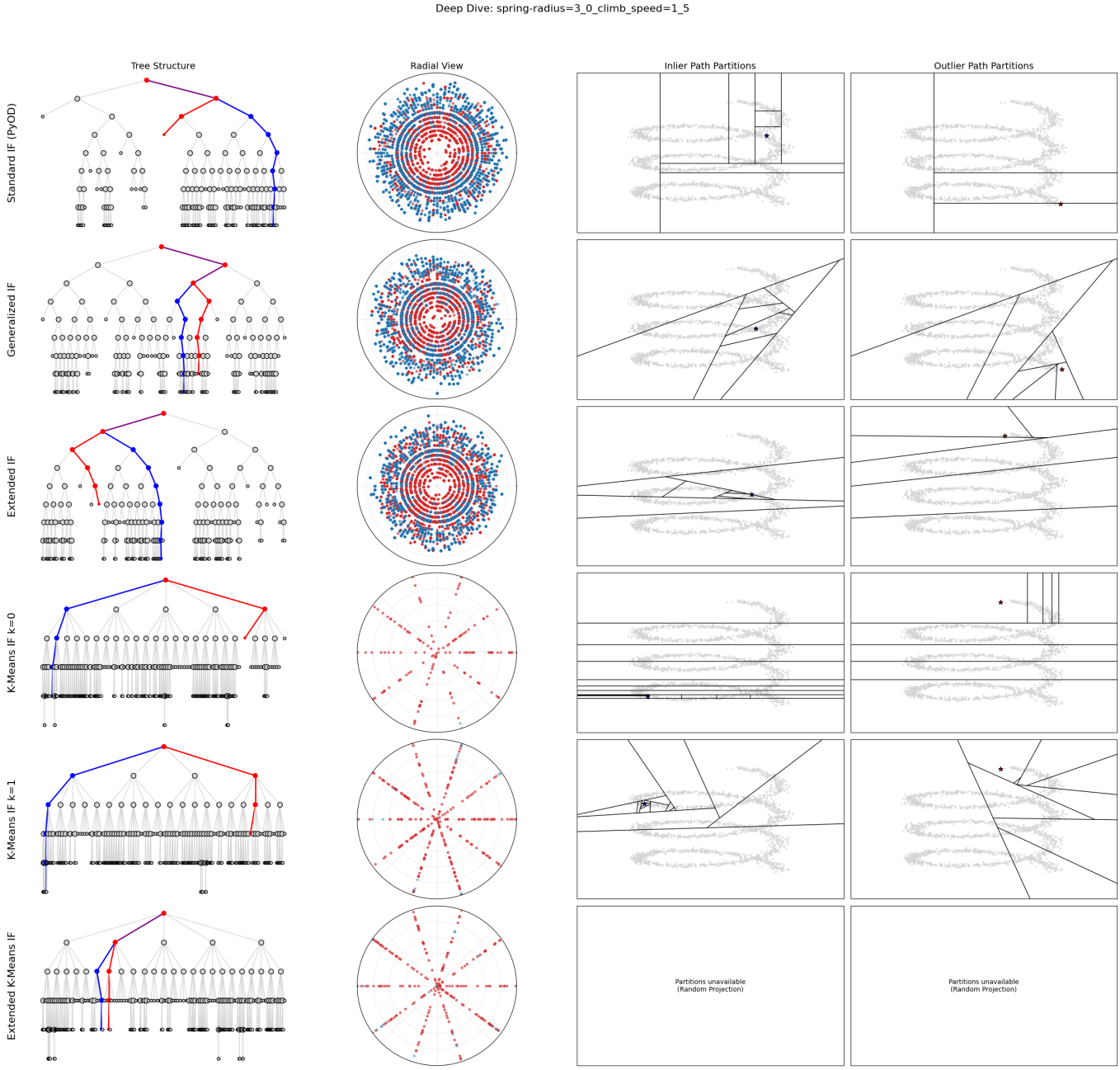


Figure 9: Detailed visualization of the internal mechanisms for each anomaly detection method, exemplified on the 'spring-radius=3_0_climb_speed=1_5' dataset. The rows represent different algorithms. The columns display (from left to right): the structure of a representative tree; a radial view of the forest scores; the partition boundaries (separation hyperplanes) along the isolation path of a selected inlier; and the partition boundaries along the isolation path of a selected outlier.

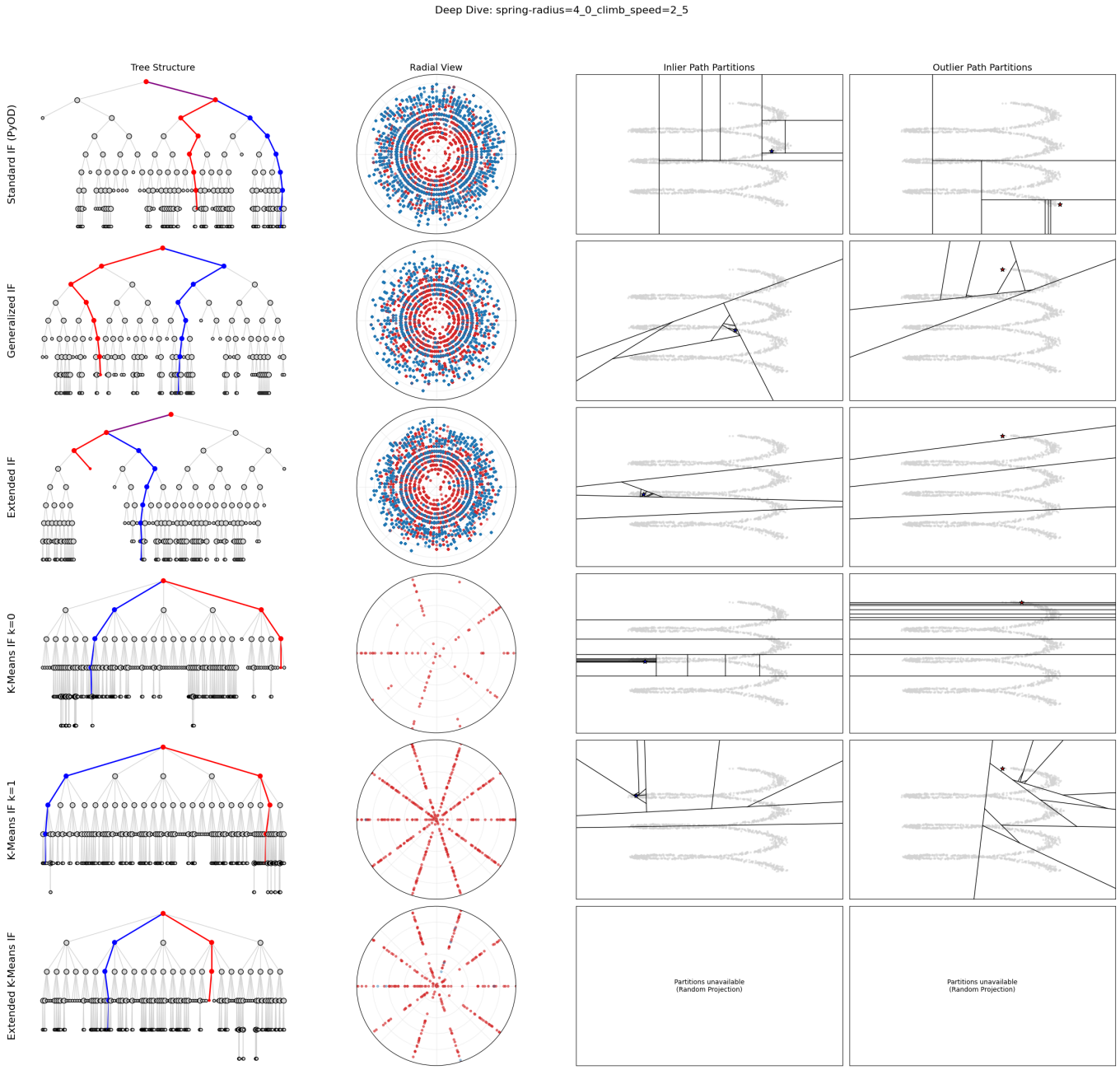


Figure 10: Detailed visualization of the internal mechanisms for each anomaly detection method, exemplified on the 'spring-radius=4_0_climb_speed=2_5' dataset. The rows represent different algorithms. The columns display (from left to right): the structure of a representative tree; a radial view of the forest scores; the partition boundaries (separation hyperplanes) along the isolation path of a selected inlier; and the partition boundaries along the isolation path of a selected outlier.