LSHiForest: A Generic Framework for Fast Tree Isolation Based Ensemble Anomaly Analysis

Conference Paper · April 2017
DOI: 10.1109/ICDE.2017.145

CITATIONS 3
READS 70

7 authors, including:

Qiang He
Swinburne University of Technology
60 PUBLICATIONS 493 CITATIONS

Rui Zhou
Swinburne University of Technology
56 PUBLICATIONS 437 CITATIONS

Kotagiri Ramamohanarao
University of Melbourne
474 PUBLICATIONS 7,785 CITATIONS

Zoran Salcic
University of Auckland
311 PUBLICATIONS 1,740 CITATIONS

Some of the authors of this publication are also working on these related projects:

Project
Keyword Search -- XML, Graph, Road Network, Service Graph View project

Project
Hot Topics -- Big Data, Machine Learning, Data Mining View project

All content following this page was uploaded by Rui Zhou on 24 January 2018.
The user has requested enhancement of the downloaded file.
Abstract—Anomaly or outlier detection is a major challenge in big data analytics because anomaly patterns provide valuable insights for decision-making in a wide range of applications. Recently proposed anomaly detection methods based on the tree isolation mechanism are very fast due to their logarithmic time complexity, making them capable of handling big data sets efficiently. However, the underlying similarity or distance measures in these methods have not been well understood. Contrary to the claims that these methods never rely on any distance measure, we find that they have close relationships with certain distance measures. This implies that the current use of this fast isolation mechanism is only limited to these distance measures and fails to generalise to other commonly-used measures. In this paper, we propose a generic framework named LSHiForest for fast tree isolation based ensemble anomaly analysis with the use of a Locality-Sensitive Hashing (LSH) forest. Being generic, the proposed framework can be instantiated with a diverse range of LSH families, and the fast isolation mechanism can be extended to any distance measures, data types and data spaces where an LSH family is defined. In particular, the instances of our framework with kernelised LSH families or learning based hashing schemes can detect complicated anomalies like local or surrounded anomalies. We also formally show that the existing tree isolation based detection methods are special cases of our framework with the corresponding distance measures. Extensive experiments on both synthetic and real-world benchmark data sets show that the framework can achieve both high time efficiency and anomaly detection quality.

I. INTRODUCTION

Anomaly detection or outlier detection aims to automatically identify or predict anomalous but insightful patterns in data sets. It is an important and powerful data mining technique which is widely adopted in a diverse range of applications such as fraud detection, system health monitoring and event detection in sensor networks. Throughout this paper we use the terms ‘anomaly’ and ‘outlier’ interchangeably as the definition of an anomaly or outlier remains rather vague. A commonly adopted one is “an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism” [1], e.g., abnormally high temperature readings in environmental sensing data may indicate an occurrence of bushfire. A wide range of methods have been proposed based on different models of anomalies in various domains [2], [3]. Some seminal examples are clustering based outlier detection [4], distance (or density) based outlier detection [5], relative density based outlier detection [6], angle based outlier detection [7] and isolation based outlier detection [8]. One can refer to [2], [3] for more details.

In the current big data era, data sets with large-volume, high-dimensional, heterogeneous, geographically distributed and fast-evolving characteristics pose a considerable challenge on traditional data mining and analytics techniques including anomaly detection [9]. The problem of anomaly detection on big data sets becomes even harder as we need to find ‘needles’ in a much larger and more complex ‘haystack’. Specifically, the large-volume and high-dimensional features usually disable traditional algorithms having quadratic (or higher) computational complexity as they often fail to produce meaningful results within a “tolerable elapsed time”. Hence, the detection algorithms for big data are often required to be highly scalable. Besides, most anomaly detection applications such as network intrusion detection and human health monitoring, demand low-latency or even real-time detection or prediction to trigger a fast response in the case of anomalous events. As such, scalability and efficiency are of paramount importance for the success of anomaly detection in big data applications.

Given these challenges, subsampling based ensemble methods are believed to be a promising way to enhance the efficiency and performance of existing detection algorithms for big data. Ensemble methods have been widely adopted as a meta-algorithm for many data mining tasks, especially for classification and clustering, aiming to reduce the dependence of models on specific datasets or data locality, and enhance the robustness of a data mining process [10]. As to anomaly detection, however, it has been recently pointed out that ensemble outlier analysis is an emerging research area and still in its infancy due to the intrinsic nature of the outlier analysis problem [11], [12]. The challenge of anomaly detection is mainly twofold, i.e., small sample space (a given data set usually contains only a small fraction of outliers) and unsupervised nature (lack of ground truth) [11]. Thus, few efforts have been made to apply ensemble methods in improving the detection performance in a principled way. The work in [11], [13] has initially formulated the key algorithmic ingredients of ensemble outlier analysis, and developed the theoretical foundation from the perspective of the well-established bias-variance trade-off theory [14].

Compared to classification and clustering, the variety of
ensemble based anomaly detection algorithms is still limited and the corresponding work in the literature is relatively complementary and sporadic. The work on feature bagging in [15] is regarded as the first attempt to use ensemble for anomaly detection in a formal and explicit way, while some earlier work using ensembles implicitly also exists as pointed out in [11]. Orthogonal to feature bagging or subspace ensemble methods, a subsampling based ensemble is leveraged in [16] to improve the performance and efficiency of the seminal Local Outlier Factor (LOF) analysis [6]. However, the computational complexity of predicting the anomaly scores of each data instance is at least linear in these algorithms with respect to the sample size, making the whole detection or prediction process quite time-consuming in the case of large-scale data. The salient work in [8] proposes iForest, an anomaly detection method that leverages a random forest to isolate data instances and derives anomaly scores based on path lengths. This method is very fast because the average computational complexity at the evaluation stage is logarithmic with respect to the sample size. To avoid confusion, we call this isolation mechanism as tree isolation because it uses a tree structure to isolate data instances. Note that another isolation mechanism based on building hyperspheres has been proposed for anomaly detection [17], but it has high computational complexity. As pointed out in [17], [18], iForest suffers from an incapability in detecting complicated anomalies such as axis-parallel, local or surrounded anomalies. Its variant SCiForest [18] uses multiple attributes at each step of data partitioning and an optimisation process to mitigate the drawbacks of iForest. However, the distance or similarity measures underlying these two methods have not been well understood since the authors inappropriately claimed that both methods never rely on any distance measure. As analysed in Section IV-C, iForest and SCiForest have close relationships with the $\ell_1$ or Manhattan distance and an angular distance measure, respectively. As a result, the current use of the fast tree isolation mechanism is limited to these two distance measures, and fails to generalise to other commonly-used distance measures, data types or data spaces.

In this paper, we investigate the tree isolation mechanism for anomaly detection in a principled way and propose a generic detection framework named LSHiForest, with the use of the Locality-Sensitive Hashing (LSH) forest data structure [19]. The basic idea is that we make use of the salient LSH mechanism to achieve fast data instance isolation with the same logarithmic computational complexity as the existing methods [8], [18]. But more importantly, our approach can appropriately incorporate the nearest neighbourhood information of data instances into the isolation process, while such information used in iForest is rather coarse or limited. Another crucial impact is that our approach makes the isolation mechanism more versatile and flexible as it can be extended into a much wider scope given the recent fruitful achievements in LSH research [20]. Specifically, we formulate the algorithmic details of the generic framework for ensemble anomaly analysis. Then, iForest and SCiForest are revisited and formally shown to be special cases of our framework. Typical instances of the framework are developed for three different LSH families, i.e., the $\ell_p$ ($p = 1, 2$) LSH (for the Manhattan and Euclidean distances) [21], the angle-based LSH [22], and the kernelised LSH (for kernelised space) [23] to demonstrate the versatility of our approach. Note that the generic framework can be applied anywhere as an LSH family is defined, e.g., Min-hash LSH for set-valued or categorical data [24]. One can instantiate the framework with corresponding LSH families to detect anomalies in the desired distance space. Extensive experiments are conducted on both synthetic and real-world benchmark data sets, and the results show that our approach can significantly improve the detection performance and time efficiency of ensemble anomaly analysis over existing methods.

The main contributions of our approach are fourfold. First, a generic framework with LSH forest isolation is proposed for fast and versatile ensemble anomaly analysis. Then, we formally show that the existing tree isolation based methods are special cases of our framework. The framework is developed with several commonly-used LSH families to demonstrate its versatility. Finally, we conduct extensive experiments on both synthetic and real-world benchmark data sets to validate the effectiveness and efficiency of our approach.

The remainder of this paper is organised as follows. The next section reviews the related research work. In Section III, we describe the problem setting and discuss our research motivation. Our generic framework and its instantiations are formulated in Section IV, followed by extensive empirical evaluation on benchmark data sets in Section V. We conclude this paper and discuss future work in Section VI.

II. RELATED WORK

As pointed out in [11], a range of previous work attains ensemble anomaly analysis implicitly by combining anomaly scores from either multiple independent component models or dependent component models (single or multiple) running sequentially on the complete data. These are orthogonal to the ensemble method our approach follows herein, where different subsamples of the data are explored to perform anomaly analysis. Interested readers could refer to [11] for more details. In this section, we briefly survey the research work closely relevant to our approach, i.e., subspace or subsampling based ensemble anomaly analysis, subsampling based anomaly detection and locality-sensitive hashing (LSH) related outlier detection.

Lazarevic and Kumar [15] proposed the feature bagging technique for anomaly detection where the subspace ensemble method is explicitly and formally elaborated. This approach generates a number of subsamples of the data and combines the anomaly scores of data instances derived from these projected subspaces as the final scores. Zimek et al. [16] proposed a subsampling based approach for ensemble anomaly detection, which derives the anomaly scores of a data instance from subsamples of original data. It is reported in [16] that the subsampling based ensemble method can improve the quality of outlier detection and reduce the computational cost considerably compared with the feature bagging technique [15] or just the single component algorithm (LOF [6]) on the complete data. Furthermore, Aggarwal and Sathe [13] exploited the bias-variance trade-off theory [14] to formally underpin ensemble anomaly detection methods, and accordingly proposed to use fixed sublinear dimensionality for random subspaces and small constant subsample sizes, as opposed to the settings in [15] and [16], respectively. However, the evaluation stage in these methods can be still very time-consuming when the data scale is large because the time complexity of testing a single data instance is at least linear with respect to the sample size.
Liu et al. [8] proposed an isolation forest based ensemble anomaly detection approach (iForest), where a forest of isolation trees are constructed from training samples. The anomaly scores of a data instance are derived in terms of its path lengths in the trees. The basic idea of iForest is that anomalies often have shorter path lengths and can thus be identified. This method has a salient feature that the time complexity of evaluating a data instance is logarithmic with respect to the sample size, enabling it to handle large-scale data efficiently. However, it is pointed out in [17], [18] that iForest suffers from an inability to detect complicated anomalies such as axis-parallel, local and surrounded anomalies. The work in [18] extended iForest for clustered anomalies and proposed a variant SCiForest, which selects the best one out of a set of randomly generated subspace hyperplanes and the best splitting point corresponding to this hyperplane. The computational cost of the training stage increases accordingly because of the search of the best hyperplane and the best splitting point. Our work also follows the tree isolation mechanism, but our generic framework uses LSH forest to achieve isolation and can be instantiated with a wide range of LSH families. Also, note that iForest and SCiForest are special cases of our framework. Bandaragoda et al. [17] uses nearest neighbours to achieve hypersphere isolation for anomaly detection and proposed a method iNNE (isolation using Nearest Neighbour Ensemble) based on the relative density measure [6], aiming to detect local or surrounded anomalies. However, it is essentially similar to ensemble LOF [16] with a special setting and has the same high computational complexity. Furthermore, their experimental results fail to show a performance improvement over the ensemble LOF method. It is worth noting that the ensemble performance of average $k$-NN distance is better than that of LOF in most cases as empirically reported in [13], while the latter often performs better in the non-ensemble context. Aggarwal and Sathe [13] further explained this surprising phenomenon well by the bias-variance trade-off theory. As such, it still makes (or even more) sense to develop distance-based detection methods for ensemble anomaly analysis.

Sampling is a simple but effective technique for handling large-scale data, widely adopted in the database and data mining communities. Some sampling based anomaly detection methods without ensembles also exist. Kollois et al. [25] proposed a density biased sampling approach for distance-based outlier detection by oversampling sparse regions. But sampling just works as a pruning process without a rigorous treatment of the effect on result quality [26]. Wu and Jermaine [26] proposed a simple iterative sampling approach, where a sample is drawn for a data instance and the distance of the instance to its $k$th nearest neighbour ($k$-NN) in the sample is reported as its anomaly score. Similarly, Sugiyama and Borgwardt [27] proposed a detection approach using only one sample to estimate anomaly scores of all data instances in a data set. Our work is orthogonal to these methods and the sampling strategies can be complementarily applied to our ensemble method. Note that the performance of a base detector running on a complete data set is usually not as good as an ensemble of the base detector in terms of the bias-variance trade-off theory [13].

LSH has been recently exploited in anomaly detection tasks due to its salient features. Wang et al. [28] proposed a ranking based outlier detection approach using LSH to identify low-density regions for refining the data instances in these regions. Pillutla et al. [29] used LSH to prune strong inliers based on the redundancy of a point in hash tables such that fewer points are used to identify true outliers. The space cost of this approach is a problem for large data sets, as it requires constructing a number of hash tables for redundancy computation. In both methods, LSH only plays a secondary role as a pruning technique and its parameter tuning is critical to detection performance. In contrast, we leverage the LSH forest structure as the core part of our framework, and no parameter tuning is required for an LSH forest [19].

### III. Problem Statement

Our generic framework is based on the isolation notion proposed in [8], [18], i.e., separating a data instance from the rest in a sample. Hence, the assumption that anomalies are more susceptible to isolation applies herein as well. As the exact notion of an anomaly often varies considerably in different application domains, most existing detection methods only apply to a limited range of data types and anomaly types. To facilitate the discussion of our core idea, we use commonly-used numerical data and point anomalies as exemplary types in our work. The point anomaly concept implies that no relationship is assumed among data instances and an individual instance can be regarded as an anomaly. This type is widely used in the literature for anomaly analysis, and contextual or collective anomaly analysis where instances are correlated can often be reduced to a point anomaly problem [2]. We are concerned with unsupervised, non-parametric methods herein, which do not require labelled training data and are widely applicable. Accordingly, the output will be a ranked list of anomaly scores that allows analysts to determine the most relevant anomalies by a domain-specific threshold.

Our method follows the subsampling based ensemble framework formulated in [11], [13] which is formally underpinned by the bias-variance trade-off theory [14]. Note that all existing subsampling based ensemble methods [8], [16–18] also follow this framework, while varying in base detection algorithms and parameter settings. The ensemble anomaly analysis framework consists of four major stages, i.e., drawing a set of subsamples, training a model for each sample by a base algorithm, evaluating each data instance against the trained models, and combining the anomaly scores of an instance from all models as the final. As the performance of existing methods is sensitive to the selection of sample size, we adopt the variable subsampling technique proposed in [13] for our framework to produce subsamples of different sizes for ensemble components. Note that the sample sizes are bounded within a fixed interval irrespective of the entire data size. [50, 1000] used in [13]. The rationale is that it achieves variance reduction not only over the selection of training data, but also over different randomised choices of base detector parameters, e.g., $k$ in $k$-NN based anomaly detection. Another advantage is that the users’ burden of parameter selection is significantly relieved, while it is usually a challenging task to find the optimal parameters that are usually data-dependent as shown in [8], [13], [16], [17]. Furthermore, we use the widely-adopted averaging method for anomaly score combination. Note that other subsampling, subspace or score combination techniques discussed in [13] can also work well with our generic framework, as our work aims to propose a novel
category of base detection methods for ensemble. We describe
the symbols and notations used throughout the paper in Table I.

<table>
<thead>
<tr>
<th>x</th>
<th>a data instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>a data set</td>
</tr>
<tr>
<td>n</td>
<td>number of data instances in a data set, $n =</td>
</tr>
<tr>
<td>m</td>
<td>number of dimensions</td>
</tr>
<tr>
<td>t</td>
<td>number of ensemble components</td>
</tr>
<tr>
<td>S</td>
<td>a subsample of $X$, $S \subseteq X$</td>
</tr>
<tr>
<td>$\psi$</td>
<td>sample size</td>
</tr>
<tr>
<td>k</td>
<td>the number of nearest neighbours</td>
</tr>
<tr>
<td>$\eta$</td>
<td>granularity adjustment factor</td>
</tr>
<tr>
<td>$\nu$</td>
<td>branching factor</td>
</tr>
<tr>
<td>$f(\cdot)$</td>
<td>hash function</td>
</tr>
<tr>
<td>$F$</td>
<td>hash function family</td>
</tr>
<tr>
<td>$h(\cdot)$</td>
<td>path length</td>
</tr>
<tr>
<td>$H$</td>
<td>tree height</td>
</tr>
</tbody>
</table>

The average-case time complexity of existing subsampling
based ensemble anomaly analysis methods is presented in
Table II to illustrate the computational advantages of forest
based isolation. We use the average-case complexity herein in
order to estimate the performance more tightly as the worst-
case behaviours rarely occur in practice. Note that the worst-
case complexity of the first three methods is the same as
the average case, while that of the last two is linear with
respect to the sample size $\psi$. For SCiForest, $\tau$ is the number
of hyperplanes tried in each split and $q, 1 \leq q \leq m$ is the number
attributes used to construct a subspace hyperplane [18].
An important observation is that the prediction stage will
dominate the execution time when a data set is very large,
because $n$ is involved in this stage. Due to the computational
advantages of subsampling based ensembles, all methods have
linear time complexity with respect to $n$. Most importantly,
the tree-isoation based methods take only logarithmic complexity
with respect to $\psi$, while others take linear complexity. This
difference makes the former much faster, especially when $n$
is very large. For instance, the speed-up can be 2 orders
of magnitude when $\psi = 1000$. Therefore, the tree isolation
mechanism offers great computational benefits.

Given the appealing computational benefits of the tree
isoation based anomaly analysis and the aforementioned
limitations of the existing methods, an interesting question
arises: Can we further extend the fast isolation mechanism
to other commonly-used distance or similarity measures (e.g.,
the Euclidean distance), data spaces (e.g., kernelised space)
or data types (e.g., categorical data), with the expectation of
increasing the versatility of the isolation based anomaly anal-
ysis and improving the detection performance while retaining
the low logarithmic computational complexity? This question
motivates our research herein, and we show that the answer
is yes. Concretely, we propose a generic framework for fast
ensemble anomaly analysis based on LSH forest isolation,
which significantly extends the fast isolation mechanism into
a much wider scope via LSH families, while the existing tree
isoation based methods are only applicable to the Manhattan
($\ell_1$) or angular distances on numerical data.

IV. ANOMALY DETECTION USING LSH FOREST

A. LSH Forest Preliminary

The core idea of Locality Sensitive Hashing (LSH) is to
hash ‘nearby’ data instances into the same bucket with higher
probability than instances that are far apart to construct indexes
for efficient similarity search [30]. Given a distance metric $d(\cdot,\cdot)$, let $d_1 < d_2$ be two distance values, and $p_1, p_2, 0 \leq p_1, p_2 \leq 1$ be two probability values. A family of functions $F$ is said to be $(d_1, d_2, p_1, p_2)$-sensitive if for any function $f \in F$, $x, y \in X$, the following conditions hold:

1. $d(x, y) \leq d_1 \Rightarrow Pr[f(x) = f(y)] \geq p_1$;
2. $d(x, y) \geq d_2 \Rightarrow Pr[f(x) = f(y)] \leq p_2$.

The event $f(x) = f(y)$ means $x$ and $y$ are hashed into
the same bucket in a hash table. It is often required that
$p_1 > p_2$ to make an LSH family useful. One salient feature
of LSH is that we can drive the probability that instances that are
far apart to construct indexes for efficient similarity search.
The core idea is that the combined key of each data instance is
made sufficiently long to ensure each instance has a distinct
key. Another salient feature is that the combined key of each data instance is
made sufficiently long to ensure each instance has a distinct
key.

As the similarity search quality is sensitive to $\alpha$, a non-
trivial task of parameter tuning is often required to obtain
good performance [19]. Accordingly, a data structure named
LSH forest is proposed in [19] to mitigate the parameter
tuning burden by using variable-length combined keys. The
core idea is that the combined key of each data instance is
made sufficiently long to ensure each instance has a distinct
key. Concretely, a (logical) prefix tree named an LSH tree is
constructed on the set of all combined keys, where each leaf

<table>
<thead>
<tr>
<th>Method</th>
<th>Training</th>
<th>Predicting</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnLOF [13], [16]</td>
<td>N/A</td>
<td>$\Theta(ntk^2\psi m)$</td>
</tr>
<tr>
<td>EnKNN [13]</td>
<td>N/A</td>
<td>$\Theta(ntk\psi m)$</td>
</tr>
<tr>
<td>iNNE [17]</td>
<td>$\Theta(t\psi^\alpha m)$</td>
<td>$\Theta(nt\psi m)$</td>
</tr>
<tr>
<td>tForest [8]</td>
<td>$\Theta(t\psi\log_2 \psi)$</td>
<td>$\Theta(nt\log_2 \psi)$</td>
</tr>
<tr>
<td>SCiForest [18]</td>
<td>$\Theta(tr\psi^\alpha\log_2 \psi q)$</td>
<td>$\Theta(nt\log_2 \psi q)$</td>
</tr>
</tbody>
</table>

TABLE II: Average-case time complexity of existing methods.
node corresponds to a data instance. A hash function from an LSH family is used at an internal node to produce different path tags to guide data partitioning. The path tags from the root to a leaf node compose the combined key of the corresponding data instance. A forest of LSH trees are often constructed for robust approximate nearest neighbour search.

B. Generic Framework: LSHiForest

From the perspective of isolation, an LSH tree can be also deemed as an isolation tree in essence because each data instance is isolated from the rest. Therefore, an LSH forest can be leveraged seamlessly for isolation based ensemble anomaly analysis in the same way as the random binary forest used iForest and SCiForest. More importantly, the fast isolation mechanism can be extended to a much wider scope as an LSH forest can work with any LSH family. This makes our framework very versatile as we can incorporate into the isolation process any similarity measure in any data space where an LSH family is defined. Like in iForest, our generic framework also consists of two stages: the training stage constructs LSH forest for subsamples from a given data set, and the evaluation stage computes anomaly scores of data instances from the built LSH forest.

Given an LSH family, LSH trees are constructed in the training stage by recursively partitioning a subsample $S \subseteq X$ until all data instances are isolated. Algorithm 1 describes how to build a forest of LSH trees. To build a tree, we draw a subsample $S_i$ for the input data in Line 2 by variable subsampling as described in [13]. Specifically, the sampling rate follows the uniform distribution $U(\min\{1, \frac{\log_2(n)}{n}\}, \max\{1, \frac{\log_2(n)}{n}\})$, i.e., $50 \leq \psi \leq 1000$, or $\psi = n$ if $n < 50$. However, tree heights generated based on this subsampling technique fail to follow the uniform distribution and prefer larger height values. Therefore, we propose to use a new subsampling technique that engenders uniformly distributed tree heights. Our sampling rate is $\min\{1, \frac{\log_2(n)}{n}\}$, where $s$ follows the uniform distribution $U(6, 10)$. Note that $\psi$ will vary in $[64, 1024]$ if $n \geq 64$, which is quite similar to $[50, 1000]$ used in [13]. Our sampling technique can generate more diversity for an ensemble because the tree height distribution is more balanced. It can be proved that the variance of the tree height in our technique is $1.3333$ while that in the previous one is $0.94375$ if we roughly assume the tree height is approximately $\log_2(\psi)$ and $64 \leq \psi \leq 1024$. Another benefit is that we can reduce the computational cost as it can also be proved that the sample size expectation $E(\psi)$ in our technique is around $346$, while that in the previous one is $544$.

Algorithm 1 Constructing LSH Forest

Input: Data set $X$; LSH family $F$; number of LSH trees $t$.
Output: A forest of LSH trees $\{T_i\}_{1 \leq i \leq t}$.

1: for $i, 1 \leq i \leq t$ do
2: $S_i \leftarrow \text{variable_subsampling}(X)$;
3: Compute a height limit: $H_i(\psi_i)$;
4: $T_i \leftarrow \text{LSHITree}(S_i, F, H_i, 0)$;
5: return $\{T_i\}$.

In Line 3, we compute a height limit $H(\psi)$ to prevent an LSH tree from growing arbitrarily large if two data instances are identical or very close. A long (unbounded) single-branch path can be generated because an LSH hash function produces the same key for two identical instances and excessive hash functions are often required to distinguish two very close instances. Note that iForest also suffers from the same problem for identical instances. From the perspective of computational cost, it is necessary to bound the height of an LSH tree. The height limitation poses little influence on the quality of anomaly detection, because the instances with greater depths are usually strong inliers rather than outliers. Besides, single-branch paths are compressed such that an anomalous instance with such a long path can also be identified by the detection algorithm. But note that the height should be long enough to make instances sufficiently isolated, requiring the upper bound to be as tight as possible. As an LSH tree can be deemed as a digital trie [19], we can use the average height of a digital trie to estimate the height limit. Let us consider random digital tries with $n$ data instances, over a digit alphabet $\{K_1, \cdots, K_v\}$, $v \geq 2$. To compose a sequence of digits for instance indexing, $K_i$ is chosen for an element of the sequence with probability $p_i, 1 \leq i \leq v$. In terms of [31], the average height of the digital tries is around $\frac{\chi}{v} \ln(n) + \frac{1}{v} \ln(2) + 1$, where $\gamma \approx 0.5772$ is the Euler constant and $g = \frac{1}{\ln(v)} \sum_{i=1}^{v} p_i^\chi$. We assume herein that all instances in $X$ are independent, and the hash keys generated by an LSH family are also independent and of equal probability, i.e., $p_i = \frac{1}{v}$. Then, an appropriate upper bound for $H(\psi)$ can be

$$H(\psi) \approx \frac{2 \ln(\psi)}{\ln(v)} + \frac{\chi - \ln(2)}{\ln(v)} + 1 \leq 2 \log_2(\psi) + 0.8327. \quad (1)$$

The tree construction subroutine in Line 3 of Algorithm 1 is a recursive process, detailed in Algorithm 2. At each step of partitioning, a hash function $f_i$ from the LSH family is used to produce hash values for all data instances, corresponding to the subroutine $\text{lsjht_split}(S, f_i)$ in Line 6 and 9. The input data are partitioned into non-overlapping subsets associated with hash keys, i.e., $\cup_{i=1}^{v} S_i = S$ and $f_i(x) = K_i$ if and only if $x \in S_i$. To produce a compact tree without single-branch paths, we repeat splitting the input data until generating multiple hash keys or reaching the tree height limit. The compact tree can be further regarded as a PATRICIA trie [19]. As the hash function index $I$ increases accordingly, it records the path length information of a data instance in the uncompressed digital trie, which is thereby retained for the anomaly prediction stage.

After constructing a forest of LSH trees in the training stage, we can evaluate the anomaly scores of instances in a data set, as described in Algorithm 3. For a data instance $x \in X$, we evaluate its path length $h_i(x)$ against the LSH tree $T_i$ in Line 4 via the subroutine $\text{path_length}$, which is elaborated in Algorithm 4. To make path lengths from different trees comparable, we normalise $h_i(x)$ based on a reference path length denoted as $\mu(\psi)$ in Line 5. Unlike iForest which uses a single reference value for all trees, different reference values are required for different trees in our approach because the trees are of different sizes. The reference path length used in iForest is the average path lengths of unsuccessful searches in a binary search tree. However, this fails to apply in our case because the branching factor $v$ is often greater than 2. The estimation of average $h(x)$ for external node terminations is the same as that of the successful searches in PATRICIA tries. According to the analysis in [32], the
Algorithm 2 Constructing LSH Tree: LSHiTree($S, F, H, I$)

Input: Input data set $S$; LSH family $F$; height limit $H$; index $I$.
Output: An LSH trees $T$.
1: if $|S| = 0$ then
2: return NULL;
3: else if $|S| = 1$ OR $I > H$ then
4: return node($Size ← |S|$, Hash_Index ← I, Children ← $\emptyset$);
5: else
6: $\{K_1 : S_1, \cdots, K_v : S_v\} ←$ lsh_split($S, f_j$), $f_j \in F$
7: while $v = 1$ AND $I ≤ H$ do
8: $I ← I + 1$
9: $\{K_1 : S_1, \cdots, K_v : S_v\} ←$ lsh_split($S, f_j$);
10: if $I > H$ then
11: return node($Size ← |S|$, Hash_Index ← I);
12: Initialise child node indexing: Convert $\emptyset$;
13: for $i, 1 ≤ i ≤ v$ do
14: $C_i ← LSHiTree(S_i, F, H, I + 1)$
15: $C ← \bigcup \{C_i\}$
16: return node($Size ← |S|$, Hash_Index ← I, Children ← $C$).

The expectation of successful search in PATRICIA tries indexing $\psi$ data instances is approximately $\frac{1}{2}(\ln(\psi) + \gamma_1 + g_1 + \frac{\ln(\psi) - 1}{2})$, where $g_1 = -\sum_{i=1}^{v} pr_i \ln(pr_i)$, $g_2 = -\sum_{i=1}^{v} pr_i \ln^2(pr_i)$, and $g_1 = -\sum_{i=1}^{v} pr_i \ln(1 - pr_i)$. Under the same assumption as the height estimation, i.e., $p_i = \frac{1}{\psi}$, we approximate $\mu(\psi)$ by
\[
\mu(\psi) = \begin{cases} 
\frac{\ln(\psi) + \ln(\psi - 1) + \gamma}{\ln(\psi)}, & 1 < \psi \leq e^3/2, \\
1, & \psi = 1 \\
0, & \text{otherwise}.
\end{cases}
\] (2)

Since the information about $v$ is unknown, we need to estimate it from trained LSH trees. Specifically, we can estimate $v$ for a tree by the average branching factor.

Algorithm 3 Predicating Anomaly Scores

Input: Test data $X$; LSH family $F$; LSH forest $\{T_i | 1 ≤ i ≤ T\}$; depth adjustment factor $\eta$; granularity level $L$.
Output: Anomaly scores $\{A_{S_x} | x \in X\}$.
1: for $x \in X$ do
2: $A_{S_x} ← 0$
3: for $i, 1 ≤ i ≤ T$ do
4: $h(x) ←$ path_length($x, F, T_i.root, \eta, L, 0$);
5: Normalisation: $A_{S_x} ← A_{S_x} + 2^{-h(x) / \ln(2)}$
6: $A_{S_x} ← A_{S_x} / t$
7: return $\{A_{S_x}\}$.

Like in iForest, we further use the exponential function $2^{-x}$, $x ≥ 0$ to non-linearly scale the normalised path lengths into the interval (0, 1] in Line 5 of Algorithm 3. But note that the non-linear scaling in iForest fails to affect the order of final anomaly scores because it occurs after the combination of normalised path lengths from ensemble components. In contrast, we conduct the non-linear scaling before the combination, making it contribute to improving diversity for an ensemble. In Line 6, we take the arithmetic mean of non-linearly scaled values as the final anomaly score, i.e., $A_{S_x} = \frac{1}{T} \sum_{i=1}^{T} 2^{-h(x) / \ln(2)}$, In fact, a final anomaly score from iForest can be regarded as the geometric mean of non-linearly scaled values, i.e., $2^{-\frac{1}{T} \sum_{i=1}^{T} h_i(x) / \ln(2)} = \sqrt[\ln(2)]{\prod_{i=1}^{T} 2^{-h_i(x)}}$, where $c(\psi)$ is the reference path length similar to $\mu(\psi)$. Although non-linear scaling can be deemed to be involved in the ensemble of iForest, its effect is cancelled by the geometric mean combination. Different anomaly score combination methods have been studied in [13], out of which a method named AOM (Average of Maximum) is shown to be a well-balanced choice and perform very well. Our method can be regarded as a variant of AOM without grouping scores from base detectors. As the non-linear scaling increases the contrast between big values and small values, the arithmetic mean of the scaled values can be dominated by originally big values to a large extent. The extent is controlled by the non-linear function.

Algorithm 4 Subroutine: path_length($x, F, node, \eta, L, I_{cur}$)

Input: Data instance $x$; LSH family $F$; current node $node$; depth adjustment factor $\eta$; granularity level $L$; index $I$.
Output: The depth of $x$ in the subtree with node as the root.
1: if node = NULL then
2: return -1;
3: else if node.Children = $\emptyset$ OR $I_{cur} > L$ then
4: return $I_{cur} + \eta \cdot \mu(node.Size)$;
5: else
6: $K ← I_{Hash.Index}(x)$. $I_{Hash.Index} \in F$
7: if $\exists (K_i : C_i) \subseteq$ node.Children AND $K = K_i$ then
8: return path_length($x, F, C_i, \eta, L, I_{cur} + 1$);
9: else
10: return $(I_{cur} + 1) \cdot \mu(node.HashIndex + 1)$;

Algorithm 4 details path_length($x, F, node, \eta, L, I_{cur}$), the recursive subroutine for computing the path length of the instance $x$ from node to the corresponding leaf node. When $I_{cur} = 0$, we can obtain $h(x)$ with respect to a whole tree. The depth level $L$, $1 ≤ L ≤ H$, is a user-defined parameter for limiting traversal depth, the same as the height limit in iForest. Isolated anomaly clusters can be detected by lowering the depth level as shown in [8]. Also, we use the term $\mu(node.Size)$ to adjust the path length in Line 4 because a leaf node or an internal node with the depth equal to $L$ may correspond to multiple data instances. The rationale is that the data instances in a larger subtree are more normal than those in a small subtree at the same depth.

Our approach is different from iForest in several aspects. The traversal can stop at both external and internal nodes in our path length computation, as shown in Line 4 and Line 10 of Algorithm 4, while it can only stop at external nodes in iForest. This implies a stronger detection ability of LSHiForest because dissimilar data instances (potential anomalies) will have much shorter path lengths. Another difference is that path lengths of $x$ from both the digital (uncompressed) trie and the PATRICIA (compressed) trie are combined into a single path length, making our method more flexible. Specifically, $h(x) = h_c(h_u)^{\eta}$, where $h_c$ and $h_n$ are the compressed and uncompressed path lengths, respectively, and $\eta$ is an adjustment factor ranging in $[0, 1]$. The parameter $\eta$ controls the granularity of preserving the distance information in the space defined for the LSH family. When $\eta = 1$, $h(x) = h_u$, implying the finest granularity of isolation is used and the detector can have the best performance for global anomalies. For local anomalies, however, it is desirable to weaken such granularity.
because the distances among points in a sparse normal region can be larger than the distance between an anomaly and a dense normal region. Hence, a smaller \( \eta \) can be used for local anomaly detection. Note that this is still insufficient to address the problem of inability to detect local anomalies, but sheds some light on the solutions. A promising solution is to transform the data into another space and apply the isolation based anomaly detection in the transformed space. An instance of our framework in the kernelised space is described in Section IV-D.

With respect to the sample size \( \psi \), the computational complexity of LSHiForest is similar to iForest. Specifically, the average-case time complexity in the training stage is \( \Theta(\psi \log_\psi(\psi)) \) and that in the evaluation stage is \( \Theta(\log_\psi(\psi)) \). The overall time complexity of the instances with specific LSH families is analysed in Section IV-D. As most LSH functions have \( O(1) \) time complexity, our framework can be very fast. Besides, it is very versatile by working with various distance metrics, data spaces and types where an LSH family is defined, and able to detect diverse types of anomaly.

### C. Revisiting iForest and SCiForest

In this section, we revisit the two existing tree isolation based anomaly detection methods, i.e., iForest and SCiForest, to study their relationship to our generic framework. Contrary to the claim that these two methods never rely on any distance measure [8], [18], we find that they are indeed associated with certain distance measures. Interestingly, iForest can be directly regarded as an instance of our framework under the \( \ell_1 \) (Manhattan) distance. This also applies to SCiForest under an angular distance without considering its optimisation part.

We first discuss the iForest case. Let \( x = (x_1, \ldots, x_m) \) and \( y = (y_1, \ldots, y_m) \) be two data instances in \( X \). Without loss of generality, we assume that all attributes of \( X \) have been normalised into the range \([0, 1]\). Formally, we use the randomised function \( f : [0, 1]^m \rightarrow \{-1, 1\} \) to denote the decision-making at each step in iForest, i.e., if \( f(x) = -1 \), \( x \) will be put into the left partition, or else it will be put into the right one. As iForest uniformly randomly chooses an attribute for splitting, \( \Pr[f(x) = f(y)] = \sum_{i=1}^m \Pr[\text{the } i^{th} \text{ attribute is chosen}] \Pr[f_i(x_i) = f_i(y_i)] = \frac{1}{m} \sum_{i=1}^m \Pr[f_i(x_i) = f_i(y_i)], \) where \( f_i : [0, 1] \rightarrow \{-1, 1\} \) is the randomised function corresponding to the decision after a splitting point \( \omega \in [0, 1] \) is given for the \( i^{th} \) attribute. Specifically, \( f_i(x) = \text{sgn}(x - \omega_i) \), where the sign function \( \text{sgn}(x) = -1 \) if \( x < 0 \) or \( 1 \) if \( x > 0 \). Then, we have \( \Pr[f_i(x_i) = f_i(y_i)] = 1 - \Pr[\omega_i \text{ lies between } x_i \text{ and } y_i] = 1 - |x_i - y_i| \). Finally, we have the following formula:

\[
\Pr[f(x) = f(y)] = 1 - \frac{1}{m} \sum_{i=1}^m |x_i - y_i|. \tag{3}
\]

Note that \( \sum_{i=1}^m |x_i - y_i| \) is the \( \ell_1 \) distance \(|x - y|_1\), a.k.a. the Manhattan distance. Let \( d(x, y) \) be the \( \ell_1 \) distance between \( x \) and \( y \). The randomised function \( f(x) \) is \((d_1, d_2, 1 - \frac{1}{m} d_1, 1 - \frac{1}{m} d_2)\)-sensitive. Therefore, the function family consisting of such hash functions is an LSH family. In essence, iForest is an instance of our framework with such an LSH family.

Let us consider SCiForest, which is essentially an extension of iForest. To avoid axis-parallel partitioning, SCiForest allows using more than one attribute in a splitting step. A randomised function \( Q : \mathbb{R}^m \rightarrow \{-1, 1\} \) is defined over the \( q \) attributes for data partitioning. Specifically, \( Q(x) = \text{sgn}(\sum_{i=1}^m \omega_i x_i + \omega_0) \), where the coefficient \( \omega_0 \) is chosen randomly from \([-1, 1]\) and \( \omega_i \), called the best splitting point, is determined by an optimisation process. To study the probability \( \Pr[Q(x) = Q(y)] \), we rewrite \( Q(x) \) as \( Q(x) = \text{sgn}(\langle \omega, x \rangle) \), where \( \omega = (\omega_0, \omega_1, \ldots, \omega_m) \) and \( \theta = (x_1, x_2, \ldots, x_m) \). Then, \( \Pr[Q(x) = Q(y)] = \Pr[\text{sgn}(\langle \omega, x \rangle) = \text{sgn}(\langle \omega, y \rangle)] \), where \( \theta(x, \dot{x}) \) is the angle between the two vectors \( \dot{x} \) and \( \dot{x} \). If we define the distance between \( x \) and \( y \) as \( d(x, y) = \theta(x, \dot{x}) \), the randomised function \( Q(x) \) forms an LSH family based the distance and \( \Pr[Q(x) = Q(y)] = 1 - \frac{1}{\theta_{\max}} \theta(x, y) \). However, it is noted that \( \theta_{\max} < \pi \) rather than \( \theta_{\max} = \pi \) in the seminal case [22]. This difference makes choosing \( \omega \) a non-trivial task, while \( \omega_0 \) can be drawn from the standard Gaussian distribution \( \mathcal{N}(0, 1) \) for the seminal angle-based LSH family [22]. Since SCiForest fails to work out how to set the value of \( \omega_0 \) when letting \( \omega_i \), \( 1 \leq i \leq q \) vary in \([-1, 1]\), it resorts to an optimisation technique. The randomised functions with the optimisation process can be essentially regarded as a learning based hashing scheme, which is data-dependent and has a computational price [20]. Our generic framework can work with learning based hashing schemes without any significant changes.

To show the difficulty in selecting \( \omega_i \) (\( 0 \leq i \leq q \)) for SCiForest without optimisation, we explore the basic cases when \( q = 1 \) (similar to iForest) and \( q = 2 \) (the default setting in [18]). Assume all attributes are normalised to \([0, 1]\), Fig 1(a) illustrates the constraints between \( \omega_0 \) and \( \omega_1 \) when \( q = 1 \). As the normal vector of the hyperplane \( \omega_0 x_1 + \omega_1 x_2 = 0 \), \( (\omega_0, \omega_1) \) must lie within the two hatched triangles in order to make sure the hyperplane varies within \([\frac{\pi}{4}, \frac{3\pi}{4}]\), i.e., \( \theta_{\max} = \frac{\pi}{2} \). Or formally, the inequality \( (\omega_0 + \omega_1) \omega_1 \geq 0, \omega_0, \omega_1 \in [-1, 1] \) should hold. The case for \( q = 2 \) is more complicated. The data space is a square region, the base surface of the inverted pyramid as shown in Fig 1(b), where we centre the region by normalising each attribute into \([-\frac{1}{2}, \frac{1}{2}]\) for better visualisation. Then, \( (\omega_0, \omega_1, \omega_2) \) must lie within the other four pyramids whose bases are perpendicular to the \( X_1 \) or \( X_2 \) axis. Hence, it can be seen that it is a non-trivial task to randomly select \( \omega \) to make \( Q(x) \) an LSH hash function. The optimisation technique in [18] mitigates this problem. Note that choosing inappropriate parameters can lead to insufficient distinguishability of hash functions. For example, if the hyperplane in Fig 1(a) can rotate out of \([\frac{\pi}{4}, \frac{3\pi}{4}]\), the probability of hashing two data instances into the same bucket cannot be arbitrarily small, which violates the second condition of a locality-sensitive hashing family.

Finally, we have the probability of putting two data instances \( x, y \in [0, 1]^m \) into the same partition as \( \Pr[f(x) = f(y)] = \frac{1}{M} \sum_{i=1}^M \Pr[Q_i(x_1, \ldots, x_m) = Q_i(y_1, \ldots, y_m)], \) where \( M = \binom{m}{q} \). Based on the above analysis on \( Q(x) \), we have

\[
\Pr[f(x) = f(y)] = 1 - \frac{1}{m} \sum_{i=1}^M \frac{1}{\theta_{\max}} \theta(x_i, y_i). \tag{4}
\]

If we define \( d(x, y) = \sum_{i=1}^M \frac{1}{\theta_{\max}} \theta(x_i, y_i) \) as the distance
between \( x \) and \( y \), the randomised function \( f(x) \) is \( (d_1, d_2, 1 - \frac{1}{\pi} d_1, 1 - \frac{1}{\pi} d_2) \)-sensitive and forms an LSH family.

### D. Instantiations with Various LSH Families

Besides iForest and SCiForest, in this section we show more instantiations of our generic framework with several commonly-used LSH families. Typically, the LSH families with the \( \ell_p \) (\( p = 1, 2 \)) distance and the angle-based distance in both original and kernelised spaces will be discussed herein.

Unlike iForest and SCiForest, we can have multi-fork LSH trees by using the LSH family under the \( \ell_p \) distance. When \( p = 1 \), the distance is the \( \ell_1 \) or Manhattan distance which is also associated with iForest, and when \( p = 2 \), it is the Euclidean distance commonly used in most existing detection methods [5], [6], [13], [16], [17]. The LSH scheme can be achieved by the following randomised hash functions [21]:

\[
f_{\omega, \omega_0}(x) = \left[ \frac{\omega^T x + \omega_0}{W} \right],
\]

where \( \omega \) is a random vector with components drawn from the \( p \)-stable distribution independently, \( \omega_0 \) is a real number uniformly sampled from \([0, W]\) and \( W \) is a user-specified parameter for the hashing bucket size. Note that the 1-stable distribution is the Cauchy distribution with the density function \( c(x) = \frac{1}{\pi(1 + x^2)} \), and the 2-stable distribution is the Gaussian (normal) distribution with the density function \( g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \). Although the parameter \( W \) can affect the performance of approximate nearest neighbour search, it poses little influence on our approach for anomaly detection because our aim is to isolate data instances, which can be always achieved even if \( W \) changes within a broad range of reasonable values.

Although the distance used in SCiForest is also angular, we prefer the seminal one due to its much easier parameter selection. The angle between vector \( x \) and \( y \) is conventionally defined as \( \theta(x, y) = \arccos(\frac{x^T y}{||x||_2||y||_2}) \). The corresponding LSH function is \( f_{\omega, \omega_0}(x) = \text{sgn}(\omega^T x) \) [22], where \( \omega_i (1 \leq i \leq m) \) is chosen from the standard normal distribution \( N(0, 1) \).

The iForest method has been criticised due to its inability to detect local anomalies [17]. As to our framework, in general its ability to detect local anomalies relies on the specific LSH families although decreasing the adjustment factor \( \eta \) can improve the capability of detecting local anomalies. To enable the framework to detect local anomalies, we propose to instantiate the framework with kernelised LSH families. Concretely, we apply the tree isolation mechanism in a kernelised space after data are transformed into the space. The rationale is that local anomalies in the original space can be mapped as global anomalies in the kernelised space and become more susceptible to be isolated and detected.

The key is to have a kernelised LSH family. Fortunately, a kernelised angle-based LSH family has been proposed recently [23]. Let \( \phi(x) \) and \( \kappa(\cdot, \cdot) \) be the map function and the kernel function, respectively. Then, the LSH function is defined as

\[
f(\phi(x)) = \text{sgn}\left( \sum_{i=1}^{\lambda} \omega(i) \kappa(x, \hat{x}_i) \right),
\]

where \( \omega = \bar{K}^{-1/2} \xi, \bar{K} \) is the \( \lambda \times \lambda \) centred kernel matrix formed by \( \{\hat{x}_1, \cdots, \hat{x}_\lambda\} \) and \( \xi \) is an \( \lambda \times 1 \) vector with ones at entries corresponding to the \( \xi \) samples. To obtain a family of hash functions, we draw a sample of size \( \lambda \) from the original data set to form the kernel matrix. Note that \( \lambda \) is independent of the sample size \( \psi \) for ensemble. Then, \( \xi \) entries out of \( \lambda \) are selected for random projection, resulting in \( \binom{\lambda}{\psi} \) possible projections. The parameters \( \lambda \) and \( \xi \) and the kernel function type will affect the similarity search quality and time efficiency. In practice, \( \lambda \) and \( \xi \) can be small enough to achieve fast and high-quality similarity search, e.g., \( \lambda = \sqrt{\pi} \) and \( \xi = 30 \) [23]. The choice of the kernel function is data-dependent. To produce multi-fork LSH trees, we can use multiple random vectors \( [\xi_1, \cdots, \xi_n] \) to generate a combined hash key of \( u \) bits, leading to the possible maximum branching factor \( 2^u, u \geq 1 \).

Table III shows the time complexity of the above three instances of our framework. The time complexity of the instances with \( \ell_p \) (\( p = 1, 2 \)) LSH and angle-based LSH is similar to that of iForest, except the dimension factor \( m \). The training time complexity of a kernelised LSH tree consists of three parts, i.e., building the LSH family \( (\Theta(\lambda^2 m + \lambda^3)) \), transforming training samples into the kernel space \( (\psi \log_\psi(\psi \lambda m)) \) and constructing the LSH tree \( (\psi \log_\psi(\lambda \psi m)) \). Similarly, the time complexity of evaluating a data instance on a kernelised LSH tree consists of two parts, i.e., transforming the data instance into the kernel space \( (\lambda \psi m) \) and testing the kernelised data instance on the tree \( (\log_\psi(\lambda \psi m)) \). Note that all the instances are logarithmic with respect to \( \psi \).

### TABLE III: Average-case time complexity of instances.

<table>
<thead>
<tr>
<th>Method</th>
<th>Training</th>
<th>Predicting</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_1, \ell_2 )</td>
<td>( \Theta((x \log_\psi(\psi m))m) )</td>
<td>( \Theta((m \log_\psi(\psi m))m) )</td>
</tr>
<tr>
<td>Angle</td>
<td>( \Theta((x \log_\psi(\lambda \psi m))m) )</td>
<td>( \Theta((m \log_\psi(\lambda \psi m))m) )</td>
</tr>
<tr>
<td>Kernel</td>
<td>( \Theta((\lambda^2 m + \lambda^3 + \psi \log_\psi(\lambda \psi m))) )</td>
<td>( \Theta((m \log_\psi(\lambda \psi m))m) )</td>
</tr>
</tbody>
</table>

One question might be asked: Which is the best LSH family for anomaly detection? The answer is that no LSH family can always beat others and the detection quality is often data-dependent and anomaly-dependent. One key reason is that in real applications anomalies are often defined based on domain-specific knowledge and are associated with certain distance or similarity measures. Therefore, it is necessary to have a diverse range of schemes of the generic framework to cater for different real-world scenarios. Besides the instantiations discussed herein, the framework can be applied anywhere an LSH family is defined. As an increasing number of LSH families have been...
developed recently [20], it is quite promising to instantiate our framework accordingly for anomaly detection in the desired distance space for specific applications.

V. EMPIRICAL EVALUATION

A. Experiment Settings

To study the effectiveness and efficiency of our framework against the state-of-the-art methods, in this section we conduct extensive experiments on both synthetic and real-world benchmark data sets. Specifically, we evaluate the aforementioned four instances of our framework, i.e., the angle-based LSH instantiation (ALSH), the \( l_p \) \((p = 1, 2)\) LSH instantiations (L1SH and L2SH), and the kernelised LSH instantiation (KLSH). To make a comprehensive comparison, we compare our approach with all state-of-the-art subsampling ensemble based anomaly detection methods, i.e., iForest (ISO) [8], SciForest (SCI) [18], average \( k\)-NN distance ensemble (EnKNN) [13], LOF ensemble (EnLOF) [13], [16] and iNNE [17].

We set the parameters as follows. For all methods, the number of ensemble members \( t \) is set as 100, the same as that used in the existing methods. We use the subsampling technique discussed in Section IV-B for all methods and the sample size varies within the range [64, 1024] which is very close to that used in [13]. For \( \ell_p \) LSH, the bucket size parameter \( W \) in (5) is set as 4, the same as the default value used in [21]. We employ the commonly-used RBF (Radial Basis Function) kernel for KLSH. Its performance is sensitive to its bandwidth parameter \( \gamma \) and the optimal selection is often data-dependent. We set \( \gamma \) as \( 10^4 \) herein to make use of the ensemble mechanism to reduce the sensitivity, where \( s \) is chosen from the uniform distribution \( U(\log_{10}\left(\frac{b}{m}\right),\log_{10}(\frac{b}{m})) \), \( m \) is the number of dimensions. The other KLSH parameters \( \lambda \) and \( \xi \) in KLSH are set differently for different groups of experiments. We use the same default parameter settings for SciForest as [18], i.e, \( q = 2 \) and \( \tau = 10 \). Instead of using a fixed value for the parameter \( k \) in EnKNN and EnLOF [13, 16], we let it vary within \([1, 10]\) for each subensemble. In this way, we again benefit from the ensemble mechanism to avoid parameter selection for \( k \), whose optimal value is often data-dependent [13].

All algorithms are implemented in Python and the source code can be found at https://dl.dropboxusercontent.com/u/55237646/LSHiForest-code.zip. All experiments are executed on a virtual machine with 4 GB main memory, 1 virtual processor and Linux installed as the guest operating system. We gauge the AUC (Area under Curve) and the execution time of each method as the performance indicators. AUC, the area under a ROC curve, is an effective quality measure for anomaly detection and has been extensively used in the literature [8], [13], [16], [17]. A higher AUC value means a better detection quality. The averaged experiment results over 10 runs of each method are reported.

B. Experiment Process and Results

Four groups of experiments are conducted on both synthetic and real-world data sets. We first evaluate how the branching factor \( v \) of an LSH forest affects the detection quality. In the second and third groups, we compare the instances of our framework with other methods in the ability to detect surrounded anomalies and local anomalies. Finally, a variety of benchmark data sets are used to evaluate detection quality and efficiency of all the aforementioned methods.

1) Effects of Branching Factor: Unlike iForest and SciForest which always produce binary trees, some instances of our framework can produce multi-fork branches, relying on the corresponding LSH families. To study how the branching factor affects the detection quality, we use KLSH with the adjustment factor \( \eta = 1 \) as a representative for evaluation. The KLSH parameters \( \lambda \) and \( \xi \) are set as \( 300 \) and \( 30 \), respectively, the same as default settings in [23]. In this experiment, we change the key length parameter \( u \) within \([1, 8]\) to have \( u \) bits of combined hash keys. The real branching factor \( v \) usually increases when \( u \) becomes larger, and the maximum value can reach \( 2^u \). We set the upper bound of \( u \) as 8, which is sufficient for our evaluation because a 1-byte key can produce 256 branches at the maximum. Similar to [17], a synthetic Twospirals data set is generated for the evaluation. The 2-dimensional data set has 5000 data instances consisting of two normal spiral patterns, 1% uniformly distributed anomalies and 1% clustered anomalies following the standard Gaussian distribution \( \mathcal{N}(0, 1) \). The data set is illustrated in Fig. 2(a). The experimental results are reported in Fig. 2(b)-(i). We plot the contour maps of anomaly scores and the detected boundaries separating the reported anomalies from normal data instances. The real average branching factor, AUC and the number of misclassification errors are also reported in the subtitles with the following format: method (u, v): (AUC, errors).

It can be seen from Fig. 2(b)-(i) that the real average branching factor \( v \) increases when the hash key length \( u \) becomes larger, but is far less than \( 2^u \). Because similar data instances tend to produce the same hash bits with a high probability, the real branching factor seldom increases exponentially with respect to \( u \). Fig. 2(b) shows that the binary case (like in iForest and SciForest) gets a relatively low detection.
performance and has the most irregular boundary. When $v$ increases, in general the boundaries become increasingly tight to dense regions, implying a stronger capability of isolating data instances in sparse regions from those in dense regions. Note that this has two effects on the detection performance. Local and surrounded anomalies become more susceptible to detect when $v$ increases, while anomaly instances in the relatively dense parts of clustered anomalies have a higher risk of being masked. Therefore, the choice of $u$ (or indirectly $v$) is often anomaly- or data-dependent. In terms of Fig. 2, choosing $u$ from 2 to 5 can make a good trade-off in practice.

2) Surrounded Anomalies: We use the Twospiral data set again for this experiment, where most anomalies are well surrounded by the normal spiral patterns. The adjustment factor $\eta$ is set as 1 for all framework instances. The experimental results are shown in Fig. 3 with subtitle format: method: (AUC, errors). The hash key length $u$ in KLSH is set as 3, and the other KLSH parameters are the same as that just used above.

We can see from Fig. 3 that the behaviour of each method varies differently. Both EnKNN and EnLOF have high detection performance, but broad separation boundary caused by large $k$ (further enlarged by the subsampling effect [13]). The boundary of iNNE is more irregular and affected by global anomalies, because it adopts the hypersphere isolation mechanism and only considers the nearest neighbour ($k = 1$). ISO has poor detection performance and fails to detect the surrounded anomalies. It can be seen from Fig. 3(d) that the contour lines formed by ISO are axis-parallel, consistent with previous empirical studies in [8], [17]. SCI can detect the surrounded anomalies to a certain extent due to its optimisation technique and the use of multiple attributes. ALSH performs the worst. The main reason is that the anomalies are not defined based on the angle distance. Note that this does not mean that ALSH always performs the worst, and it can outperform others on certain data sets as shown in the fourth experiment. Most importantly, the other three instances, L1SH, L2SH and KLSH, have very high detection performance and tight boundaries separating most surrounded anomalies. This experiment shows that different instances of the generic framework can have different ability to detect surrounded anomalies.

3) Local Anomalies: We reuse the single-anomaly synthetic data set in [17], as shown in Fig. 4(a). The data set contains only 1 anomaly and 2499 normal data instances. The normal instances consist of two clusters of different densities. Let $C_d$ and $C_s$ denote the dense and sparse clusters, respectively. To simulate a local anomaly, we put the single anomaly instance around $C_d$. Let $r_{C_s}$ denote the maximum nearest neighbour distance in $C_s$, and $r_X$ denote the distance from $X$ to the boundary of $C_s$. We use the ratio $R = r_X/r_{C_s}$ to signify the locality of an anomaly. Roughly, $R < 1$ implies $X$ is a local anomaly. The ratio varies from 0 to 4 in our experiment. Fig. 4(b) reports the results. The parameters for KLSH are set as follows in terms of [23]: $\lambda = \sqrt{2500} = 50$, $\zeta = \frac{1}{4}$, and $\gamma = \frac{1}{m}$ (for local anomalies), where $m = 2$. To study how the adjustment factor $\eta$ affects the performance of our framework, the results with both $\eta = 0$ and $\eta = 1$ are reported. Since the AUC values of ALSH are very low (lower than 0.5), they are excluded from Fig. 4(b) for conciseness.

From Fig. 4(b), it can be observed that only EnLOF, iNNE, SCI and KLSH can detect the anomaly when it is local ($R < 1$). Both EnLOF and iNNE leverage the notion of relative density proposed for local anomaly detection [6], [17]. But note that they have very high computational cost. SCI and KLSH show the capability of our framework in detecting local anomalies. The success of SCI implies instantiations of our framework with learning based hashing have the potential to detect local anomalies. KLSH shows the success of transforming data into kernelised space and applying the isolation mechanism for detecting local anomalies in the original space. Note that both the optimisation process in learning based hashing have the potential to detect local anomalies. KLSH shows the success of transforming data into kernelised space and applying the isolation mechanism for detecting local anomalies in the original space. Note that both the optimisation process in learning based hashing and the computation of kernel functions in KLSH often incur certain computational cost. Although there is no free lunch for our framework to detect local anomalies, the cost can be much less than EnLOF and iNNE. Another observation is that in the original space the framework instances with $\eta = 0$ have a stronger ability to detect local anomalies than those with $\eta = 1$, adhering to our analysis of Algorithm 4 in Section IV-B. Also note that the AUC of all methods except ISO (and ALSH) reaches 1.00 finally with $R$ increasing, also revealing the shortcomings of ISO. Although both ISO and
L1SH are based on the $l_1$ distance, L1SH has a stronger ability in local anomaly detection.

4) **Real-world Benchmark Data Sets:** Besides the synthetic data sets, we evaluate the instances of our framework on a variety of real-world benchmark data sets which have been extensively used in the literature [8], [13], [16]–[18]. All the data sets are available at the UCI Machine Learning Repository.1 Most of the data sets are preprocessed and anomalies are labelled as the same as the previous work in the literature. All attributes are normalised into Z-scores. The data sets are summarised in Table IV. In this experiment, $\eta$ is set as 1 for the framework instances. The KLSH parameters $\lambda$ and $\xi$ are set as follows: $\lambda = \min(\sqrt{n}, 300)$, $\xi = \min(\frac{1}{4}, 30)$.

The AUC of all methods are reported in Table V. For each data set, the leading values are highlighted. The methods with highlighted values can be deemed to have similar performance as their AUC difference is less than 5%. It can be seen from the table that EnKNN and L2SH are the two most robust methods as they perform best on most of the data sets. SCI, L1SH and KSLH also perform quite well and can be considered stable on most data sets. Although SCI can produce the best AUC values on some data sets, it can also perform nearly the worst on certain data sets, e.g., the Power data set. This is caused by its data-dependent optimisation process. In contrast, ISO and all of our framework instances except ALSH have relatively stable performance and can avoid the worst case, attributed to their data-independent hashing feature. ALSH and EnLOF can only perform well on certain data sets and are unstable over all data sets, making their applicability heavily anomaly- or data-dependent. The INNE method seldom performs well. Note that EnKNN fails to perform well on the high-dimensional data set Har, while ISO, SCI, L1SH and L2SH can still work well. This is because the LSH schemes are originally proposed to address the high-dimensional problems in nearest neighbour search. Overall, the instances of our framework can perform robust anomaly detection on a diverse range of data sets, even in high-dimensional space. In particular, L2SH has both advantages of high detection quality and robustness. Although EnKNN has a comparable performance, it becomes less appealing due to its high computational cost as shown in Table VI.

The execution times of all methods are reported in Table VI. The execution time of EnKNN, EnLOF and iNNE are close to each other, because they have the same core step, i.e., finding the nearest neighbour or top-k nearest neighbours. The time complexity of this operation is linear to the sample size. ISO, ALSH, L1SH and L2SH have similar execution time, attributed to their logarithmic time complexity with respect to the sample size. Roughly, these tree isolation based methods can be faster than the aforementioned three methods by 2 orders of magnitude. Both SCI and KLSH have higher execution time than the data-independent instances of our framework. As the additional computation in SCI is engendered by the training stage, it can still perform efficiently when $n$ becomes very large. In contrast, the execution time of KLSH increases considerably with respect to $n$ as the parameter $\lambda$ is set as $\min(\sqrt{n}, 300)$. A user can set a smaller value for $\lambda$ by choosing a better bandwidth parameter $\gamma$ as the detection performance of KLSH is more sensitive to $\gamma$. For high-dimensional data, we can employ the fixed-dimensional feature bagging technique proposed in [13] to enhance the detection efficiency.

In summary, most instances of our framework can achieve high detection performance and time efficiency. In particular, the L2SH instance makes the best trade-off among time efficiency, detection quality and robustness in terms of the experimental results. More importantly, the experiment shows that no instance of our framework can always beat the others due to the variety of anomaly and data types in real-world applications. Therefore, it is a necessity to have the proposed versatile and fast framework for ensemble anomaly analysis.

### VI. Conclusion

The proliferation of big data applications demands highly scalable and fast data analytics techniques. In this paper, we have investigated the recently developed tree isolation mechanism for fast ensemble anomaly analysis in a principled way. We have analysed the existing tree isolation based detection methods and pointed out their limitation that they fail to generalise to other commonly-used distance measures.

<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>r</th>
<th>Outlier or Inlier Labels</th>
<th>Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glass0905</td>
<td>214</td>
<td>9</td>
<td>class 6 vs. others</td>
<td>4.2%</td>
</tr>
<tr>
<td>Iris</td>
<td>150</td>
<td>34</td>
<td>&quot;bad&quot; vs. &quot;good&quot;</td>
<td>35.9%</td>
</tr>
<tr>
<td>Brech</td>
<td>478</td>
<td>9</td>
<td>&quot;sulphate vs. benzoic&quot;</td>
<td>35.5%</td>
</tr>
<tr>
<td>Pima</td>
<td>985</td>
<td>8</td>
<td>&quot;positive&quot; vs. &quot;negative&quot;</td>
<td>34.9%</td>
</tr>
<tr>
<td>WBC</td>
<td>4255</td>
<td>12</td>
<td>speaker 1 (smooth samples) vs. 6, 7 &amp; 8</td>
<td>34.1%</td>
</tr>
<tr>
<td>Power</td>
<td>879</td>
<td>4</td>
<td>the 10 attribute $\geq 1/9$ std</td>
<td>3.7%</td>
</tr>
<tr>
<td>waveform</td>
<td>2505</td>
<td>21</td>
<td>class 0 vs. others</td>
<td>4.62%</td>
</tr>
<tr>
<td>Wine</td>
<td>5178</td>
<td>11</td>
<td>class 3.3 or K vs. others</td>
<td>3.5%</td>
</tr>
<tr>
<td>Sensor</td>
<td>8406</td>
<td>24</td>
<td>Single-category vs. others</td>
<td>63.7%</td>
</tr>
<tr>
<td>Occupancy</td>
<td>6881</td>
<td>5</td>
<td>&quot;1&quot; vs. &quot;0&quot;</td>
<td>6.24%</td>
</tr>
<tr>
<td>Skin</td>
<td>5521</td>
<td>3</td>
<td>&quot;1&quot; vs. &quot;2&quot;</td>
<td>3.67%</td>
</tr>
<tr>
<td>Satellite</td>
<td>6435</td>
<td>36</td>
<td>class 2, 4 &amp; 5 vs. others</td>
<td>31.6%</td>
</tr>
<tr>
<td>Shmidt</td>
<td>4999</td>
<td>9</td>
<td>class 1, 5, 6 &amp; K vs. class 1</td>
<td>7.15%</td>
</tr>
<tr>
<td>Cementype</td>
<td>26838</td>
<td>10</td>
<td>class 4 vs. class 3</td>
<td>30.6%</td>
</tr>
<tr>
<td>MNIST138</td>
<td>30524</td>
<td>784</td>
<td>others vs. &quot;spatial&quot;, &quot;artificial&quot; &amp; normal</td>
<td>1.79%</td>
</tr>
<tr>
<td>Anholtiyuma</td>
<td>452</td>
<td>274</td>
<td>class 3, 4, 5, 7, 8, 9, 11 &amp; 15 or others</td>
<td>14.0%</td>
</tr>
<tr>
<td>Micr</td>
<td>666</td>
<td>689</td>
<td>class 8 &amp; 9 (10% samples) vs. others</td>
<td>9.11%</td>
</tr>
<tr>
<td>Har</td>
<td>5135</td>
<td>561</td>
<td>class 4, 5 &amp; 6 (10% samples) vs. others</td>
<td>10.1%</td>
</tr>
<tr>
<td>Isot</td>
<td>6080</td>
<td>617</td>
<td>class 9, 13 &amp; 18 (25% samples) vs. others</td>
<td>4.25%</td>
</tr>
</tbody>
</table>

1https://archive.ics.uci.edu/ml/datasets.html

<table>
<thead>
<tr>
<th>Data Set</th>
<th>AUC (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnKNN</td>
<td>77.45</td>
</tr>
<tr>
<td>EnLOF</td>
<td>82.45</td>
</tr>
<tr>
<td>NNN</td>
<td>93.52</td>
</tr>
<tr>
<td>ISO</td>
<td>96.59</td>
</tr>
<tr>
<td>SCI</td>
<td>97.88</td>
</tr>
<tr>
<td>ALSH</td>
<td>98.00</td>
</tr>
<tr>
<td>L1SH</td>
<td>98.35</td>
</tr>
<tr>
<td>L2SH</td>
<td>98.55</td>
</tr>
<tr>
<td>KLSH</td>
<td>98.70</td>
</tr>
</tbody>
</table>

### VI. Conclusion

The proliferation of big data applications demands highly scalable and fast data analytics techniques. In this paper, we have investigated the recently developed tree isolation mechanism for fast ensemble anomaly analysis in a principled way. We have analysed the existing tree isolation based detection methods and pointed out their limitation that they fail to generalise to other commonly-used distance measures.
as the underlying distance measures in these methods have not been well understood. We have proposed a generic framework named LSHiForest for fast tree isolation based ensemble anomaly analysis with the use of a Locality-Sensitive Hashing (LSH) forest. In this way, the fast tree isolation mechanism can apply to any distance or similarity measure, data space or type where an LSH family is defined. The existing methods iForest and SCiForest have been formally shown to be two special cases of our framework. We have further instantiated the framework with three commonly-used LSH families, i.e., $\ell_p$ ($p = 1, 2$) LSH, angle-based LSH and kernelised LSH. Extensive experiments have been conducted on both synthetic and real-world benchmark data sets to evaluate the instances of the framework and compare with the existing ensemble anomaly detection methods. The experimental results have validated the effectiveness and efficiency of the proposed framework. In particular, the framework instance based on the $\ell_2$ LSH family has the best overall performance on the benchmark data sets in terms of time efficiency, detection quality and robustness. In future work, we intend to apply LSHiForest instantiated with certain LSH families in real-time or low-latency anomaly detection applications given its high time efficiency and detection performance.

ACKNOWLEDGMENT

This paper is partially supported by the Open Project of State Key Laboratory for Novel Software Technology KFKT2015A03, the National Science Foundation of China under the grant No. 61672276, and the Australian Research Council Discovery Projects DP170104747 and DP170101932.

REFERENCES


TABLE VI: Execution time of all methods (s).

<table>
<thead>
<tr>
<th>Method</th>
<th>L-NN</th>
<th>L-LOF</th>
<th>NSNE</th>
<th>ISIO</th>
<th>SCi</th>
<th>ROC</th>
<th>4LSH</th>
<th>2LSH</th>
<th>KMLSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>iForest</td>
<td>1.4</td>
<td>4.1</td>
<td>1.8</td>
<td>0.9</td>
<td>2.9</td>
<td>1.4</td>
<td>9.8</td>
<td>1.5</td>
<td>1.2</td>
</tr>
</tbody>
</table>