



Stockholms
universitet

A Comparative Study of Unsupervised Anomaly Detection Methods: Isola- tion Forest and LOF

Li Wu

A Comparative Study of Unsupervised Anomaly Detection Methods: Isolation Forest and LOF

Li Wu*

September 2026

Abstract

This study compares two unsupervised anomaly detection methods: Isolation Forest and Local Outlier Factor (LOF). Experiments are conducted on two datasets with varying sizes, dimensionalities, and anomaly ratios. Performance is evaluated using Receiver Operating Characteristic (ROC) analysis and Precision-Recall (PR) analysis, together with an analysis of hyperparameter sensitivity and computational efficiency.

The results indicate that Isolation Forest provides a strong balance between accuracy and efficiency, making it a robust general-purpose method. In contrast, LOF performs well only when anomalies are locally separable and the data are low-dimensional, while being sensitive to parameter settings.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden.
E-mail: catrina.wuli@gmail.com. Supervisor: Chun-Biu Li.

Acknowledgements

I acknowledge the use of AI tools (DeepSeek, ChatGPT) for language polishing, coding assistance, and formatting during the writing process.

Contents

1	Introduction	4
1.1	Background	4
1.2	Objective	4
1.3	Organization	4
2	Outlier Analysis: Fundamental Concepts and Definitions	5
2.1	What Is an Outlier?	5
2.2	Characterising Outliers: Distance and Density	5
2.2.1	Distance-Based View	5
2.2.2	Density-Based View	6
2.3	Categories of Outlier Detection Algorithms	6
3	Unsupervised Machine Learning Techniques	7
3.1	Isolation Forest (iForest)	7
3.1.1	Isolation Tree (iTree)	8
3.1.2	Path Length	11
3.1.3	Binary Search Trees (BST)	12
3.1.4	From Path Length to Anomaly Score	15
3.2	Local Outlier Factor (LOF)	16
3.2.1	LOF Definitions and a Step-by-Step Example	17
3.2.2	The Impact of Parameter k	26
3.2.3	Theoretical Threshold for LOF	26
4	Experimental Results and Analysis	28
4.1	Experimental Setup	28
4.1.1	Setup	28
4.1.2	Model evaluation: ROC and PR analysis	29
4.2	Experimental Results	33
4.2.1	Results on the Mulcross Dataset	34
4.2.2	Results on the CreditCardFraud Dataset	35
4.3	Comparative Summary	37
5	Conclusion and Discussion	38

1 Introduction

1.1 Background

Anomaly detection is a fundamental task in data mining, aiming to identify patterns that deviate significantly from expected behavior. It has wide applications in areas such as fraud detection, network intrusion detection, and industrial monitoring [10]. Unsupervised anomaly detection methods, which do not require labeled data to train the models, are particularly valuable in real-world scenarios where labels are scarce or unreliable.

Among the many existing approaches, two representative categories are widely used: isolation-based tree ensemble methods (e.g., Isolation Forest) [1, 12], and density-based methods (e.g., Local Outlier Factor, LOF) [4, 11, 12]. However, their relative performance under different data characteristics such as dimensionality, sample size, and anomaly proportion remains unclear. This makes it challenging for practitioners to select the most appropriate method for a given task.

1.2 Objective

The objective of this study is to compare two unsupervised anomaly detection methods, Isolation Forest and LOF, across multiple datasets with different characteristics, while also examining their theoretical foundations. Specifically, we evaluate their performance using Receiver Operating Characteristic (ROC) and Precision-Recall (PR) analysis, and analyse their sensitivity to key hyperparameters, computational efficiency, and robustness.

1.3 Organization

The remainder of this paper is organized as follows. Section 2 introduces some fundamental concepts and definitions. Section 3 introduces the anomaly detection methods. Section 4 describes the experimental setup, including datasets, preprocessing, and parameter configurations, and presents the results on Mulcross and CreditCardFraud datasets, followed by a sensitivity analysis and interpretation. Section 5 discusses the findings, limitations, and directions for future work.

2 Outlier Analysis: Fundamental Concepts and Definitions

Before examining specific detection methods, it is essential to establish a rigorous theoretical foundation for outlier analysis. This section introduces the formal definition of outliers, the key mathematical tools used to characterize them, and a taxonomy of detection approaches that places the two methods studied in this thesis - Isolation Forest and Local Outlier Factor (LOF) - into a broader methodological framework.

2.1 What Is an Outlier?

A widely accepted definition of an outlier, originally proposed by Hawkins [9] and cited in Aggarwal's comprehensive monograph [8], states:

An outlier is an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism.

In the data mining literature, outliers are frequently referred to as *anomalies*, *abnormalities*, *discordants*, or *deviants* [8]. The key insight is that outliers are not merely random noise; instead, they often contain valuable information about unusual system behaviour, faults, fraud, or other rare events.

From a statistical perspective, a data set is assumed to be generated by one or more underlying processes. When these processes operate as expected, they produce *normal* observations. An anomaly arises when a generating process behaves unusually, thereby producing an observation that lies far from most of the data.

In this section, we follow the terminology of Aggarwal (2017) [8] and refer to unusual observations as outliers. Throughout the rest of this thesis, we use the more common machine learning term anomaly (and anomaly detection) to denote the same concept. The two terms are used interchangeably.

2.2 Characterising Outliers: Distance and Density

Two fundamental principles form the basis of most outlier detection techniques: distance-based and density-based approaches [8, 11]. These principles directly motivate the methods compared in this thesis.

2.2.1 Distance-Based View

The simplest way to identify an outlier is to measure how far it lies from the majority of the data. If a point is far from other points, it is likely to be an outlier. However, the concept of "far" must be carefully defined, especially in high-dimensional spaces.

Euclidean Distance. The most common distance measure is the Euclidean distance. For two points $\mathbf{a}$ and $\mathbf{b}$ in p -dimensional space, it is defined as

$$d(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{j=1}^p (a_j - b_j)^2}. \quad (1)$$

Mahalanobis Distance. Euclidean distance treats all dimensions equally and ignores correlations among features. The Mahalanobis distance addresses this limitation by taking into account the covariance structure of the data [8]. Given a data set with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, the Mahalanobis distance from a point $\mathbf{x}$ to the data centre is defined as

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}. \quad (2)$$

In practice, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are estimated from the training set. The Mahalanobis distance effectively standardises each dimension by its variance and linearly decorrelates the features, transforming the original data into a space where distances are measured in units of standard deviation along principal directions. A point with a large Mahalanobis distance lies far from the data centre, accounting for the natural shape of the distribution, and can be flagged as an outlier, similarly to a point with a high anomaly score. However, the Mahalanobis distance is limited by its assumption of linear relationships and is sensitive to the quality of the covariance estimate.

Both Euclidean and Mahalanobis distances are global distance measures that quantify how far each data point is from a global reference, such as the mean vector or the underlying data distribution.

2.2.2 Density-Based View

The fundamental intuition behind density-based methods is simple: points that are close to each other lie in a dense region, while points that are far apart lie in a sparse region. In other words, density is inversely related to distance – shorter distances indicate higher density, and longer distances indicate lower density.

A natural way to apply this intuition is through a global density estimate: for a given point, one measures the average distance to its neighbours, or counts how many points fall within a fixed radius. Points lying far from the majority receive a low density estimate and can be flagged as outliers. This is the essence of global density methods, which evaluate each point against the entire dataset.

However, global methods rest on a fundamental assumption: that density is roughly uniform across the data space. When clusters of different densities coexist, this assumption breaks down. A point that is normal within a sparse cluster may appear anomalous by global standards, while a genuine local outlier near a dense cluster may go undetected.

To address this limitation, density must be redefined as a local concept. A point should be judged not against the entire dataset, but against its immediate neighbourhood. The Local Outlier Factor (LOF) operationalises this idea by comparing the local density of a point with the local densities of its neighbours. This allows LOF to detect both global outliers – points far from all clusters – and local outliers – points that deviate from their immediate neighbourhood even when they are not globally distant from the data centre.

Thus, while global density methods ask "is this point far from the majority?", LOF refines the question to: "is this point sparse relative to its neighbours?"

2.3 Categories of Outlier Detection Algorithms

Aggarwals monograph [8] organises outlier detection algorithms into several broad categories, each reflecting a different mathematical principle.

- **Probabilistic Models:** These methods assume a parametric form for the data distribution (e.g., Gaussian) and identify points that have very low probability under the fitted model as outliers. Extreme value analysis belongs to this class.

- **Linear Models:** Techniques such as PCA and regression analyse linear correlations and treat points with large residuals as outliers.
- **Proximity-Based Models:** This family includes clustering methods, nearest-neighbor methods and density-based methods. Points are considered outliers if their neighbourhood is sparsely populated or if their distance to neighbours is exceptionally large. LOF falls into this category.
- **High-Dimensional (Subspace) Methods:** These methods address the curse of dimensionality by searching for subspaces in which a points projection onto a particular low-dimensional subspace lies far from the bulk of the data in that subspace.
- **Ensemble Methods:** By combining the outputs of multiple algorithms, ensemble methods improve robustness and stability. Isolation Forest can be viewed as an ensemble of random partitioning trees.
- **Neural Network and Deep Learning Models:** Autoencoders and other neural architectures learn flexible nonlinear representations of normal data and identify outliers via reconstruction error.

The two methods selected for this study represent two distinct categories:

- **Isolation Forest** is an ensemble method based on random recursive partitioning. Although not a classical proximity-based model, its path length can be interpreted as a reversed proximity measure: shorter paths indicate less similarity to the majority.
- **LOF** is a proximity-based method rooted in local density estimation.

3 Unsupervised Machine Learning Techniques

In an unsupervised learning setting, also known as "learning without a teacher", we are given a collection of N observations $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}$, which are realisations of p -dimensional random vectors $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \dots, \mathbf{X}^{(N)}$ each drawn from the same distribution as $\mathbf{X} = (X_1, \dots, X_p)$ with joint probability density $\Pr(\mathbf{X})$ [3].

The objective is to infer the underlying characteristics of the joint probability distribution of $\mathbf{X}$. In this context, the components $X_1, \dots, X_p$ are often referred to as *features*, *variables*, or *attributes*, depending on the application. And each observation is denoted $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \dots, x_p^{(i)})$.

This section presents the theoretical foundations of two unsupervised learning techniques employed in anomaly detection: Isolation Forest (tree-based ensemble) and Local Outlier Factor (LOF a density-based method). These methods were selected because they are based on fundamentally different mathematical principles and are among the most widely used techniques in anomaly detection [11, 12].

Throughout this section, we follow the notation conventions of the original papers for each method. In particular, iForest (Section 3.1) use the notation x for a data point, while LOF (Section 3.2) uses $o, p \in D$ for a object (or data point).

3.1 Isolation Forest (iForest)

Isolation Forest (iForest), proposed by Liu et al. [1], detects anomalies by isolating them rather than modeling normal behavior. It leverages the principle that anomalies are few and different, making them easier to separate through random partitioning.

Algorithm 1 presents the overall procedure for constructing the iForest. The algorithm takes three inputs: the input data X , the number of trees t , and the subsampling size ψ . For each tree, it randomly draws a subsample X' of size ψ without replacement from the input data X , and grows an isolation tree. The resulting set of isolation trees is then stored as the final forest.

Algorithm 1 Isolation Forest (iForest)

Require: Input: X - input data, t - number of trees, ψ - subsampling size

Ensure: Output: A set of t iTrees, denoted as *Forest*

```

1: Forest  $\leftarrow \emptyset$ 
2: Set height limit  $l \leftarrow \lceil \log_2 \psi \rceil$ 
3: for  $i \leftarrow 1$  to  $t$  do
4:    $X' \leftarrow \text{Sample}(X, \psi)$ 
5:   Forest  $\leftarrow \text{Forest} \cup \text{iTree}(X', 0, l)$ 
6: end for
7: return Forest

```

There are only two parameters in this method: the subsampling size ψ and the number of trees t .

Regarding the choice of the subsampling size ψ , unlike many conventional machine learning methods, where larger sample sizes are generally preferred, the iForest tends to perform better with relatively small subsamples. The default value is set to $\psi = \min(256, N)$, where N is the total number of samples.

This choice is motivated by two main factors. First, empirical evidence shows that the detection performance stabilizes once ψ exceeds 256, while the computational cost grows with ψ . Second, a smaller subsample size helps reduce the effects of *swamping* where normal instances are misidentified to anomalies and *masking* where anomalies conceal each other when forming dense clusters, two common phenomena in anomaly detection. As noted in the original paper, “Note that both swamping and masking are a result of too many data for the purpose of anomaly detection[1].”

The choice of the number of trees t which controls the ensemble size, will be discussed in a subsequent section.

3.1.1 Isolation Tree (iTree)

The previous section described the high-level procedure of building a forest. We now turn our attention to the internal structure of a single isolation tree, as its construction directly determines the path length, the core metric for anomaly detection which will be discussed in a subsequent section. The formal definition of an iTree is given as follows:

Definition: Isolation Tree. Let T be a node of an isolation tree. T is either an *external-node* (leaf) with no child, or an *internal-node* with one test and exactly two daughter nodes (T_l, T_r) . A test consists of an attribute q and a split value p such that the test $q < p$ divides data points into T_l and T_r .

Figure 1 shows a iTree diagram on a synthetic 2D dataset.

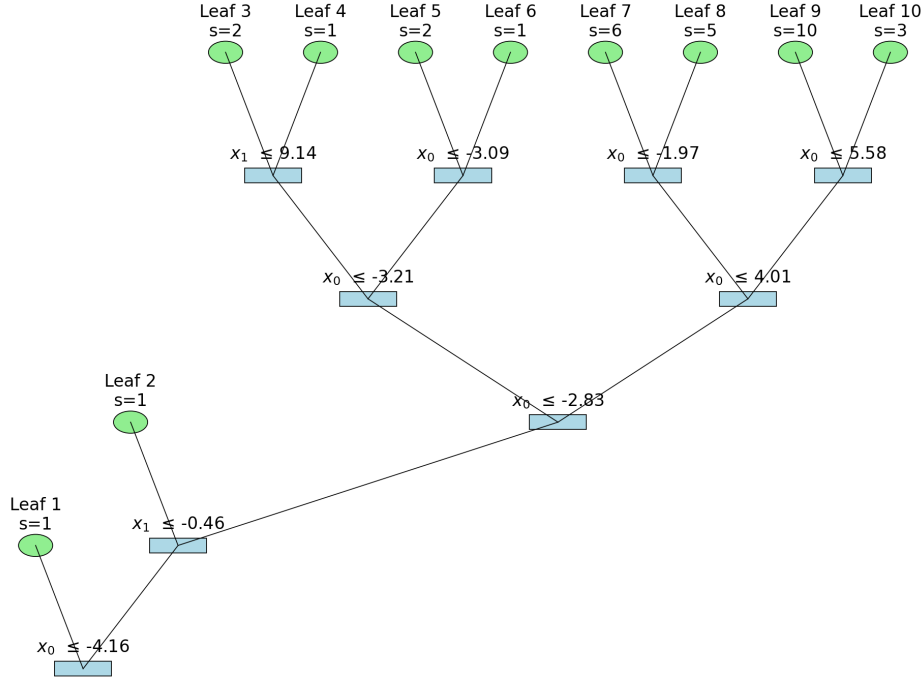


Figure 1: Structure of an iTree: The root is at the bottom, and branches extend upward. Each internal node (light blue rectangle) contains a split condition and the split value. For example the root (light blue rectangle) contains a split condition of the form $x_0 \leq -4.16$, where x_0 denotes the feature 0 and -4.16 is the split value. And each leaf node (light green circle) is numbered and labeled with the number of data points that fall into that subspace (e.g., Leaf 1 $s=1$ means the first leaf contains only one data point).

To grow an iTree, the following procedure is applied. For each isolation tree, we first draw a random subsample of size ψ from the full dataset X without replacement, ensuring that each observation appears at most once within a given subsample. The same subsampling step is performed to build every tree in the forest.

The procedure for growing an iTree Let $x_{1:\psi} = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\psi)}\}$ be a subsample of size ψ , where each point follows a p -dimensional joint distribution. An iTree is built by recursively dividing $x_{1:\psi}$ by randomly selecting a feature q and a split value d which is uniformly chosen from the interval

$$\left[\min(x_q^{(1)}, \dots, x_q^{(\psi)}), \max(x_q^{(1)}, \dots, x_q^{(\psi)}) \right],$$

until one of the following conditions is met:

1. The node contains a single data point;
2. All data points at the node have identical feature values;
3. The tree reaches a predefined height limit l . This height limit is set by the sampling size ψ : $l = \lceil \log_2 \psi \rceil$.

The height limit is fixed at $l = \lceil \log_2 \psi \rceil$ for two primary reasons. First, because each internal node performs a binary split in isolation tree, a perfectly balanced tree ((i.e., a tree where all leaves are at the same depth)) of depth l has at most 2^l leaf nodes. Isolating ψ samples requires leaf nodes $2^l \geq \psi$, which directly gives $l \geq \log_2 \psi$. Second, to minimize computational cost while maintaining sufficient isolation capability, the algorithm sets the height limit precisely to this bound, i.e., $l = \lceil \log_2 \psi \rceil$.

Algorithm 2 presents the procedure for growing the iTree.

Algorithm 2 Isolation Tree (iTree)

Require: X' - data, e - current depth, l - height limit

Ensure: An iTree node

```

1: if  $e \geq l$  or  $|X'| \leq 1$  then
2:   return EXNODE $\{|X'|\}$ 
3: else
4:   Randomly select a feature  $q$  from  $X'$ 
5:   Randomly select a split point  $p$  between the min and max value of  $q$  in  $X'$ 
6:    $X'_l \leftarrow \text{filter}(X, q < p)$ 
7:    $X'_r \leftarrow \text{filter}(X, q \geq p)$ 
8:   return INNODE $\{q, p, \text{iTREE}(X'_l, e + 1, l), \text{iTREE}(X'_r, e + 1, l)\}$ 
9: end if

```

Figure 2 illustrates the first two splits performed in the iTree shown in Figure 1. The iTree is constructed by first selecting Feature 0 with a split value of "-4.16" as the split condition, followed by Feature 1 with a split value of "-0.46" (as shown in Figure 1). The figure also displays two normal clusters as blue points and the outlier as a red cross. Points that are consistently isolated after only a few splits across the iTrees are likely to be anomalies.

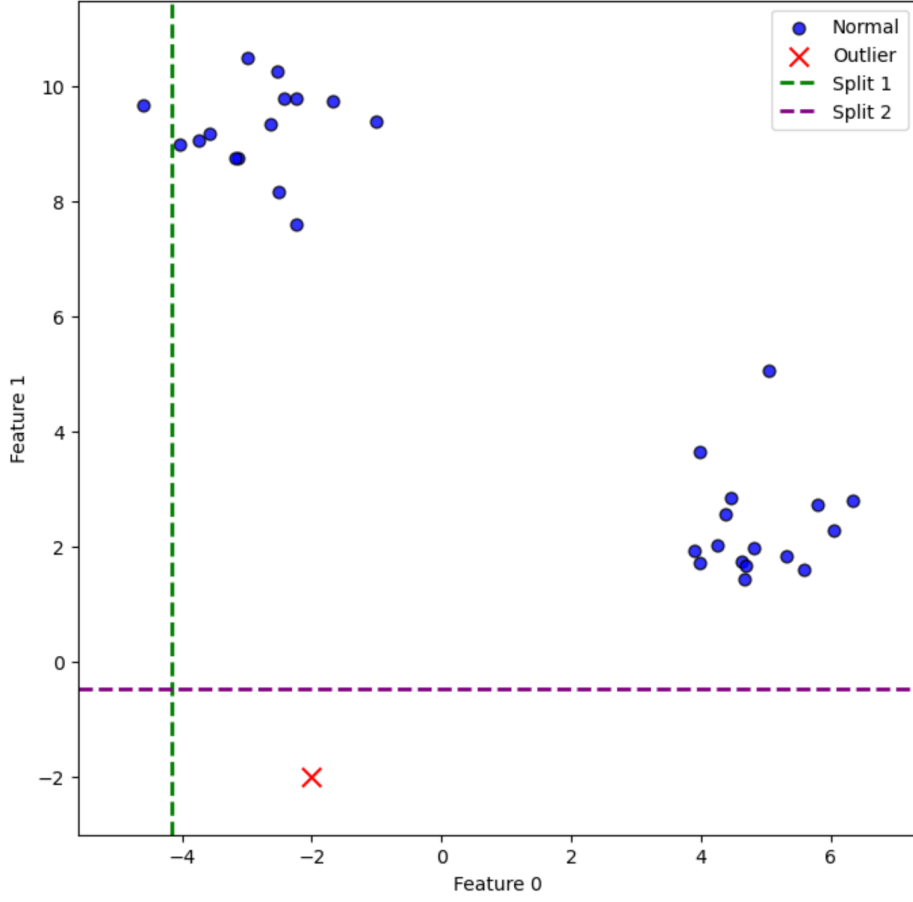


Figure 2: First two splits of an isolation tree: The scatter plot (axes: Feature 0 horizontal, Feature 1 vertical) contains 32 synthetic data points (blue dots), including two dense clusters and an outlier (red cross). The green vertical dashed line (Split 1) and purple horizontal dashed line (Split 2) illustrate the first two random axis-aligned splits. The legend (upper right) identifies the split order. In this subsample, the outlier is already isolated after two splits.

3.1.2 Path Length

The previous sections outlined the construction procedures for an iForest and its constituent iTrees. We now turn our attention to the path length of a given point within a iTree, a quantity that is central to the computation of the anomaly score. The formal definition is given as follows:

Definition (Path Length). The path length $h(x)$ of a point x is the number of edges x traverses in an iTree from the root node until it reaches an external node.

We now turn to the statistical estimation of the path length. For a fixed test point x , the splitting process in each iTree is randomised; consequently, the path length $h(x)$ is a random variable, where the randomness arises solely from the stochastic construction of the tree. To detect anomalies, we

need to estimate its expected value $\mu(x) = \mathbb{E}[h(x)]$, and quantify the uncertainty of this estimate using the t iTrees in the forest.

To establish the statistical properties of the ensemble average, we first clarify the independence structure of the generated iTrees. In the Isolation Forest algorithm, the construction of each iTree is governed by an independent random seed, and the subsampling procedure is re-initialized for every tree. Hence, the tree structures

$$T_1, T_2, \dots, T_t$$

constitute a sequence of independent and identically distributed (i.i.d.) random variables defined on the space of all possible iTrees.

For this fixed x , once the random structure of a tree T_j is determined, the corresponding path length $h_j(x)$ is uniquely determined. Thus, $h_j(x)$ is a measurable function of T_j , and by the standard result that measurable transformations preserve independence [16, Chapter 2, Theorem 10.4], the path lengths

$$h_1(x), h_2(x), \dots, h_t(x)$$

are i.i.d random variables.

Let $\mu(x) = \mathbb{E}[h(x)]$ and $\sigma^2(x) = \text{Var}[h(x)]$ denote the mean and variance of this random variable, respectively.

To estimate $\mu(x)$, we use the sample mean

$$\bar{h}(x) = \frac{1}{t} \sum_{j=1}^t h_j(x), \quad (3)$$

which is an unbiased estimator of $\mu(x)$. Its expectation and variance are given by

$$\mathbb{E}[\bar{h}(x)] = \mu(x), \quad \text{Var}[\bar{h}(x)] = \frac{\sigma^2(x)}{t}. \quad (4)$$

This shows that increasing the number of trees t reduces the variance of the sample mean, $\lim_{t \rightarrow \infty} \text{Var}[\bar{h}(x)] = 0$.

It is worth noting that the algorithm imposes a height limit $l = \lceil \log_2 \psi \rceil$ on each iTree, which bounds the path length by $0 \leq h(x) \leq l$. This boundedness allows us to quantify the convergence rate of the ensemble average using Hoeffding's inequality [16, [Chapter 3, Theorem 1.3]. For any $\varepsilon > 0$,

$$\Pr(|\bar{h}(x) - \mathbb{E}[h(x)]| \geq \varepsilon) \leq 2 \exp\left(-\frac{2t\varepsilon^2}{l^2}\right). \quad (5)$$

This bound decays exponentially in t , implying that the ensemble average converges rapidly to its expectation. In practice, this justifies the default choice of $t = 100$ trees, as further increasing the number of trees yields negligible improvements in estimation accuracy while adding computational cost.

3.1.3 Binary Search Trees (BST)

A key insight of Isolation Forest is that an iTTree shares structural properties with a BST. Before presenting the formula, we briefly introduce the relevant BST definitions and theorems (proofs omitted; see [2, Chapter 10]). However, we stress that our adoption of BST theory is limited to the normalisation constant $c(n)$, the average path length of an external node, which serves to normalise path lengths; all other BST-specific details are orthogonal to this work.

In the following BST derivation, we denote by n the size of the subsample used to build a single isolation tree (i.e., $n = \psi$), consistent with the standard notation in binary search tree theory.

Definition: Binary Search Tree. A binary search tree T is a finite totally ordered set of keys. Either the set is empty, $T = \emptyset$; or the set consists of a root r and exactly two binary search trees T_L and T_R , such that the following properties are satisfied:

1. All the keys contained in left subtree, T_L , are less than r .
2. All the keys contained in the right subtree, T_R , are greater than r .

Under this recursive definition, **internal nodes** store a key and have at least one child (i.e., non-empty subtrees), while **external nodes** (empty subtrees) are null pointers that contain no keys and represent unsuccessful search paths, as illustrated in Figure 3.

Figure 3 illustrates a BST containing the ordered set of keys A through H. Internal nodes(keys) are shown as circles, while external nodes (empty subtrees) are shown as grey boxes. Since the nodes of the tree carry alphabetic rather than numeric keys, the ordering of the keys is alphabetic. That is, all the keys in the left subtree of a given node precede alphabetically the root of the node, and all the keys in the right subtree of a given node follow alphabetically the root of that node. The empty trees are shown explicitly as boxes.

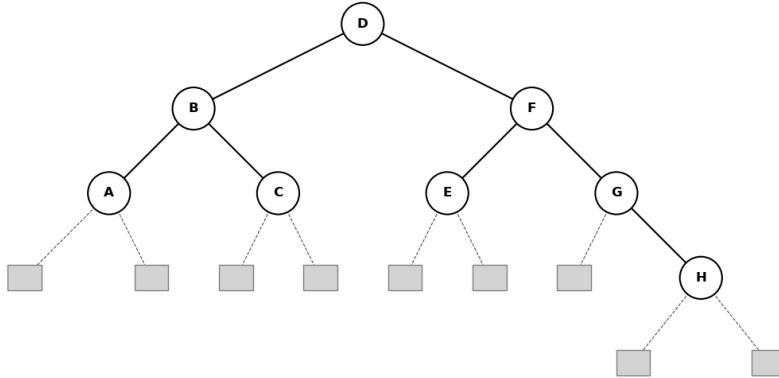


Figure 3: A binary search tree containing the keys A through H: Internal nodes (keys A, B, C, D, E, F, G, H) are drawn as blackbordered white circles. External nodes (empty subtrees) are drawn as lightgray filled rectangles with black borders. A BST with n internal nodes always has exactly $n + 1$ external nodes (empty child positions).

Definition: Depth of i^{th} internal node d_i : the number of edges on the path from the root to internal node i (the root has depth 0). When a search is successful, exactly $d_i + 1$ internal nodes are visited.

Definition: Internal path length I : sum of the depths of all internal nodes $\sum_{i=1}^n d_i$.

Definition: External path length E : sum of the depths of all external nodes.

For a BST with n internal nodes, there are exactly $n + 1$ external nodes, shown as the light-gray filled rectangles in Figure 3.

Theorem 1. *Consider a binary tree T with n internal nodes and internal path length I . Then the external path length E of T is given by:*

$$E = I + 2n.$$

In a BST, we store n distinct keys (which can be numbers or other comparable values) in a tree such that all keys in the left subtree of a node are smaller than the nodes key, and all keys in the right subtree are larger. Searching for a value that is *not* in the tree (an unsuccessful search) ends at an external node - the empty node with no child.

In an iTree, the recursive partitioning continues until external node contains a single point, mirroring the BST structure where each split generates a node and each terminal leaf corresponds to an external node (as illustrated in Figure 3). Consequently, the average path length of an external node - which represents an unsuccessful search in a BST - is the average path length of an external node in an iTree. This equivalence enables established results from BST theory to be directly applied to the iForest framework.

Successful Search For a successful search, the process terminates at an internal node. The average internal path length is given by

$$I(n) = 2(n+1)H_n - 4n, \quad (6)$$

where $H_n = \sum_{k=1}^n \frac{1}{k}$ denotes the n -th harmonic number. We use this result without proof and refer the reader to [2], Section “Search Trees”, for a detailed derivation.

Unsuccessful Search By Theorem 1, $E = I + 2n$ holds for any binary tree. Taking average, the average external path length becomes:

$$E = I + 2n = 2(n+1)H_n - 2n. \quad (7)$$

Since there are $n + 1$ external nodes, the average path length of an external node is:

$$c_{\text{BST}}(n) = \frac{E}{n+1} = 2H_n - \frac{2n}{n+1}. \quad (8)$$

Relationship with iForest. iForest adopts a slightly different convention (counting root depth as 1 instead of 0), which leads to the equivalent form:

$$c(n) = 2H_{n-1} - \frac{2(n-1)}{n}. \quad (9)$$

It is important to note that this BST-based normalisation constant $c(n)$ is derived under the idealised assumption that the tree is fully grown. In practice, iForest imposes a height limit $l = \lceil \log_2 \psi \rceil$, which truncates the path lengths. However, as shown in the original iForest paper, this approximation is sufficient for normalisation purposes, because the anomaly score remains monotonic with respect to path length and the threshold 0.5 provides a stable reference across different n .

3.1.4 From Path Length to Anomaly Score

Section 3.1.2 and Section 3.1.3 introduce two key quantities: the path length $h(x)$ for a given point x (a fixed test instance), and $c(n)$ the average path length of an external node given the sample size n (Eq.9). The underlying intuition, as discussed in previous sections, is that anomalies, being rare and isolated, require fewer splits and thus yield a path length smaller than $c(n)$. In contrast, normal points demand more splits, leading to path lengths close to or greater than $c(n)$. The anomaly score is precisely designed based on this contrast.

To obtain a standardised and interpretable score bounded in $(0, 1]$, iForest transforms the normalised expected path length via the monotonic mapping $z \mapsto 2^{-z}$, defining the theoretical anomaly score as

$$s(\mathbf{x}, n) = 2^{-\mathbb{E}[h(\mathbf{x})]/c(n)}. \quad (10)$$

As established in Section 3.1.2, the sample mean $\bar{h}(\mathbf{x})$ is an unbiased estimator of $\mathbb{E}[h(\mathbf{x})]$. Replacing the expectation by this empirical estimate yields the practical, computable score

$$\hat{s}(\mathbf{x}, n) = 2^{-\bar{h}(\mathbf{x})/c(n)}. \quad (11)$$

The exponential form 2^{-z} maps the non-negative argument $z = \mathbb{E}[h(x)]/c(n)$ onto the interval $(0, 1]$. Moreover, it is strictly decreasing: a shorter path length (more anomalous) yields a larger score. In Equation(10):

- When $\mathbb{E}[h(x)] \rightarrow l$, $s \rightarrow 0$.
- When $\mathbb{E}[h(x)] \rightarrow c(n)$, $s \rightarrow 0.5$;
- When $\mathbb{E}[h(x)] \rightarrow 0$, $s \rightarrow 1$;

The choice of base 2 is not arbitrary. In an iTree, each split is a binary decision that halves the remaining data in expectation under uniform partitioning. Consequently, the probability that a point survives t splits is roughly 2^{-t} . By using base 2, the score s directly reflects this exponential decay: a point with $\mathbb{E}[h(x)] = c(n)$ receives a score of 0.5, which serves as a natural decision threshold.

And normalizing by $c(n)$ makes the score independent of the sample size n , allowing direct comparison across datasets with different n .

Figure 4 illustrates the relationship between $s(x, n)$ and $\mathbb{E}[h(x)]$.

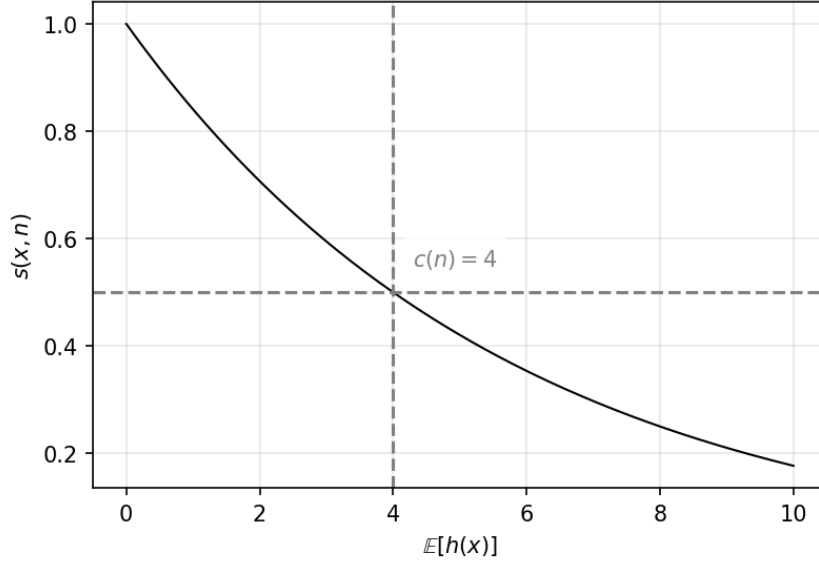


Figure 4: Relationship between $\mathbb{E}[h(x)]$ and $s(x, n)$: The figure has x-axis labeled $\mathbb{E}[h(x)]$ and y-axis labeled $s(x, n)$. A solid black curve depicts the function $s(x, n)$. The horizontal and vertical dashed lines intersect at $(c(n), 0.5)$, marking the decision boundary where $s = 0.5$ and $\mathbb{E}[h(x)] = c(n)$. Points with expected path lengths smaller than $c(n)$ yield anomaly scores above 0.5, indicating anomalies. (Adapted from [1])

3.2 Local Outlier Factor (LOF)

Local Outlier Factor (LOF), introduced in [4], is a density-based method that quantifies the degree to which an object (point) is an outlier by comparing its local density with that of its neighborhood.

To illustrate this, consider the example in Figure 5. There are two clusters, C_1 (blue) and C_2 (green), and two additional objects, o_1 and o_2 . A purely global method, such as one based on Euclidean distance, would likely only flag o_2 as an outlier, as it is far from all other points. The object o_1 , however, would be missed because, from a global perspective, it appears to be a legitimate part of cluster C_1 . Density-based clustering algorithms like DBSCAN, which rely on a global density threshold, face a similar challenge and may fail to detect such local outliers. LOF is specifically designed to address this limitation by considering not just global, but also **local** information. It introduces a new density-like measure—the **local reachability density**—to capture the degree to which a point deviates from its local neighborhood. By comparing an object’s local density to that of its neighbors, LOF can effectively detect local outliers like o_1 , which would be missed by global approaches.

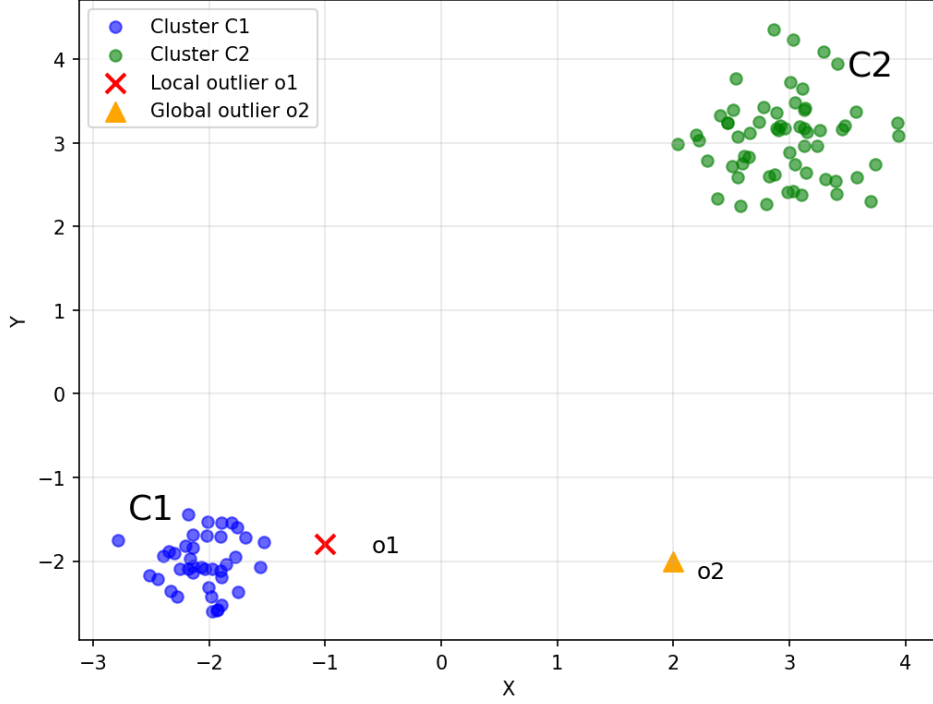


Figure 5: Local Outlier Factor (LOF) on simulated 2D data with local and global outliers: Two clusters are shown in the scatter plot: a blue cluster C_1 , and a green cluster C_2 . LOF successfully assigns high outlier scores to both the local outlier o_1 (which is close to C_1 but in a low-density region) and the global outlier o_2 . (Adapted from [4]).

3.2.1 LOF Definitions and a Step-by-Step Example

This section introduces the LOF definitions from [4], proceeding through the hierarchy of k -distance, reachability distance, local reachability density, and finally the LOF score. To ground these concepts, we walk through a concrete example on a synthetic dataset (Figure 6), showing the numerical computation at each level. The goal is to make the LOF mechanism clear.

Definition 1: (k -distance of an object o)

Given an object o in a dataset D , for any positive integer k , the k -distance of object o in a dataset D , denoted as $k\text{-dist}(o)$, is defined as the distance $d(o, p)$ between o and an object $p \in D$ such that:

- (i) for at least k objects $p' \in D \setminus \{o\}$ it holds that $d(o, p') \leq d(o, p)$, and
- (ii) for at most $k - 1$ objects $p' \in D \setminus \{o\}$ it holds that $d(o, p') < d(o, p)$.

In other words, the k -distance of an object o is the smallest radius centered at o that contains at least k other points from D (counting ties at the boundary).

we will illustrate this using an example. For notational simplicity, we write the k -distance of the object o as $k\text{-dist}(o)$, where $k \in \mathbb{Z}^+$ (e.g., for $k = 2$, we write $2\text{-dist}(o)$).

Consider the dataset D consisting of six objects as shown in the scatter plot in Figure 6. The dataset is intentionally designed with three objects exactly at distance 1 from o to demonstrate how LOF handles ties (multiple neighbors at the same distance). In continuous data, exact ties have zero probability, but they may arise in discrete or quantized settings, and the algorithm's treatment of them is an important conceptual point.

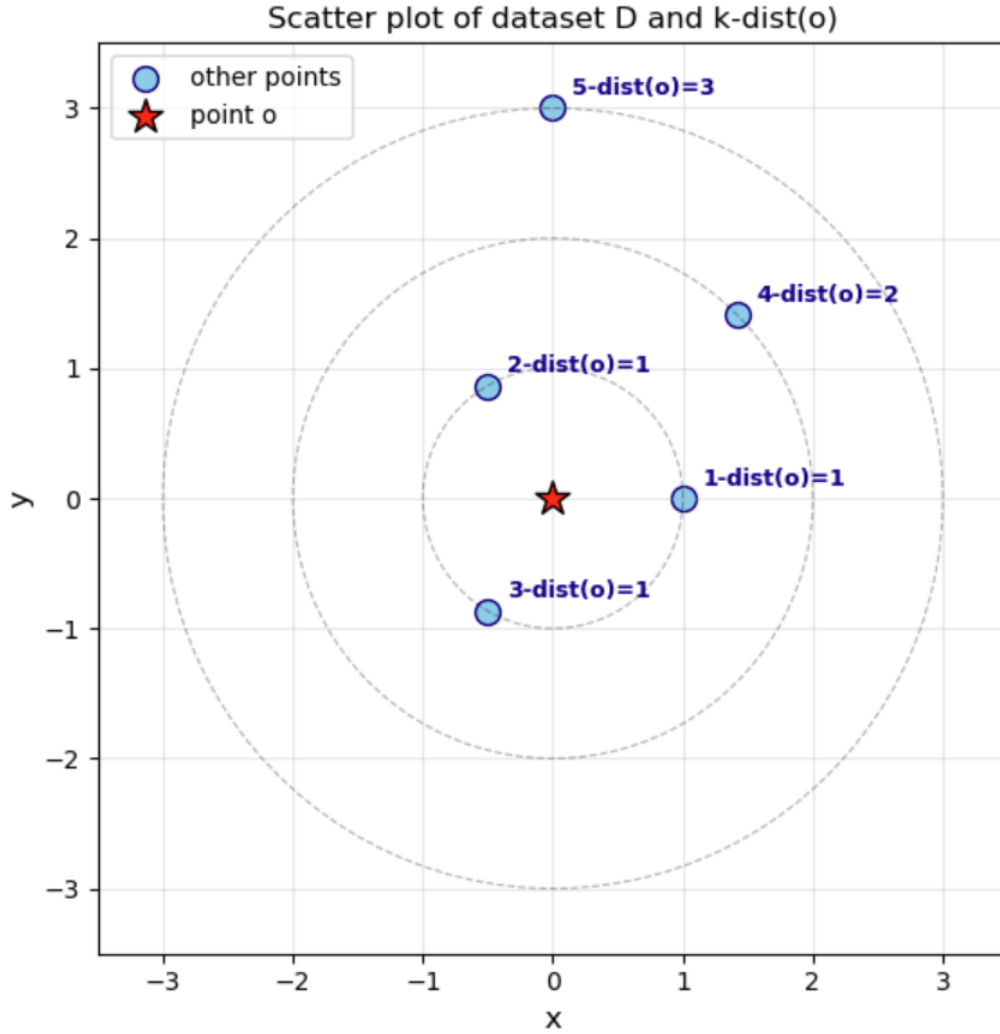


Figure 6: Illustration of k -dist(o) for the dataset D . Blue objects are other data points. Circles centered at o with radii 1, 2, 3 are drawn as dashed lines. For this dataset, 1-dist(o) = 2-dist(o) = 3-dist(o) = 1, 4-dist(o) = 2, and 5-dist(o) = 3.

For a fixed $k = 2$, each object in dataset D has a well-defined 2-distance, corresponding to the radius of the circle centered at that object that encloses at least 2 other objects, as illustrated in Figure 7.

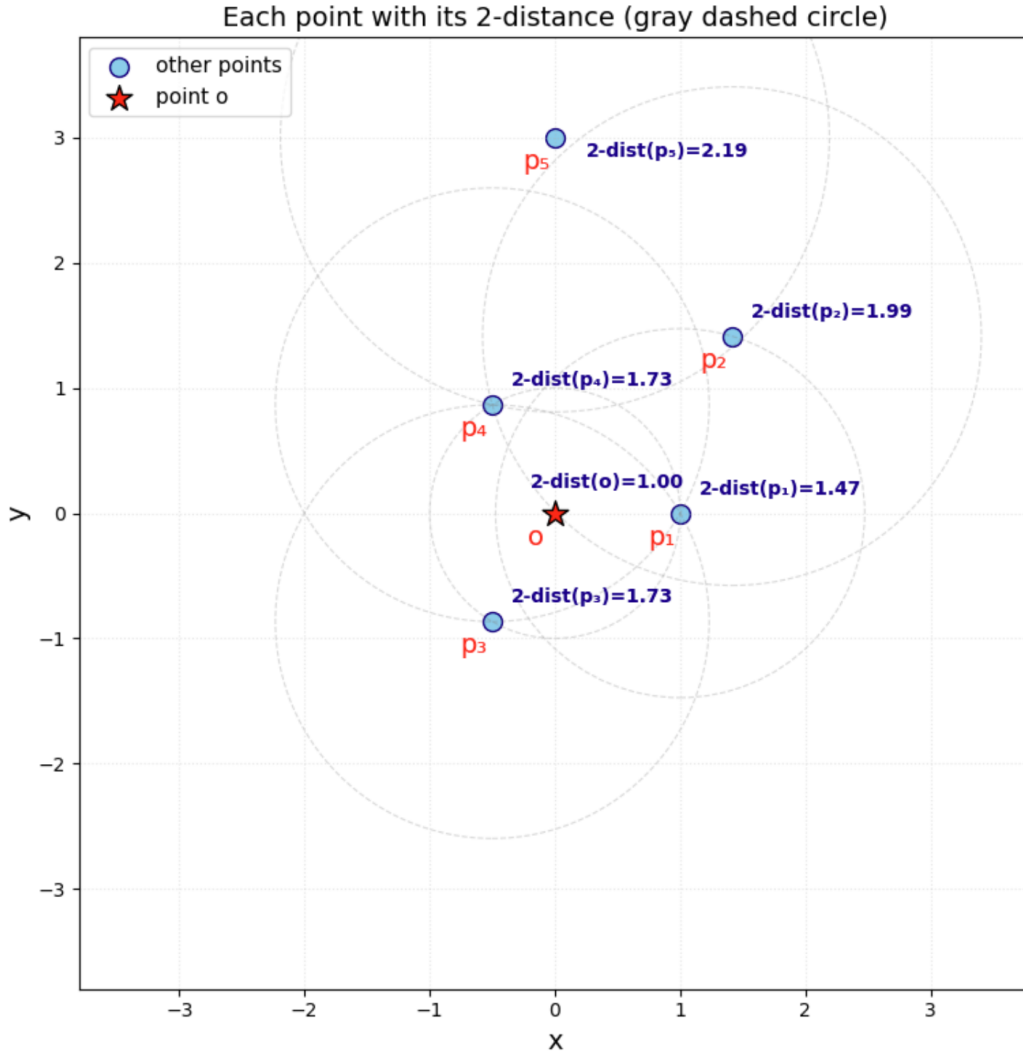


Figure 7: Illustration of the 2-distance for each object in dataset D . The red star marks the object o , and the blue objects denote the other objects p_1 through p_5 . The gray dashed circle around each object has radius equal to its 2-distance, whose value varies across objects. $2\text{-dist}(p_1) = 1.47$, and $2\text{-dist}(p_3) = 1.73$

Important note: The k -distance is a point-specific property. Every point $p \in D$ has its own $k\text{-dist}(p)$, defined in exactly the same manner but based on the distances from p to all other points in D . These values are generally distinct across different points, as they depend on the local data distribution around each individual object, rather than on the geometry around any particular reference point such as o . Consequently, the radius illustrated in Figure 6 for o should not be generalized to other objects without recomputation.

It is instructive to contrast this with DBSCAN. In DBSCAN, the parameter ϵ defines a fixed global radius: every object uses the same distance threshold to determine its neighbourhood. Changing ϵ uniformly expands or shrinks this radius for all objects simultaneously. In LOF, by contrast, there is no global radius. Each object's k -distance defines its own local radius, which is generally

different from object to object. Thus, changing k affects each object's radius locally and individually, rather than globally and uniformly.

Definition 2: (*k-distance neighborhood of an object o*)

Given the k -distance of o , the k -distance neighborhood of o , denoted as $N_k(o)$, contains every object whose distance from o is not greater than the k -distance of o , i.e.

$$N_k(o) = \{p \in D \setminus \{o\} \mid d(o, p) \leq k\text{-dist}(o)\}. \quad (12)$$

In the example of Figure 7, the neighborhood $N_2(o)$ consists of the three objects on the circle of radius 1, so its cardinality is 3. And the cardinality of $N_2(p_1)$ is 2. In general, because the definition uses $\leq$ (not $<$), the cardinality of the k -distance neighbors of an object will be $\geq k$; ties on the boundary cause the cardinality to be strictly larger than k .

Definition 3: (*reachability distance of object p w.r.t. reference object o*)

The reachability distance of object p with respect to a reference object o is defined as

$$\text{reach-dist}_k(p, o) = \max\{d(p, o), k\text{-dist}(o)\}. \quad (13)$$

The reachability distance is a second key innovation of LOF, after the k -distance. It defines a distance-like measure between two objects. Unlike a conventional metric, it is not required to be symmetric or to satisfy the triangle inequality. In general, $\text{reach-dist}_k(p, o) \neq \text{reach-dist}_k(o, p)$.

Although the computation may appear tedious at first glance, working through a concrete example is worthwhile, as it reveals how the maximum operator smooths local distance fluctuations and forms the basis for density computation in LOF.

We now illustrate the notion of reachability distance using the same dataset D . Figure 8 and Figure 9 show how $\text{reach-dist}_k(p_1, o)$ and $\text{reach-dist}_k(p_2, o)$ are derived, respectively.

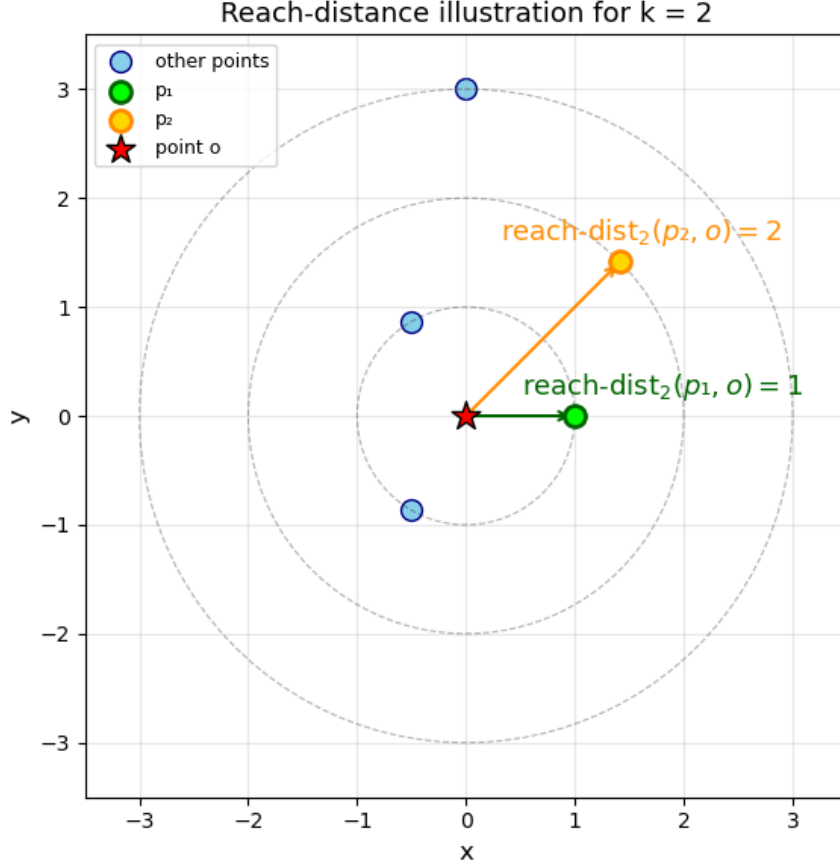


Figure 8: Illustration of reachability distance reach-dist_k for $k = 2$ on dataset D . Object o (red star) is the reference. Two example objects are highlighted: p_1 (green), and p_2 (orange). The closest dashed gray circle marks $2\text{-dist}(o) = 1$. For p_1 : $\text{reach-dist}_2(p_1, o) = \max\{1, 1\} = 1$. For p_2 : $\text{reach-dist}_2(p_2, o) = \max\{1, 2\} = 2$.

For $k = 2$, we have $2\text{-dist}(o) = 1$ shown in Figure 8, the reachability distance of p_1 , p_2 with respect to reference object o is

$$\text{reach-dist}_2(p_1, o) = \max\{d(p_1, o), 2\text{-dist}(o)\} = 1.$$

$$\text{reach-dist}_2(p_2, o) = \max\{d(p_2, o), 2\text{-dist}(o)\} = \max(2, 1) = 2.$$

Similarly, for $k = 4$, we have $4\text{-dist}(o) = 2$ shown in Figure 9. the reachability distance of p_1 , p_2 with respect to reference object o is

$$\text{reach-dist}_4(p_1, o) = \max\{d(p_1, o), 4\text{-dist}(o)\} = \max\{1, 2\} = 2.$$

$$\text{reach-dist}_4(p_2, o) = \max\{d(p_2, o), 4\text{-dist}(o)\} = \max(2, 2) = 2.$$

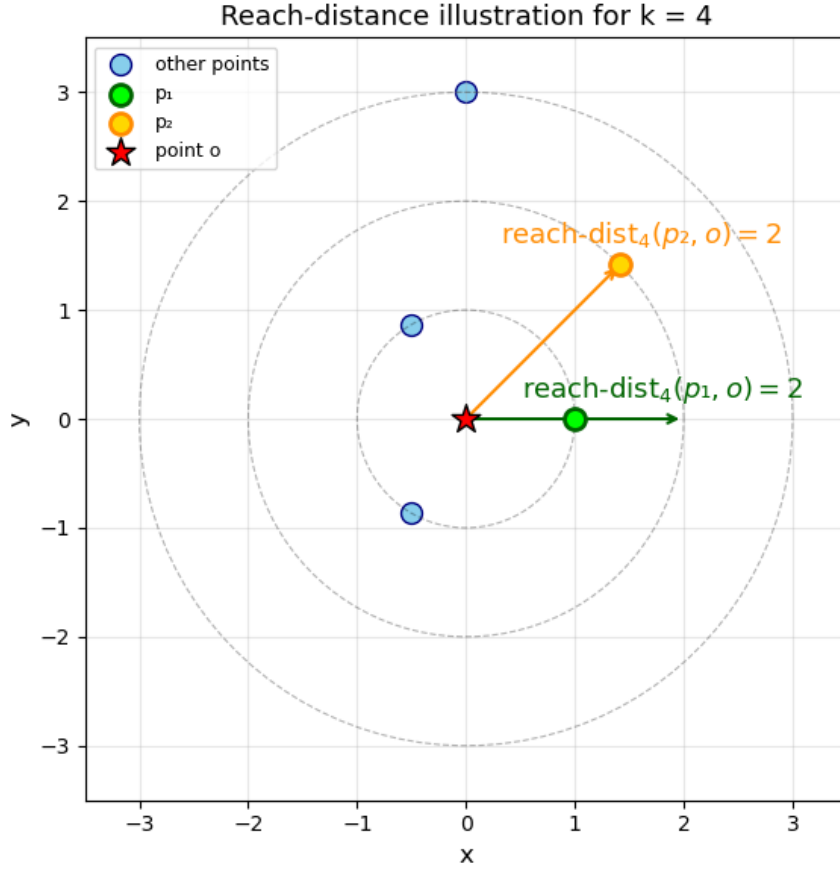


Figure 9: Illustration of reachability distance for $k = 4$ on dataset D . Object o (red star) is the reference. Two example objects are highlighted: p_1 (green), and p_2 (orange). The second closest dashed gray circle marks $4\text{-dist}(o) = 2$. For p_1 : $\text{reach-dist}_4(p_1, o) = \max\{2, 1\} = 2$. For p_2 : $\text{reach-dist}_4(p_2, o) = \max\{2, 2\} = 2$.

Thus, given reference object o , the reachability distance of p with respect to o *smooths* the distance of objects that lie within or on the $k\text{-dist}(o)$ radius by raising them to at least $k\text{-dist}(o)$ shown in Figure 10, while objects outside this boundary retain their original distances.

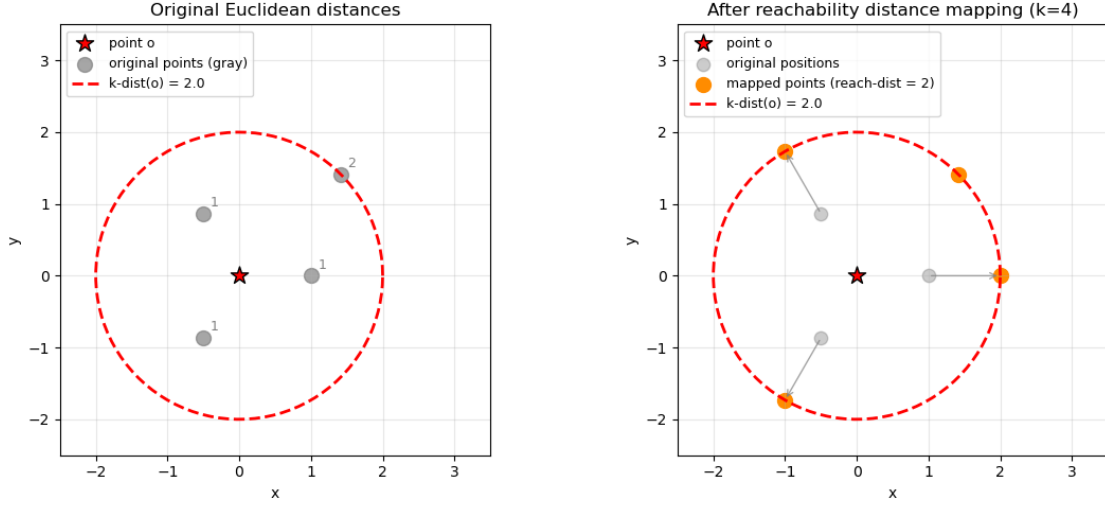


Figure 10: After reachability distance mapping $k = 4$: **(Left)** Original Euclidean distances: three objects at distance 1 and one object at distance 2. **(Right)** After reachability distance mapping for $k = 4$. All objects originally within the circle of radius 2 (i.e., with $d(o, p) \leq 2$) are mapped onto the circle (arrows). Their reachability distances become $\text{reach-dist}_4(p, o) = 2$, regardless of the original distance.

Thus, for any $p \in N_k(o)$, we have $d(p, o) \leq k\text{-dist}(o)$ by the definition of k -distance. Consequently,

$$\text{reach-dist}_k(p, o) = \max\{d(p, o), k\text{-dist}(o)\} = k\text{-dist}(o). \quad (14)$$

Thus, the use of the maximum operator in Definition 3 is not arbitrary. By taking the maximum, the reachability distance is guaranteed to be strictly positive, avoiding division-by-zero issues in subsequent density computations. More importantly, it *smooths* the statistical fluctuations within the k -neighborhood: for all points p close enough to o (i.e., $p \in N_k(o)$), the reachability distance w.r.t. object o is raised to a constant $k\text{-dist}(o)$, thereby smoothing the fluctuations of the local density estimate caused by uneven distances in the neighborhood for a fixed choice of k .

Note that the reachability distance is not symmetric: in general, $\text{reach-dist}_k(p, o) \neq \text{reach-dist}_k(o, p)$, because the k -distance of the reference object (the second argument) is used as the baseline. For $\text{reach-dist}_k(p, o)$, the baseline is $k\text{-dist}(o)$; for $\text{reach-dist}_k(o, p)$, it is $k\text{-dist}(p)$. In this sense, the reachability distance acts analogously to a distance metric tailored to each ordered pair of points, even though it does not satisfy symmetry or the triangle inequality.

Definition 4: (local reachability density of an object o)

The *local reachability density* of o is defined as the reciprocal of the average reachability distance from o to its k -neighbors:

$$\text{lrd}_k(o) = \frac{1}{\frac{1}{|N_k(o)|} \sum_{p \in N_k(o)} \text{reach-dist}_k(o, p)} \quad (15)$$

Note that the summation iterates over each neighbour $p \in N_k(o)$. For each term $\text{reach-dist}_k(o, p)$, the reference object is that neighbour p (its k -distance appears in the max operator), not the central

object o . This ensures that the density estimate at o is built from the local perspectives of its neighbouring objects.

For $k = 2$ and point o , the neighborhood $N_2(o)$ consists of three points, namely p_1 , p_3 , and p_4 (see Figure 7). Thus, $|N_2(o)| = 3$. By definition, the local reachability density of o is

$$\text{lrd}_2(o) = \frac{1}{\frac{1}{3}(\text{reach-dist}_2(o, p_1) + \text{reach-dist}_2(o, p_3)) + \text{reach-dist}_2(o, p_4)} \approx 0.60. \quad (16)$$

We compute the Local reachability density (LRD) of all objects in the synthetic dataset D with $k = 2$. Figure 11 shows the resulting LRD values for each object.

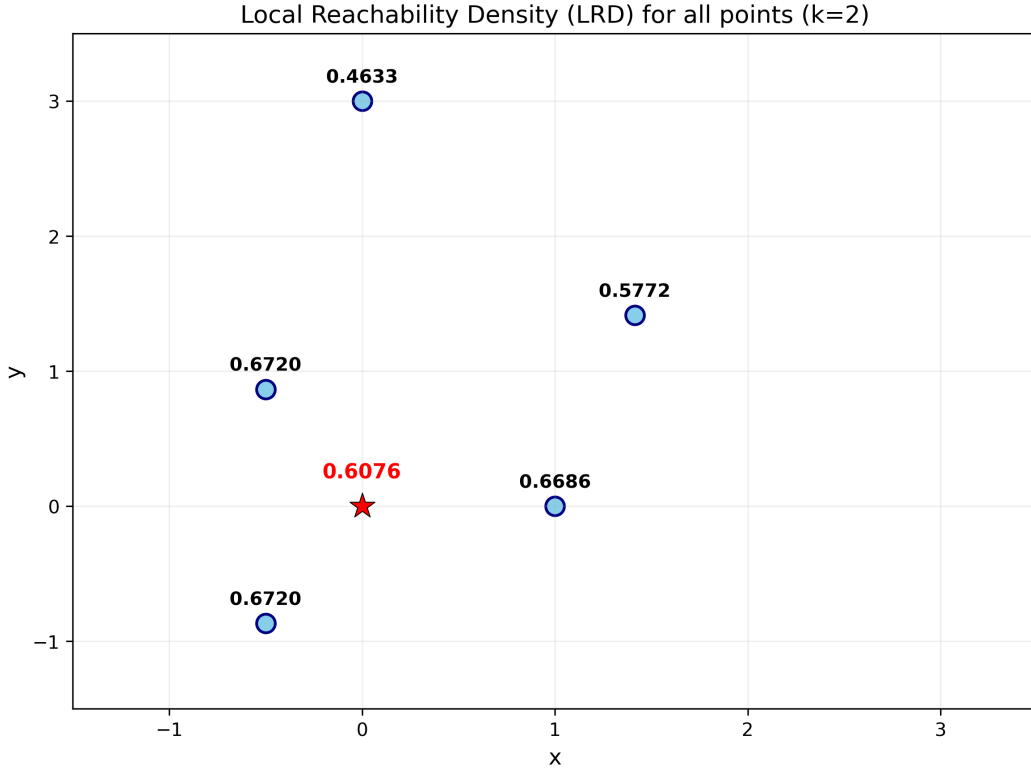


Figure 11: Local reachability density (LRD) for all objects. The object o and its neighbours exhibit comparable LRD values (approximately 0.6), while point 5 has a notably lower LRD (0.4633). These local density differences are precisely what the LOF score will later quantify.

Although the construction of these definitions is layered, their purpose is straightforward: they provide a distinct local density value for every object - a value that will later serve as the basis for comparing each object with its neighbours.

Definition 5: ((local) outlier factor of an object o)

The (local) outlier factor of o is defined as

$$\text{LOF}_k(o) = \frac{\sum_{p \in N_k(o)} \frac{\text{lrd}_k(p)}{\text{lrd}_k(o)}}{|N_k(o)|} \quad (17)$$

The local outlier factor (LOF) measures the extent to which an object o deviates from the local density of its neighbors. It is the average of the ratio of the local reachability density of o and those of o 's k -distance neighbors, and serves as an anomaly score.

The key intuition behind the Local Outlier Factor (LOF) is to compare this density with the average density of its neighbours. For points lying within a cluster of homogeneous density, this ratio is approximately 1. In contrast, anomalies deviate from this baseline, yielding a ratio that significantly differs from 1. The critical boundary is therefore exactly 1, which defines the LOF score and distinguishes normal objects from outliers.

- $\text{LOF}_k(o) \approx 1$: o has a similar density to its neighbors.
- $\text{LOF}_k(o) < 1$: o has a higher density than its neighbors (often considered a strong inlier or a core point).
- $\text{LOF}_k(o) > 1$: o has a lower density than its neighbors (outlier).

We compute LOF of all objects in the synthetic dataset D with $k = 2$. Figure 11 shows the resulting LOF values for each object.

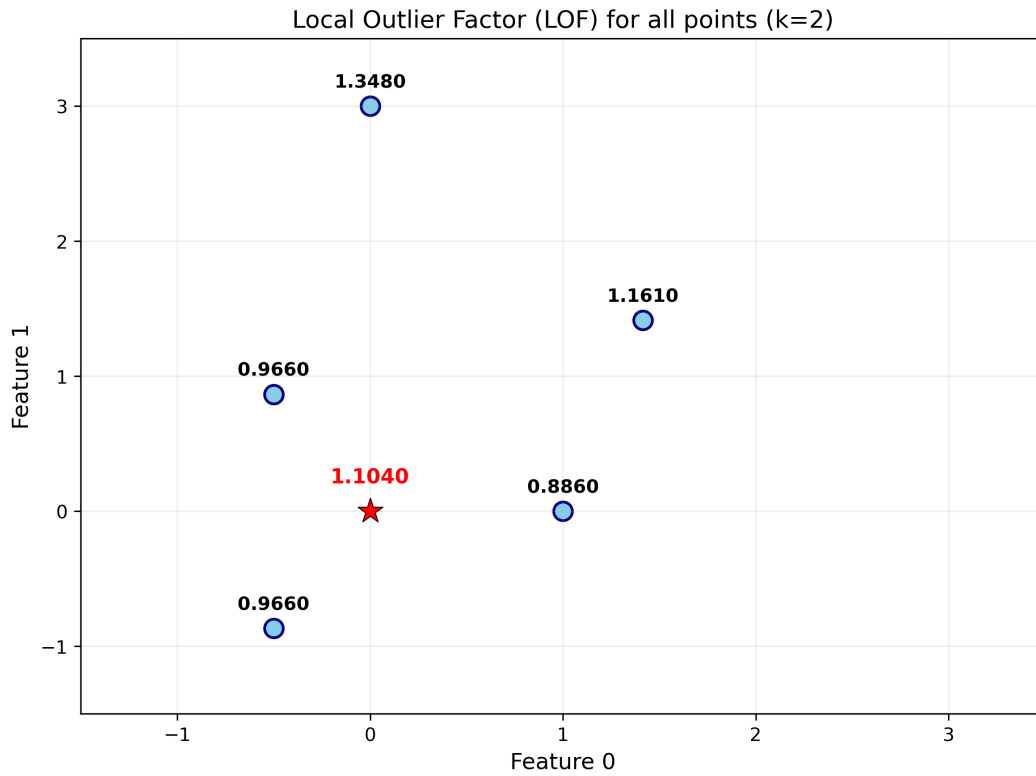


Figure 12: LOF values for all objects. The object o and its neighbours exhibit comparable LOF values around 1, while p_5 has a distinctly higher value of 1.348, marking it as a possible outlier.

We can see that the computation of LOF is a purely deterministic mathematical mapping: input data $\rightarrow$ reachability distance $\rightarrow$ density ratios $\rightarrow$ final scores.

3.2.2 The Impact of Parameter k

There is no universally optimal choice for the neighbourhood size k in LOF; instead, it is recommended to evaluate the LOF scores over a range of k values, as suggested by Breunig et al. [4], and select a value that balances sensitivity to local density variations against statistical robustness.

Three guidelines are given for selecting a suitable range of k :

1. The lower bound should be at least 10 to remove unwanted statistical fluctuations. Empirical evidence from the original LOF paper supports this: the standard deviation of LOF values only stabilizes when $k \geq 10$ for a Gaussian distribution; more strikingly, for a uniform distribution (where no outlier should exist), objects can still obtain LOF values significantly greater than 1 when $k < 10$. Hence, the first guideline is $k \geq 10$.
2. A cluster must contain at least k objects (i.e., k should not exceed the cluster size); otherwise, the objects of the cluster and an external point become mutual neighbors, making their LOF scores too similar to separate the outlier. A typical choice for the lower bound of k is 10–20.
3. The upper bound of k is the largest number such that a tightly clustered set of objects can still contain local outliers. Intuitively, k defines the maximum size of a group of objects that can still be considered as local outliers. For k values exceeding this bound, the neighborhoods of these objects become large enough to include objects from surrounding clusters, causing their LOF values to converge toward those of normal objects. Consequently, objects that were previously detectable as outliers are absorbed into the larger density structure and no longer stand out.

In this thesis, we will try to adopt these guidelines for the selection of k and its range, namely the lower bound and the upper bound. However, in practice, these guidelines are often impractical, especially in high-dimensional data. We cannot visualize the data to observe cluster sizes or tightness, and the guidelines heavily rely on such observations, which may be impossible in real-world high-dimensional scenarios. Therefore, in the experimental section 4, we will treat k as a tunable hyperparameter and evaluate its influence empirically.

Note: The synthetic dataset D used in the Section 3.2.1 is designed solely to illustrate how the definitions work.

3.2.3 Theoretical Threshold for LOF

In the original LOF paper [4], a lemma is established for objects that lie deep inside a dense cluster. Intuitively, if a point p and all its k -distance neighbors (as well as their neighbors) are contained within a tight cluster C , then the local reachability densities of p and its neighbors are almost identical.

Lemma 1. *Let C be a collection of objects. Let reach-dist-min denote the minimum reachability distance of objects in C , i.e.,*

$$\text{reach-dist-min} = \min\{\text{reach-dist}_k(p, q) \mid p, q \in C\}.$$

Similarly, let reach-dist-max denote the maximum reachability distance of objects in C . Define ε as

$$\varepsilon = \frac{\text{reach-dist-max}}{\text{reach-dist-min}} - 1.$$

Then for all objects $p \in C$ such that:

- (i) all the k -distance neighbors q of p are in C , and
- (ii) all the k -distance neighbors o of q are also in C ,

it holds that

$$\frac{1}{1+\varepsilon} \leq \text{LOF}(p) \leq 1+\varepsilon.$$

Proof. Under the two assumptions, for any $p \in C$ satisfying conditions (i) and (ii), and for any neighbor $q \in N_k(p)$ and the equation (14), we get

$$\text{reach-dist}_k(p, q) = k\text{-dist}(q).$$

Let the global minimum and maximum reachability distances within cluster C be denoted by

$$m := \text{reach-dist-min}, \quad M := \text{reach-dist-max}.$$

Then for any object $x \in C$, all its reachability distances to its neighbors lie in the interval $[m, M]$. Hence, the average of these distances also lies in $[m, M]$. By the definition of local reachability density,

$$\text{lrd}_k(x) = \frac{|N_k(x)|}{\sum_{o \in N_k(x)} \text{reach-dist}_k(x, o)},$$

which immediately implies the following boundedness:

$$\frac{1}{M} \leq \text{lrd}_k(x) \leq \frac{1}{m}.$$

Now, for the target point p satisfying the two conditions, for any neighbor $q \in N_k(p)$, applying the above bounds yields:

$$\frac{\text{lrd}_k(q)}{\text{lrd}_k(p)} \leq \frac{1/m}{1/M} = \frac{M}{m} = 1+\varepsilon,$$

and similarly,

$$\frac{\text{lrd}_k(q)}{\text{lrd}_k(p)} \geq \frac{1/M}{1/m} = \frac{m}{M} = \frac{1}{1+\varepsilon}.$$

Since $\text{LOF}_k(p)$ is defined as the arithmetic mean of these ratios over all $q \in N_k(p)$, the same bounds are preserved under averaging. Therefore,

$$\frac{1}{1+\varepsilon} \leq \text{LOF}_k(p) \leq 1+\varepsilon,$$

which completes the proof. $\square$

Consequently, the LOF of such a object is bounded between $\frac{1}{1+\varepsilon}$ and $1+\varepsilon$, where ε is a positive number whose size depending on the tightness of C . For a sufficiently compact cluster, ε becomes very small, forcing the LOF of these deep points to be approximately 1.

This result provides a theoretical baseline: normal points deeply embedded in a cluster should have LOF values close to 1. Points whose LOF deviates significantly from 1 can thus be considered candidates for outliers.

However, in real-world applications, the LOF of normal points may not be exactly 1 due to sampling variations, noise, or imperfect cluster structures. Therefore, choosing a decision threshold solely based on 1 is often too strict. A more practical approach is to:

- Use the theoretical value 1 as a reference point for anomaly score, and
- Combine it with domain knowledge, such as the expected proportion of outliers (contamination rate) or the desired classification ratio, to select an empirical threshold via rank-based methods (e.g., take the top $t\%$ of LOF scores as outliers).

4 Experimental Results and Analysis

In this section, we train models using Isolation Forest and LOF introduced in Section 3 to compare their performance on two publicly available datasets: the real-world Credit Card Fraud Detection dataset from Kaggle [17] and the synthetic Mulcross dataset from OpenML [18].

The two datasets were selected to represent contrasting scenarios for anomaly detection. The Mulcross dataset is synthetic and well-separated, meaning anomalies are clearly distinct from normal instances closely matching the theoretical assumptions of iForest and LOF. The CreditCardFraud dataset, by contrast, is a real-world dataset where anomalies (fraudulent transactions) overlap with normal instances in the feature space. This contrast allows us to assess how iForest and LOF perform both under ideal conditions and under more challenging, realistic conditions where the theoretical assumptions are not fully satisfied.

Table 1: Properties of the experimental datasets: The table has two columns: n (number of instances), d (number of dimensions), and Anomaly class which gives the anomaly proportion in parentheses (e.g., 0.17%).

Dataset	n	d	Anomaly Proportion
CreditCardFraud	284807	31	Fraud (0.17%)
Mulcross	262144	4	2 clusters (10%)

These datasets also vary in sample size (n), dimensionality (d), and anomaly proportion, as summarized in Table 1. By comparing the results across these datasets, we aim to assess how each method behaves with respect to increasing dimensionality, different levels of anomaly proportion, and sample size.

4.1 Experimental Setup

4.1.1 Setup

All experiments are implemented in Python 3.11.14. The anomaly detection models are constructed using the following libraries:

- **Isolation Forest** and **LOF** are implemented with `scikit-learn` (version 1.8.0).

In line with the evaluation protocol adopted in the isolation forest paper, we employ a random split of each dataset into a training set and a test set. The training set is used exclusively for the unsupervised model fitting process, where the model learns the underlying structure of the data without access to any labels. The test set is held out and used solely for evaluating the models performance, at which point the labels are utilized to compute metrics such as the ROC-AUC and PR-AUC.

For data preprocessing, each feature is standardized using `StandardScaler` which Standardize features by removing the mean and scaling to unit variance from `scikitlearn`.

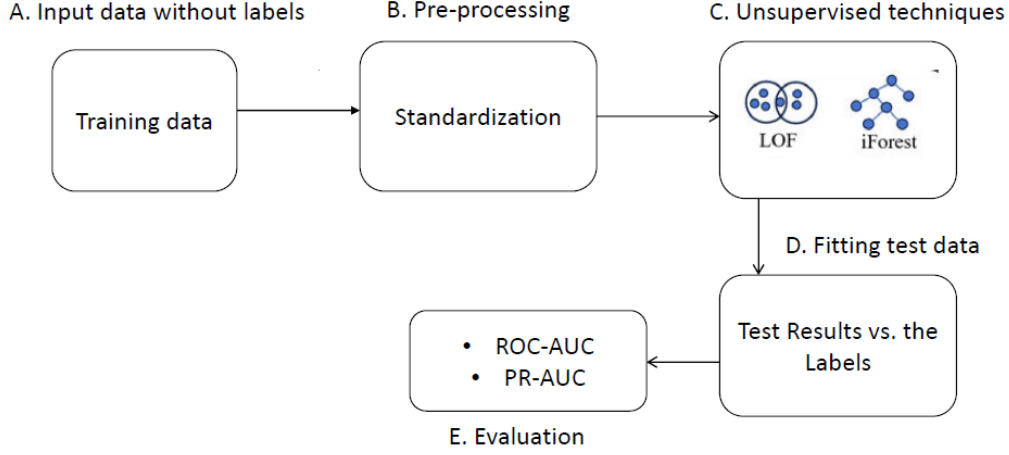


Figure 13: Framework of unsupervised anomaly detection. Adapted from [13].

Sensitivity Analysis. Additionally, a sensitivity analysis is conducted on key hyperparameters to assess model robustness across different configurations while keeping other parameters with their default or commonly recommended parameter configurations as reported in the original papers [1] [4]:

- **Isolation Forest:** number of trees $n_{\text{estimators}} \in \{50, 100, 200\}$;
- **LOF:** number of nearest neighbors $k \in \{10, 30, 50\}$;

4.1.2 Model evaluation: ROC and PR analysis

For model evaluation, we use Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves. Both provide threshold-independent measures of model performance by evaluating the model's continuous output (e.g., anomaly scores or probabilities) across a range of decision thresholds [5]. This is suitable for the two methods considered in this study, as both Isolation Forest and LOF produce continuous anomaly scores.

Although iForest and LOF are unsupervised methods, both datasets provide ground truth labels - synthetic labels in Mulcross and fraud labels in the Credit Card dataset. These labels are used exclusively for evaluation purposes, not for training. This allows us to apply supervised evaluation metrics such as ROC-AUC and PR-AUC, which provide meaningful comparative insights into how well each method ranks anomalies against normal instances - an approach consistent with established practice in anomaly detection, including the original iForest paper (Liu et al., 2008). The results themselves, rather than the mere existence of labels, provide evidence for whether the detection criteria (sparsity, low density, isolation) align with the phenomenon of interest.

Figure 14 presents the confusion matrix and the basic single-threshold metrics derived from it (e.g., precision, recall, specificity). While these metrics are computed at a fixed threshold, the ROC and PR curves are obtained by sweeping the threshold across its entire range and summarising the resulting trade-offs. In this way, the confusion matrix provides the building blocks for the curve-based evaluation.

Confusion Matrix and Common Metrics

True/Prediction	Positive	Negative
Positive	True Positives(TP)	False Negatives(FN)
Negative	False Positives(FP)	Ture Negatives(TN)

$$\text{TP rate (recall/sensitivity)} = \frac{TP}{TP+FN}$$

$$\text{FP rate (specificity)} = \frac{FP}{FP+TN}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

Figure 14: Confusion Matrix: For binary classification, a confusion matrix is computed at a specific threshold. From it, several common performance metrics can be derived. Adapted from [5].

ROC analysis ROC analysis evaluates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) over all possible thresholds, summarized by the Area Under the ROC Curve (ROC-AUC). An AUC of 0.5 indicates a random classifier (no better than guessing), while an AUC of 1.0 corresponds to a perfect classifier.

To illustrate the interpretation of the ROC curve, we first simulate a score distribution (Figure 15) and then construct the corresponding ROC curve (Figure 16) from the same simulated data. The simulated data are used solely to illustrate the relationship between the score distribution and the ROC/PR curve and do not represent actual experimental results.

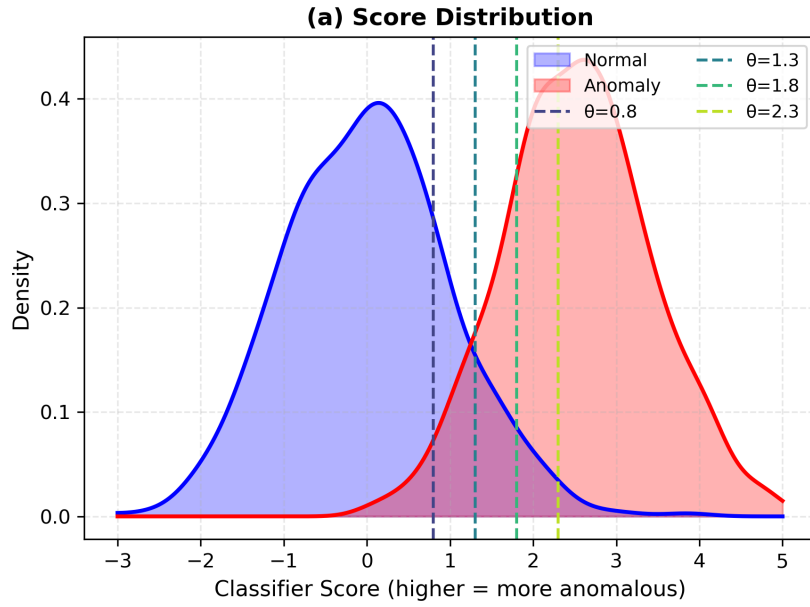


Figure 15: Simulated Score distribution: Simulated Score distribution of negative (normal, blue) and positive (anomaly, red) instances, with vertical dashed lines indicating the selected thresholds. The data are generated synthetically for illustrative purposes only. Visualization adapted from [15].

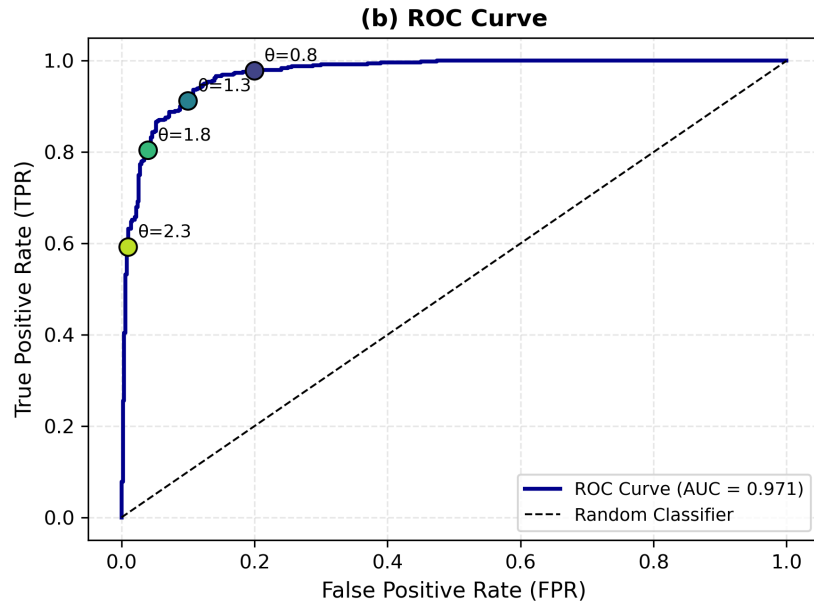


Figure 16: ROC curve derived from the simulated score distribution shown in Figure 15. The curve achieves an ROC-AUC of 0.971, indicating strong discriminative ability. The highlighted points correspond to the same thresholds indicated by the vertical dashed lines in Figure 15. Visualization adapted from [15].

From Figure 16, we can observe that when the decision threshold is set to 0.8 (highlighted as a blue point on the curve), the model achieves a true positive rate (TPR) of approximately 0.98, meaning that 98% of the simulated anomalies are correctly detected. However, this high recall comes at the cost of a false positive rate (FPR) of around 0.20, indicating that approximately 20% of the simulated normal instances are incorrectly classified as anomalies.

Precision-Recall (PR) Curves ROC curves are widely used, but as shown by the FPR formula in Figure 14, the denominator is true negative instances (FP + TN). In highly imbalanced anomaly detection scenarios, this makes the FPR insensitive to false positives, causing the ROC curve to appear overly optimistic.

To address this limitation, we additionally employ the Precision-Recall (PR) curve, which plots precision against recall (TPR) across different thresholds. The PR curve focuses on the performance of the positive (anomaly) class and is more informative than ROC when the positive class is rare. Specifically, it captures the critical trade-off between not missing positive cases (high recall) and not falsely labeling negatives as positives (high precision).

The Area Under the PR Curve (PR-AUC) serves as a complementary, threshold-independent metric that better reflects a model's ability to correctly identify anomalies without being misled by the majority negative class. For a random classifier, the expected AUC-PR equals the proportion of positive instances in the dataset, i.e., $\frac{P}{P+N}$, where P is the number of anomalies and N the number of normal samples. Consequently, a model with PR-AUC above this baseline outperforms random guessing, while an AUC of 1.0 corresponds to a perfect classifier. [14].

Figure 17 presents the PR curve, achieving a PR-AUC of 0.967. At a threshold of 0.8, the model achieves near-perfect recall (TPR ≈ 1), detecting almost all anomalies, while maintaining a precision of approximately 0.83 - indicating that 83% of the predicted positives are correct, with 17% false alarms (falsely labeling negatives as positives).

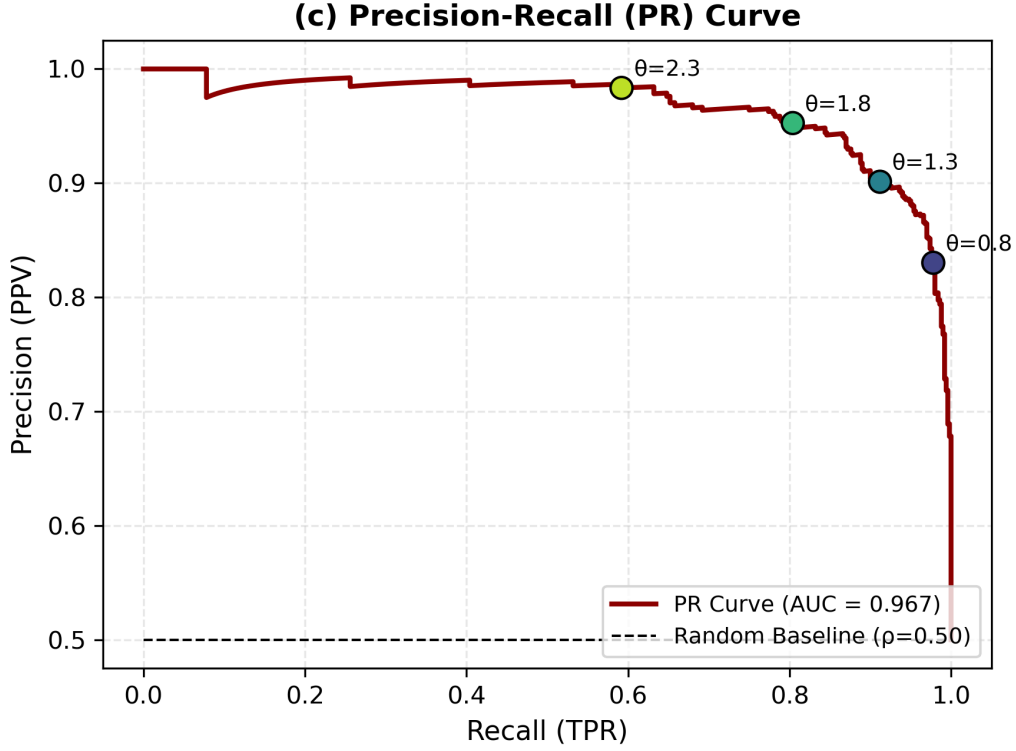


Figure 17: PR curves: PR curve derived from the simulated data for illustrative purposes. The curve achieves a PR-AUC of 0.967, indicating strong performance on the positive (anomaly) class. The highlighted points correspond to the same thresholds shown in Figures 15 and 16. Data are simulated for illustration only. Visualization adapted from [15].

4.2 Experimental Results

This section presents and discusses the results of the iForest and LOF models on the synthetic Mul-cross dataset and the real-world CreditCardFraud Detection dataset described in Section 4 - under two distinct training regimes:

- **Clean training:** The model is fitted exclusively on normal samples. This corresponds to the *novelty detection* paradigm.
- **Contaminated training:** The training set contains both normal and anomalous instances, simulating the realistic *unsupervised outlier detection* setting.

For each dataset, we report the area under the ROC curve (ROC-AUC), the area under the Precision-Recall curve (PR-AUC), and the training time (in seconds) for three different hyperparameter configurations. All performance metrics are computed using the ground-truth labels, which are used only for evaluation to assess how well the unsupervised scores rank anomalous instances above normal ones.

4.2.1 Results on the Mulcross Dataset

In this section, We use a synthetic dataset consisting of 262,144 instances, each described by four numerical features (V1–V4) and a binary label indicating whether the instance is normal or anomalous. The anomaly class constitutes 10% of the data, a moderately imbalanced proportion, and is well separated from the normal class. This clear separation makes the dataset ideal for evaluating LOF and iForest under ideal conditions.

Mahalanobis Distance Distribution As discussed in Section 2.2.1 Mahalanobis distance measures how many standard deviations a point is away from the centre of the normal distribution, while accounting for feature correlations.

Figure 18 the density distribution of Mahalanobis distances for normal and anomalous instances on the Mulcross dataset. The two classes are completely separated, with normal samples concentrated at low distances and anomalous samples shifted toward significantly higher values.

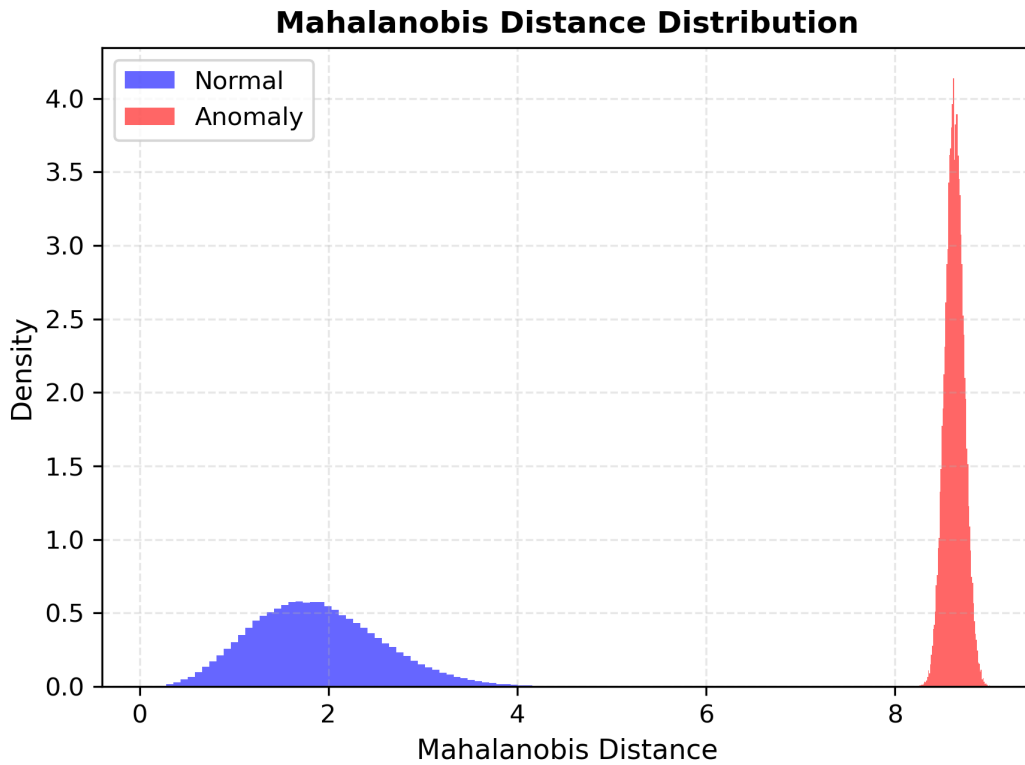


Figure 18: Distribution of Mahalanobis distances for normal (blue) and anomalous (red) instances: Normal instances have a peak at lower distances, while anomalous instances tend to have higher distances. The two distributions are well separated, showing that Mahalanobis distance can distinguish between normal and anomalous instances.

Table 2: Performance on dataset Mulcross comparison of LOF vs. iForest under clean and contaminated training. The best results are boldfaced. iForest is significantly faster than LOF. In terms of ROC-AUC, iForest performs similarly well in both clean and contaminated data. LOF gets perfect ROC-AUC and PR-AUC 1 in the clean data.

Training Regime	LOF (Density-based)				Isolation Forest (Partition-based)			
	Params	ROC-AUC	PR-AUC	Time (s)	Params	ROC-AUC	PR-AUC	Time (s)
Clean	k=10	1	1	5.0	$n = 50$	0.993	0.968	0.24
	$k = 30$	1	1	11.50	$n = 100$	0.999	0.992	0.64
	$k = 50$	1	1	15.93	$n = 200$	0.999	0.992	1.36
Contam.	$k = 10$	0.576	0.160	7.4	$n = 50$	0.958	0.531	0.33
	$k = 30$	0.589	0.174	13.37	n=100	0.968	0.592	0.67
	$k = 50$	0.587	0.172	17.57	$n = 200$	0.964	0.562	1.13

Parameters: *Clean*: model fitted only on normal samples; tested on both classes.

Contam. (Contaminated) : both training and test sets contain both normal and anomalous instances (unsupervised mode). LOF uses k as k -distance neighbors; Isolation Forest uses n as iTrees.

Interpretation: Three observations are worth highlighting.

First, under clean training, LOF achieves *perfect* ROC-AUC and PR-AUC (1.000) across all k values, while Isolation Forest also performs excellently ($\text{ROC-AUC} \geq 0.993$). This demonstrates that when the training data are not contaminated, both models can perfectly separate Mulcross anomalies from normal points—an expected outcome given the well-separated synthetic nature of the data.

Second, under contaminated training, LOF’s performance drops dramatically ($\text{ROC-AUC} \approx 0.58$, $\text{PR-AUC} \approx 0.16$). A probable explanation lies in the data structure of the anomalies: while Mahalanobis distance measures global deviation from the centre, LOF relies on local density comparisons. Therefore, even under global separation, local structures - such as compact anomaly clusters - can mislead LOF’s density estimates under contaminated training. When anomalies are included in training, they form local clusters that LOF interprets as normal high-density regions, causing it to assign low outlier scores to anomalies that are surrounded by other anomalies.

Third, Isolation Forest exhibits remarkable robustness to contamination. Even with anomalies in the training set, it maintains ROC-AUC of approximately 0.96 to 0.97 and PR-AUC of approximately 0.53 to 0.59. This resilience is inherent to its partitioning mechanism: random splits isolate anomalies early regardless of whether they appear in the training set, making it less sensitive to data contamination. And iForest is significantly faster than LOF across all parameter settings.

Crucially, these results on Mulcross serve as a *sanity check*: when the data satisfy the assumptions, the evaluation metrics correctly reflect the expected behaviour, confirming that using labels as an evaluation reference is both valid and informative.

4.2.2 Results on the CreditCardFraud Dataset

In this section, We use CreditCardFraud dataset. The dataset consists of 284,807 transactions, each described by 30 numerical features: time, amount, and 28 principal components (V1-V28) obtained from an undisclosed transformation. Only 492 transactions (0.17%) are labelled as fraud (Class 1), the rest are normal (Class 0).

Unlike Mulcross, this dataset exhibits significant overlap between normal and fraudulent transactions, with many fraud cases embedded in dense regions, as shown in Figure 19.

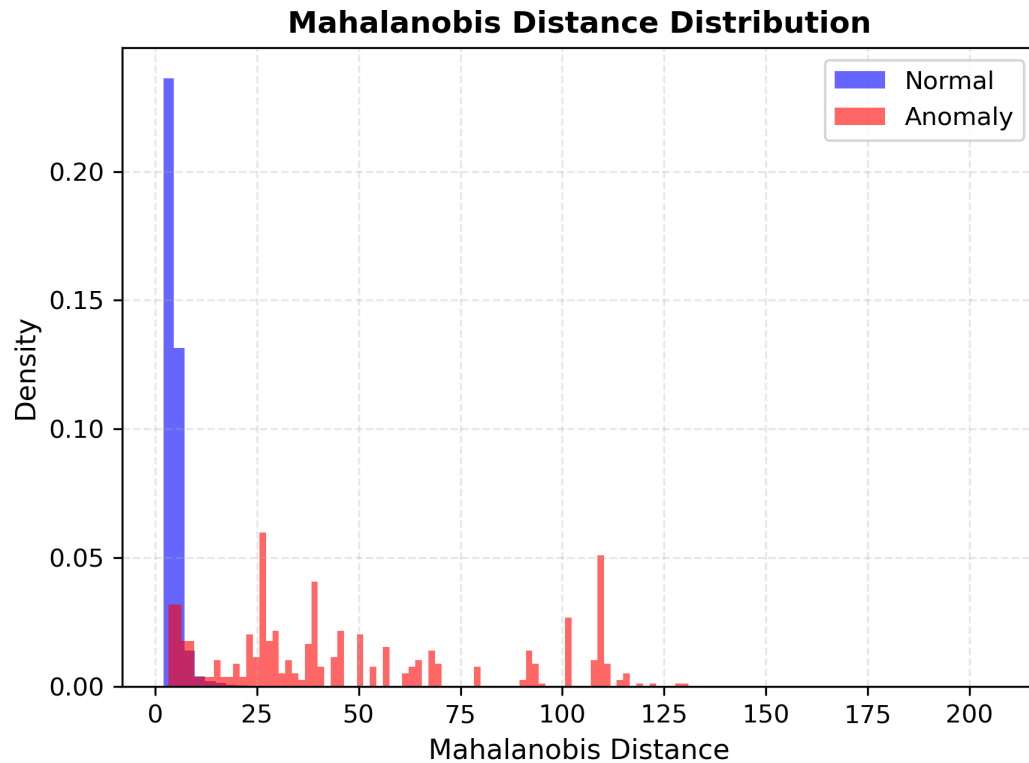


Figure 19: Density of Mahalanobis distances for normal and fraudulent: The density plot shows Mahalanobis distance for normal (blue) and fraudulent (red) transactions with horizontal axis Mahalanobis Distance and vertical axis Density. The normal transactions, with a sharp peak at low distance (mean 2.40) and a long right tail extending beyond 125; fraudulent distances are shifted rightward. The two distributions are only partially separated.

Table 3: Performance on the dataset CreditCardFraud comparison of LOF vs. iForest under clean and contaminated training. The best results are boldfaced. iForest is significantly faster than LOF. In terms of ROC-AUC, iForest performs similarly well in both clean and contaminated data.

Training Regime	LOF (Density-based)				Isolation Forest (Partition-based)			
	Params	ROC-AUC	PR-AUC	Time (s)	Params	ROC-AUC	PR-AUC	Time (s)
Clean	$k = 10$	0.66	0.046	376.71	$n = 50$	0.949	0.295	0.44
	$k = 30$	0.787	0.040	387.16	$n = 100$	0.951	0.344	0.87
	$k = 50$	0.933	0.177	538.88	$n = 200$	0.950	0.343	1.45
Contam.	$k = 10$	0.597	0.003	362.81	$n = 50$	0.953	0.177	0.55
	$k = 30$	0.582	0.004	608.29	$n = 100$	0.955	0.301	0.99
	$k = 50$	0.575	0.005	318.49	$n = 200$	0.954	0.191	1.54

Interpretation: Three key findings emerge from this real-world dataset.

First, Isolation Forest consistently achieves strong performance ($\text{ROC-AUC} \approx 0.95$) under both clean and contaminated training, with minimal variation across the number of trees - a finding that aligns with the theoretical convergence bound established in Section 3.1.2. This indicates that the isolation-based detection mechanism generalises well to realistic, high-dimensional data where the anomaly structure is not as cleanly separated as in synthetic benchmarks. Its PR-AUC, however, remains notably low (clean: ≈ 0.34 ; contaminated: ≈ 0.30).

Second, LOF shows mixed results. Under clean training, LOF improves with larger k ($0.660 \rightarrow 0.933$ for ROC-AUC), suggesting that LOF is highly sensitive to the neighborhood size k , a finding consistent with the theoretical analysis in Section 3.2. However, its PR-AUC remains low (best 0.177). Even with reasonable ranking ability in terms of ROC, the majority of positive-prediction labels are false positives.

Third, this disparity between ROC-AUC and PR-AUC confirms the earlier methodological discussion (Section 4.1.2) that ROC-AUC can be overly optimistic in imbalanced settings.

In summary, the CreditCard results show that even when the data deviate from the ideal assumptions of the detectors, the evaluation framework remains informative. It quantifies not whether the models *perfectly* replicate the labels, but rather whether they capture meaningful deviations that align with fraudulent activity to a statistically significant degree.

4.3 Comparative Summary

Across both datasets, three consistent patterns emerge:

- **LOF excels when training is clean and the data satisfy its density assumptions** (Mulcross perfect AUC), but degrades substantially under contamination-confirming its theoretical limitation.
- **Isolation Forest is notably robust to training contamination**, making it a safer choice in fully unsupervised deployment scenarios.
- **PR-AUC is more discriminative than ROC-AUC in highly imbalanced settings**, especially when assessing model utility for rare-event detection.

5 Conclusion and Discussion

In this study, we examine the theoretical foundations, implement, and empirically compare two unsupervised anomaly detection methods: Isolation Forest (iForest) and Local Outlier Factor (LOF).

Our experimental results highlight complementary strengths of the two approaches: iForest demonstrates greater robustness than LOF on the two datasets, while LOF achieves perfect scores under ideal conditions but is more sensitive to data contamination and parameter k .

Several limitations of this study should be acknowledged. First, our evaluation is restricted to two datasets, which may limit the generalizability of our conclusions. Second, our use of supervised evaluation metrics (ROC-AUC and PR-AUC) to assess unsupervised detectors relies on the availability and quality of ground-truth labels; in real-world deployment without labels, such quantitative validation is not possible, and practitioners must rely on alternative strategies such as model stability checks or domain expert feedback.

To address the limitations above, future work could extend the empirical evaluation to more diverse datasets, including high-dimensional and time-series data, and explore additional anomaly detection methods as well as unsupervised evaluation metrics. We also plan to investigate unsupervised or self-adaptive strategies for parameter selection and ensemble techniques to enhance robustness across diverse data regimes.

References

- [1] Liu F T, Ting K M, Zhou Z H. *Isolation forest*, 2008 Eighth IEEE International Conference on Data Mining (ICDM), 2008, pp. 413-422.
- [2] B.R.Preiss. *Data Structures and Algorithms with Object- Oriented Design Patterns in Java..* Wiley, 1999. Available at: <https://book.huihoo.com/data-structures-and-algorithms-with-object-oriented-design-patterns-in-java/html/page9.html>
- [3] T. Hastie, R. Tibshirani and J. Friedman. *Elements of Statistical Learning*, Ed. 2, Springer, 2009.
- [4] Breunig, Markus M., et al. *LOF: identifying density-based local outliers.*, Proceedings of the 2000 ACM SIGMOD international conference on Management of data. 2000.
- [5] Fawcett, Tom. *An introduction to ROC analysis.*, Pattern recognition letters 27.8 (2006): 861-874.
- [6] Allan Gut. *An Intermediate Course in Probability.*, Springer (2009).
- [7] Leonhard Held, Daniel Sabanés Bové. *Likelihood and Bayesian Inference.*, Springer (2020).
- [8] Charu C. Aggarwal. *Outlier Analysis.*, Springer (2017).
- [9] D. Hawkins. *Identification of Outliers.*, Chapman and Hall (1980).
- [10] C. Phua, V. Lee, K. Smith, and R. Gayler. *A Comprehensive Survey of Data Mining-based Fraud Detection Research*, (2010).
- [11] V. Chandola, A. Banerjee, and V. Kumar. *Anomaly Detection: A Survey*, ACM Computing Surveys (CSUR), 41(3):1–58 (2009).
- [12] N. West and J. Deuse. *A Comparative Study of Machine Learning Approaches for Anomaly Detection in Industrial Screw Driving Data*, (2024).
- [13] Chen, Zugang, et al. *Unsupervised Anomaly Detection Methods for In Situ Observation Data: A Comparative Study.*, IEEE Access 13 (2025): 187347-187360.
- [14] Saito, Takaya, and Marc Rehmsmeier. *The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets.*, PloS one 10.3 (2015): e0118432.
- [15] Richardson E, Trevizani R, Greenbaum J ... *The receiver operating characteristic curve accurately assesses imbalanced datasets.*, Patterns, 2024; 5
- [16] Gut, A. (2013). *Probability: A Graduate Course* (2nd ed.). Springer(2013).
- [17] Machine Learning Group - ULB. *Credit Card Fraud Detection*. Kaggle. Available at: <https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>
- [18] OpenML. *MULCROSS*. Available at: <https://www.openml.org/>

Masteruppsats 2026:19
Matematisk statistik
September 2026

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm