A Performance Comparison between Isolation Forest and K-Means Isolation Forest on Anomaly Detection Tasks

Răzvan-Nicolae Căpățînă Mihnea-Vicențiu Bucă

Faculty of Mathematics and Informatics
University of Bucharest

February 2, 2026

Outline

Isolation Forest (IF) - Implementation

Core Idea

Anomalies are **few and different** $\rightarrow$ easier to isolate

Tree Construction (Binary Splitting)

1. **Feature Selection:** Choose random dimension q (uniform)
2. **Split Value:** Choose random value $v \in [\min_q, \max_q]$
3. **Partition:**
   - Left child: $D_L = \{x \mid x_q < v\}$
   - Right child: $D_R = \{x \mid x_q \geq v\}$
4. **Recurse** until termination conditions

Termination Conditions

- Maximum depth l reached
- Node contains ≤ 1 sample
- All samples identical

Isolation Forest - Anomaly Score

Expected Path Length

$$h(x) = \begin{cases} d(x) \text{ if } x \text{ is a natural leaf (not max-depth blocked)} \\ d(x) + c(m(x)) \text{ if } x \text{ is a leaf due to max-depth pruning} \end{cases}$$

$m(x) = \#$ of trained samples for leaf x

Anomaly Score Formula

$$s(x, n) = 2^{-\frac{\mathbb{E}[h(x)]}{c(n)}} \quad \text{where} \quad c(n) = 2[\ln(n-1) + 0.5772] - \frac{2(n-1)}{n}$$

- $\mathbb{E}[h(x)] =$ mean path length across all trees
- $c(n) =$ normalization factor (avg. BST search length)
- $0.5772 =$ Euler-Mascheroni constant

Interpretation

Score $\approx 1 \rightarrow$ Anomaly — Score $\approx 0 \rightarrow$ Normal

Isolation Forest - Ensemble

Ensemble Construction

Build t independent Isolation Trees, each on a random subsample

Recommended Parameters (Original Paper)

- **Sub-sampling size:** $n = 256$
- **Maximum depth:** $l = \lceil \log_2 n \rceil = 8$
- **Ensemble size:** $t = 100$ trees

Anomaly Score Aggregation

1. For each tree i: compute path length $h_i(x)$
2. Average across ensemble: $\bar{h}(x) = \frac{1}{t} \sum_{i=1}^{t} h_i(x)$
3. Final score: $s(x) = 2^{-\bar{h}(x)/c(n)}$

Threshold

Use contamination rate or default threshold $= 0.5$

Outline

k-Means-Based Isolation Forest (kMIF)

Key Modification

Replace **binary splits** with **multi-branch k-Means clustering**

Tree Construction

1. **Feature Selection:** Choose random dimension q (same as IF)
2. **k-Means Clustering:** Apply to values along dimension q
3. **Optimal k:** Determined dynamically via **Elbow Rule**
4. **Partition:** Split into k subsets (children nodes)
5. **Store:** Cluster centers and radii for scoring

Advantages

- **Data-aware partitioning** adapts to local density
- More children in dense regions, fewer in sparse regions
- Theoretically better fit to complex data structures

k-Means IF - Anomaly Score

Normality Score at Each Node

$$s_j(x) = \max\left(0, 1 - \frac{d(x, c_q)}{d(c_l, c_q)}\right) \in [0, 1]$$

- $d(x, c_q)$ = distance from x to its cluster center
- $d(c_l, c_q)$ = cluster radius (distance to boundary)
- Clamped to $[0, 1]$ for out-of-sample data

Final Anomaly Score

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

M_i = path depth in tree i, score depends on **cluster distance**, not path length

k-Means IF - Ensemble

Ensemble Construction

Build t multi-branch trees, each using k-Means splits

Parameters

- Max depth: $l = 9$
- Sub-sample: $n = 256$
- Trees: $t = 100$
- k candidates: $\{2, 3, 5, 7\}$

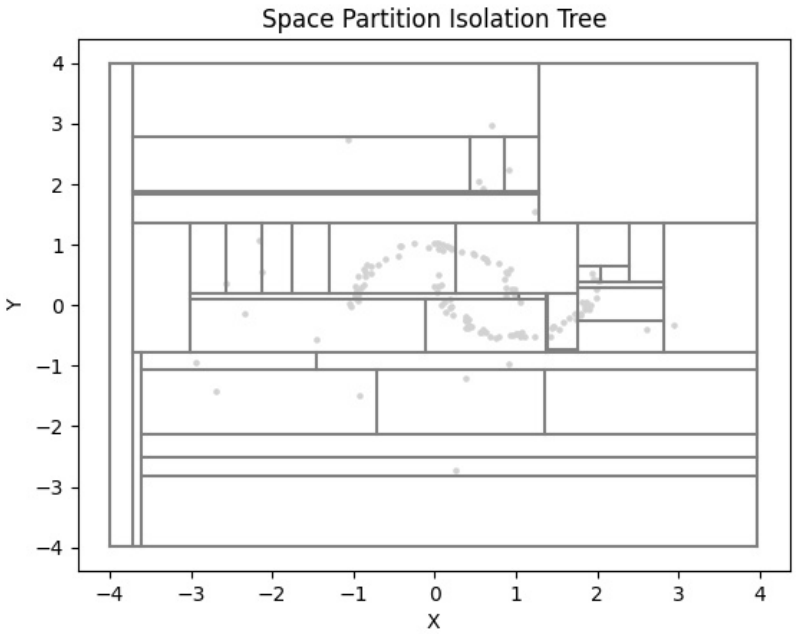
Score Aggregation

1. Sum normality scores per tree
2. Average across ensemble
3. Anomaly $= 1-$ avg. normality

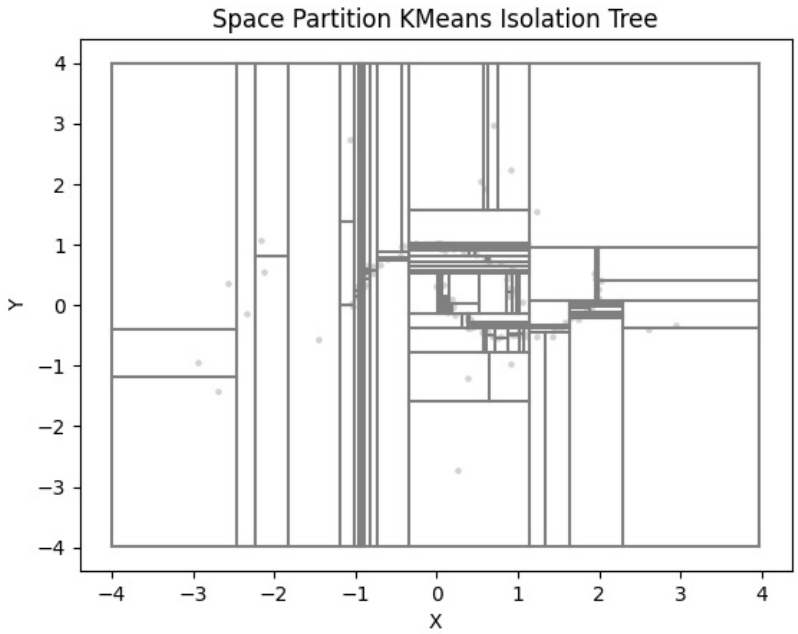
Key Difference from IF

Score depends more on **cluster membership distance** than path length

ITree vs k-Means ITree



ITree Partition on Two Moons Dataset



k-Means ITree Partition on Two Moons Dataset

Outline

Elbow Rule for Optimal k

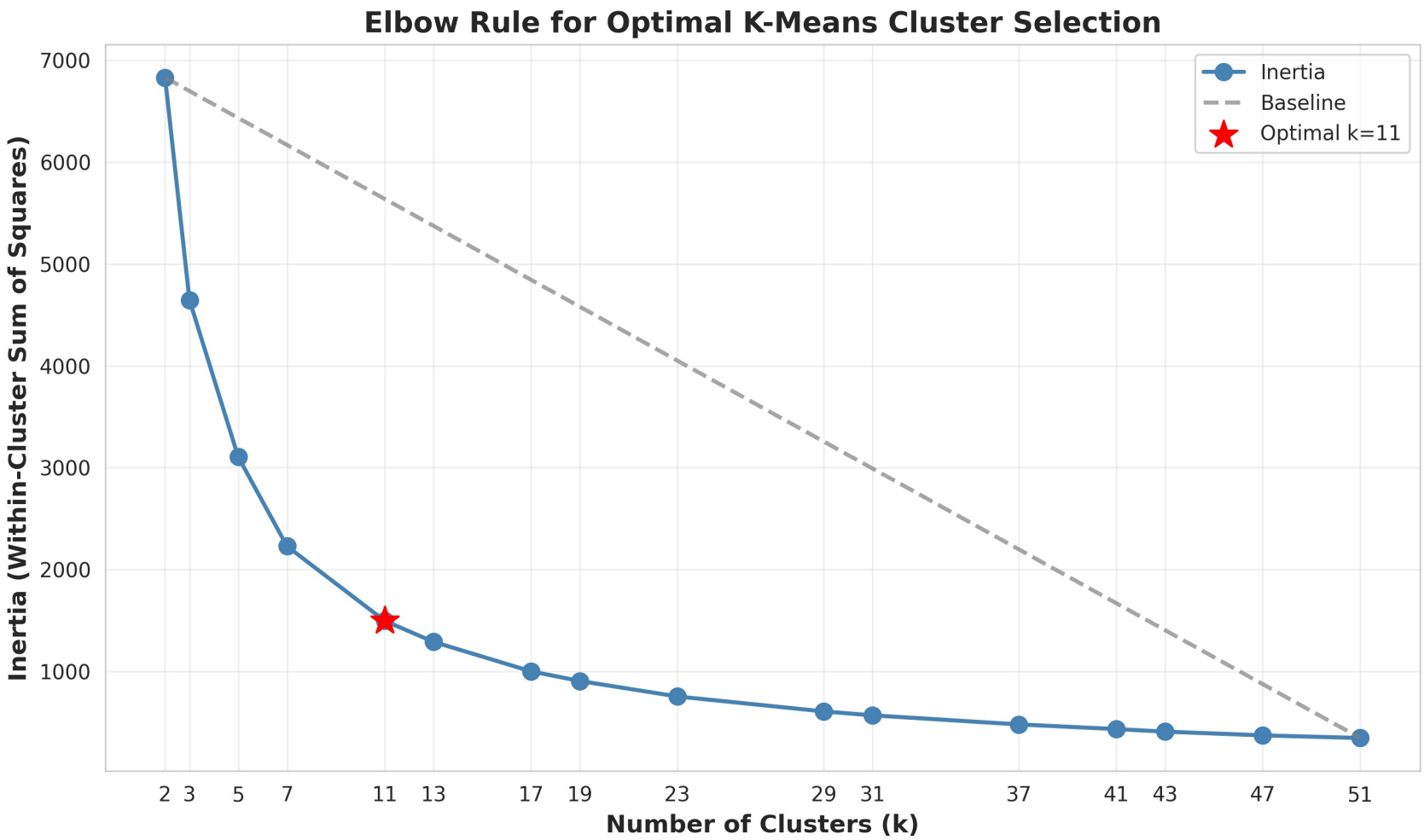
How It Works

1. Run k-Means for $k \in \{2, 3, 5, 7\}$
2. Track **WCSS** (Within-Cluster Sum of Squares)
3. Find the "elbow" point

Finding the Elbow

- Draw baseline from first to last point
- Select k with max perpendicular distance

Elbow Rule for Optimal k



Optimal k at the "elbow"

Elbow Rule for Optimal k

Cost

k-Means at **every node** = major overhead

Outline

Parallelization Strategy

Key Insight

Each tree in the ensemble is **independent** $\rightarrow$ parallelizable

Training Phase

- Distribute t trees across cores
- Independent tree construction
- Unique seeds for reproducibility

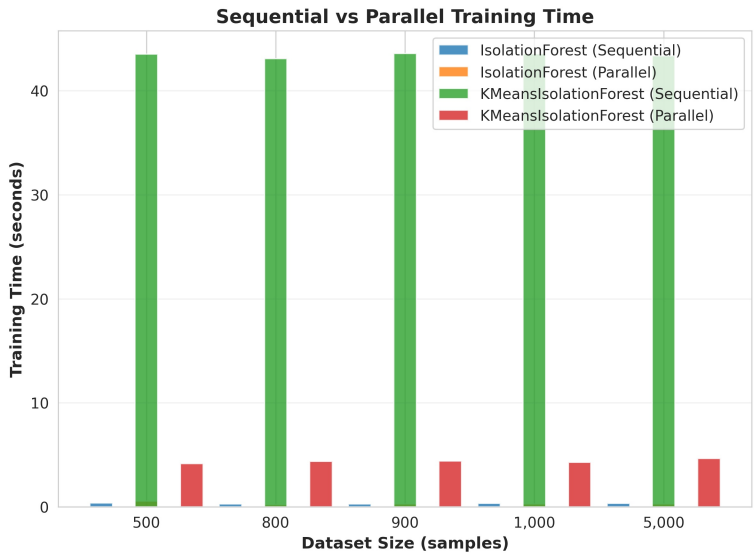
Scoring Phase

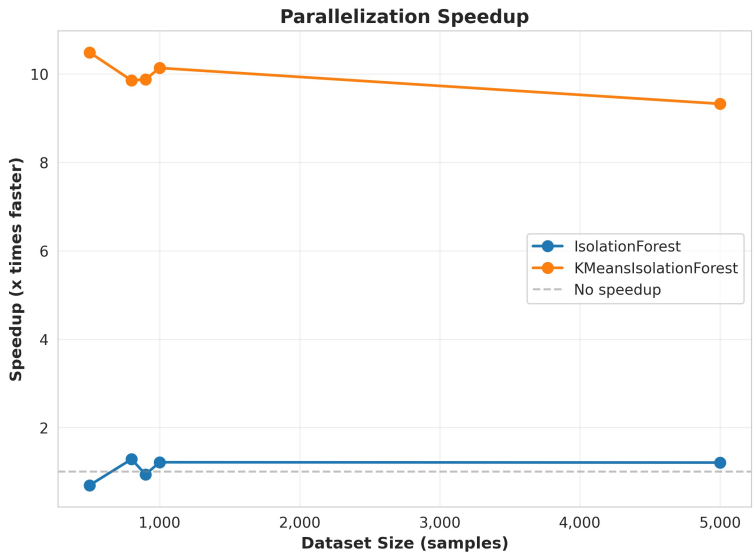
- Distribute scoring across cores
- Process tree subsets in parallel
- Aggregate scores at the end

Why Essential for kMIF?

IF: $O(1)$ node split $\rightarrow$ already fast — kMIF: $O(P \cdot k \cdot n')$ $\rightarrow$ needs parallelization

Parallelization Results





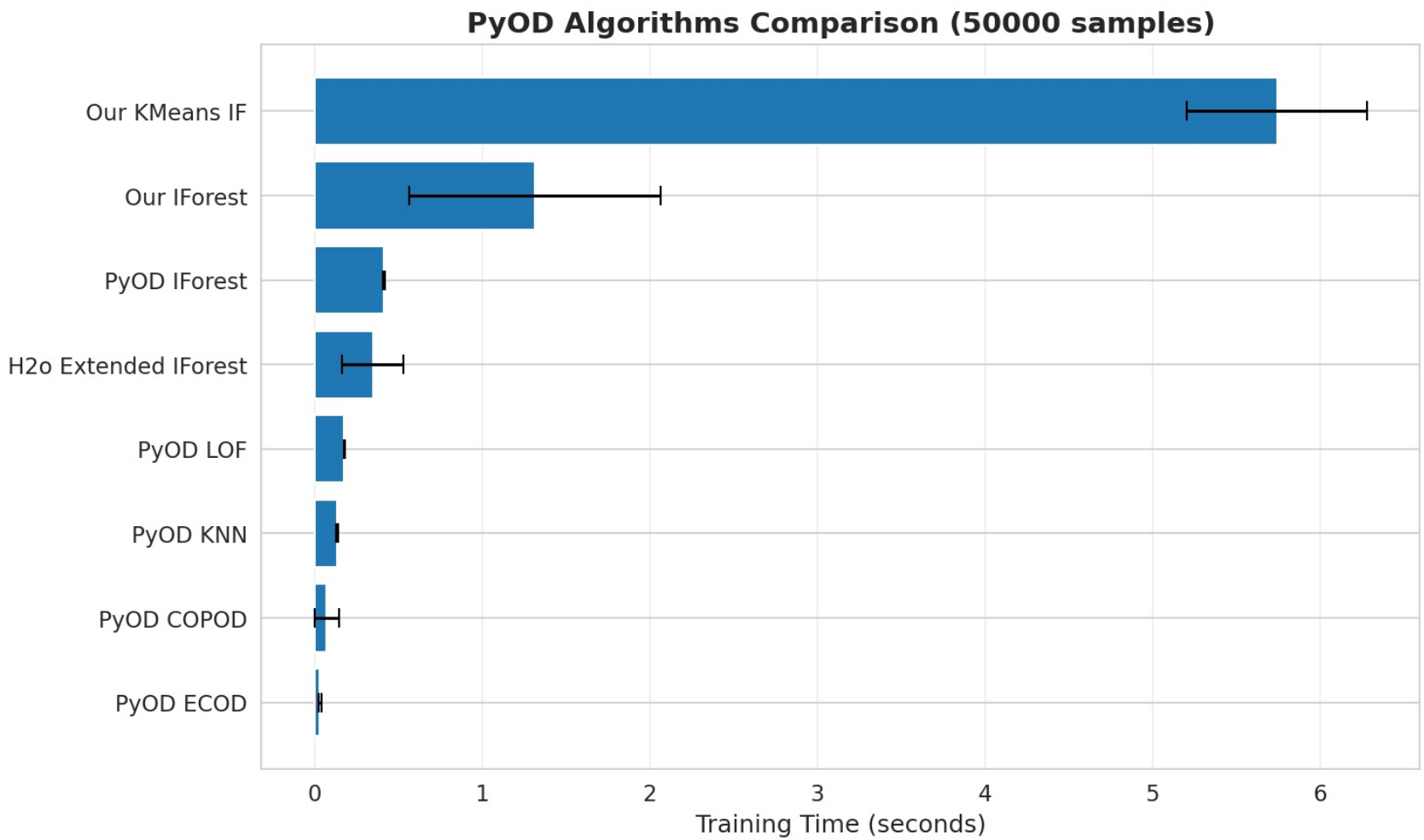
kMIF Speedup

- Sequential: > 43s
- Parallel: ~ 4s
- **9-10x speedup**

IF Speedup

- Already fast (< 0.4s)
- Minimal parallel benefit
- Overhead $>$ gains

Parallelization Comparison



Training time comparison of our custom algorithms against H2O EIF and various PyOD models on 50,000 samples.

Outline

Synthetic Dataset Experiments

Experimental Setup

- **Ensemble:** $t = 100$ trees, $n = 256$ samples, $l = 9$ depth
- **Metrics:** Accuracy, F1-Score, Precision, Recall, ROC-AUC, PR-AUC

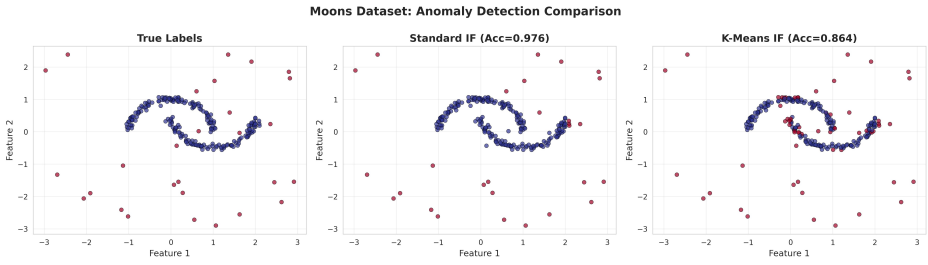
Synthetic Datasets Tested

1. **9-Rectangle Dataset** - Rectilinear, axis-aligned clusters
2. **Circles Dataset** - Concentric, non-convex clusters
3. **Moons Dataset** - Interlocking, non-convex manifolds
4. **S-Curve Dataset** - Complex, high-dimensional manifold
5. **High-Dim Synthetic** - 10D and 20D datasets

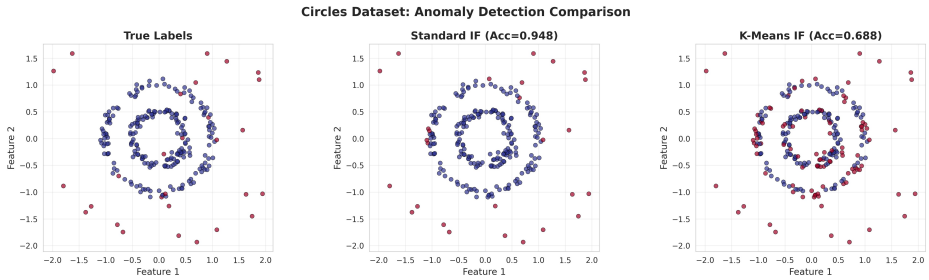
Goal

Test how each model handles different geometric challenges

Synthetic Data Results



Moons Dataset

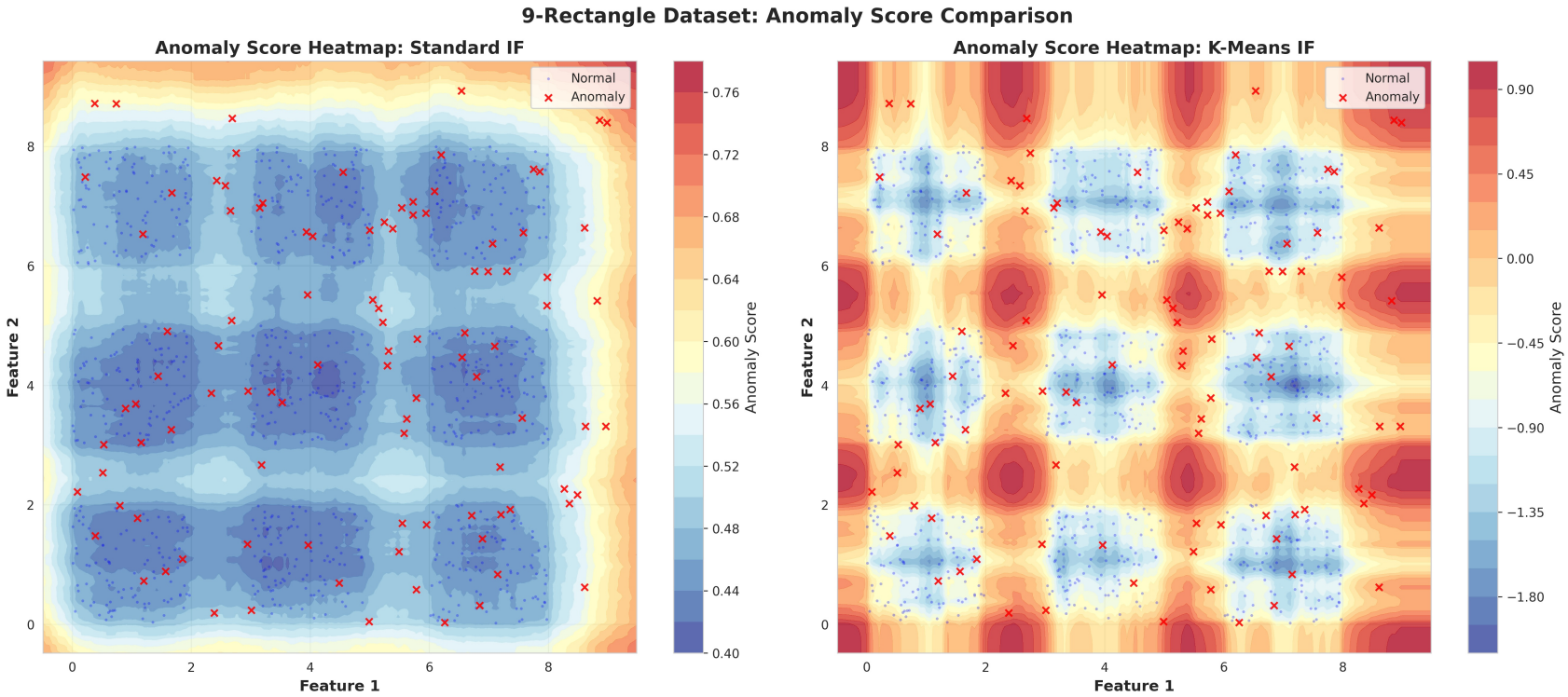


Circles Dataset

Key Findings

- **IF** outperforms kMIF on non-convex data (Moons F1: 0.88 vs 0.51)
- **kMIF** misclassifies dense core points as anomalies
- Both perform well on high-dimensional data

Synthetic Data Results



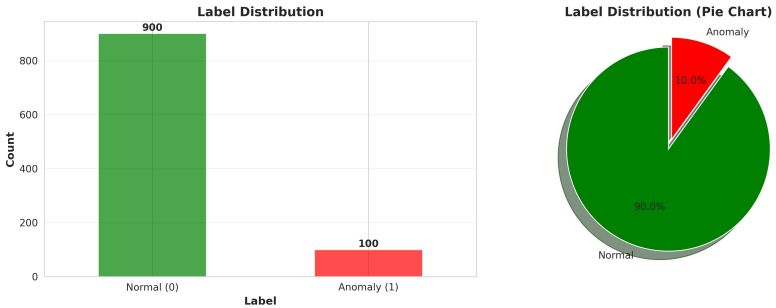
The KMeansIF method shows a clearer distinction and lower anomaly scores within the nine distinct, convex normal regions (blue), compared to the Standard IF, which exhibits more uniform scores across the feature space.

Outline

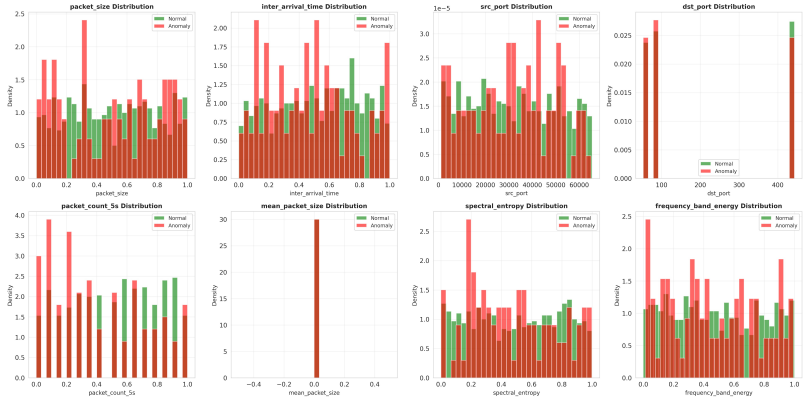
Real-World Dataset: ESNSDP

Embedded System Network Security Dataset

- Real network traffic data
- **Extreme class imbalance:** 90% Normal, 10% Anomaly
- High feature overlap between classes

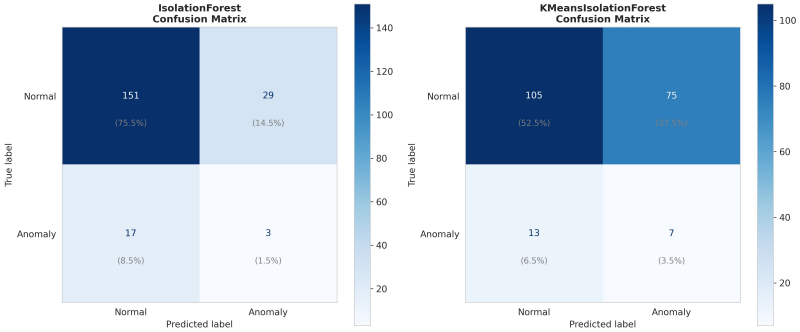


Class Distribution

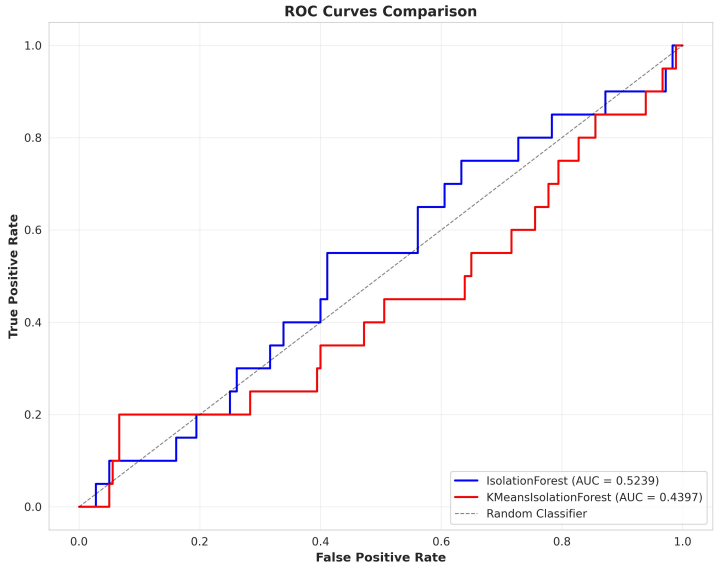


Feature Overlap

ESNSDP Results



Confusion Matrices



ROC Curves

Performance

IF: Precision=0.094, ROC-AUC=0.524 — **kMIF:** Recall=0.35, 75 FPs

Cost

kMIF: 2x slower training, 4.5x slower prediction

Outline

Conclusions & Future Work

Key Takeaways

- IF wins on non-convex data
- kMIF: complexity without gains
- Parallelization: 9-10x speedup
- IF: best for limited resources

Future Work

- Hybrid models
- Optimize kMIF overhead
- Alternative scoring
- More real-world datasets

Recommendation

Classic Isolation Forest remains optimal for general-purpose anomaly detection