Random Histogram Forest for Unsupervised Anomaly Detection

Andrian PUTINA, Mauro SOZIO
Telecom Paris, Paris, France
name.surname@telecom-paris.fr

Dario ROSSI, José M. NAVARRO
Huawei Technologies SASU, Paris, France
name.surname@huawei.com

***Abstract*—Roughly speaking, anomaly detection consists of identifying instances whose features significantly deviate from the rest of input data. It is one of the most widely studied problems in unsupervised machine learning, boasting applications in network intrusion detection, healthcare and many others. Several methods have been developed in recent years, however, a satisfactory solution is still missing to the best of our knowledge. We present *Random Histogram Forest* an effective approach for unsupervised anomaly detection. Our approach is probabilistic, which has been proved to be effective in identifying anomalies. Moreover, it employs the fourth central moment (aka *kurtosis*), so as to identify potential anomalous instances. We conduct an extensive experimental evaluation on 38 datasets including all benchmarks for anomaly detection, as well as the most successful algorithms for unsupervised anomaly detection, to the best of our knowledge. We evaluate all the approaches in terms of the average precision of the area under the precision-recall curve (AP). Our evaluation shows that our approach significantly outperforms all other approaches in terms of AP while boasting linear running time.**

I. Introduction

Hawkins [1] defines an *anomaly* (aka *outlier*) as "*an observation, which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism*." Identifying those anomalies is crucial in many applications, as it might reveal an unauthorized access in computer networks or a disease in a patient, for example.

Thus, *anomaly detection* boasts applications in network intrusion detection, fraud detection, healthcare, and many others, while providing a valuable tool in data cleaning.

It is one of the most widely studied problems in both unsupervised and supervised machine learning. Because of the difficulties in obtaining large amounts of labeled data, as well as, in dealing with high class imbalance [2], unsupervised approaches preserve their appeal. As a result, unsupervised anomaly detection has received increasing attention in recent years (e.g. *Isolation Forest* [3], *OCSVM* [4], *LOF* [5]). Among the supervised approaches we mention random forest [6] and autoencoders [7].

One of the most effective algorithms for unsupervised anomaly detection is Isolation Forest (*iForest*), as confirmed also by our experimental evaluation. In our work, we present *Random Histogram Forest* (RHF) an effective approach for unsupervised anomaly detection. Similarly to *iForest*, our approach is probabilistic while relying on an "ensemble" of "weak" building blocks (trees) for effectively identifying anomalies. This has been proved to be effective for a wide range of tasks (e.g. Random Forest [6], Isolation Forest [3]).

Our approach builds a random forest based on all input instances, whereas *iForest* builds a random forest based solely on some random samples of the data. The latter strategy has the drawback that some anomalies might be neglected entirely in the construction of the forest, thereby impairing the capability of the algorithm of finding those anomalies. Nevertheless, our algorithm boasts linear running time in the size of the input. Another key idea of our algorithm is to employ the fourth central moment (aka *kurtosis*), so as to guide the search for anomalous instances. Our algorithm computes a score for every instance, measuring the likelihood of such an instance of being an anomaly. Such a score is defined as the *Shannon's information content* of the leaf containing the corresponding instance.

We conduct an extensive experimental evaluation on 38 datasets including all benchmarks for anomaly detection, as well as the most successful algorithms for unsupervised anomaly detection, to the best of our knowledge. We evaluate all the approaches in terms of the average precision of the area under the precision-recall curve (AP). We observe that as suggested in [8], ROC might not reflect the real performances of the algorithm, in that, anomalies typically represent a small fraction of the input data.

Our experimental evaluation shows that RHF outperforms all other approaches in terms of AP. In particular, it outperforms *iForest* in terms of AP by a factor of 10% on average and up to a factor of 2 in some datasets. Moreover, it shows that the performances of our algorithm are consistently good over a wide range of parameter values, which allows for an effective parameter tuning.

The rest of the paper is organized as follows. We start by overviewing top performer anomaly detection algorithms in section II and introduce the Random Histogram Forest algorithm in section III. We then describe our experimental evaluation and results in section IV. Finally, we discuss and summarize our main findings in section V.

II. Related Work

Several unsupervised anomaly detection algorithms have been developed in recent years. We can classify the related work as follows: (i) *Probabilistic/Linear* based methods (e.g.

PPCA, *HBOS*, *OCSVM*, etc.); (ii) *Proximity/Nearest-Neighbor* based methods (e.g. *K-NN*, K^{th}*-NN*, *Local Outlier Factor*); (iii) *Ensemble/Isolation* based methods (e.g. *Isolation Forest*). Among the most recent comparative studies of unsupervised techniques, [9] compares most of the existing proximity-based methods on 10 different datasets and conclude that it is of great importance the initial assumption whether the anomalies in the datasets are global or local: they recommend to use a *global* anomaly detection methods if there is no further knowledge about the nature of anomalies in the dataset to be analyzed. [10] compares 14 methods belonging to all the groups previously described on 15 different datasets (12 publicly available and 3 private ones). However, their study assesses if the models are able to generalize to future instances, so they perform a Monte Carlo cross validation of 5 iterations, using 80$\%$ of the data for the training phase and 20$\%$ for the prediction which indicates a semi-supervised setup to our understanding. While [9] study does not include the latest methods presented (e.g. Isolation Forest, thought to be the state-of-art), [10] describes the models generalization capacity using labels that most of the time are not available. We compare the methods which have proven to be the best in previous studies [9], [10] using default or reasonable parameters and use labels only to assess their performance in a completely *unsupervised* environment.

Probabilistic based methods estimate the parameters θ of the dataset X and assigns to each instance $x \in X$ an anomaly score equal to the likelihood $P(X \mid \theta)$.

Probabilistic Principal Component Analysis (PPCA) [11]: estimates the principal components of the data and projects the d-dimensional dataset to a q-dimensional one estimating the latent variables by iteratively maximizing the likelihood using an expectation-maximimation algorithm.

Histogram-based Outlier Score (HBOS) [12]: generates a histogram for each feature assuming they are independent. Similar to the Naive Bayes approach in which all the independent feature probabilities are multiplied, HBOS outputs an anomaly score given by the multiplication of the inverse height of the bins of all the features.

One-Class SVM [4]: determines a separating hyperplane in a higher dimensional space by maximizing the distance from the hyperplane to the origin. The ν parameter acts as a regularization parameter as determines an upper bound on the percentage of data falling outside the boundary and a lower bound on the number of support vectors.

Proximity/Nearest-Neighbor based methods compute the neighborhood of all the instances $x \in X$ and uses the distances of each instance x to describe abnormality.

K-NN [13] and *Kth-NN [14]*: Both *K-NN* and K^{th}*-NN* compute the distances for each instance $x \in X$ to the k-nearest-neighbors and assign them either the mean distance (*K-NN*) or the max one (k^{th}*-NN*).

Local Outlier Factor (LOF) [5]: introduced first the idea of local anomalies and detects them by comparing the local density of each instance against the local density of neighbors.

Ensemble/Isolation based methods isolate anomalies instead of profiling normal instances by recursively splitting the data through a random tree and generating so isolation forests.

Isolation Forest [3]: builds a forest of randomly generated trees and assigns to each instance $x \in X$ the average path length from the root to the node containing x as anomaly score. The authors show empirically that shorter path lengths are representative of anomalies as they are more easily to isolate with respect to normal data.

Based on the most recent comparison studies [9], [10], [15], the algorithms previously described have proven to be among the best in regards *anomaly detection*. In general both [10] and [15] suggest *iForest* to be, on average, the best one closely followed [10] by *PPCA* and *OCSVM*.

III. Random Histogram Forest (RHF)

RHF is an ensemble model apt at splitting the dataset into η different groups. The algorithm randomly partitions the input points by means of several splits drawn at random. Intuitively, points that end up in a relatively large group are less likely to be anomalies. By iterating such a process multiple times, we can measure how likely a point is an anomaly.

To put this idea into practice, *RHF* splits the data into η different bins by means of a Random Histogram Tree (*RHT*). Each *RHT* is built by recursively splitting the whole dataset: each splitting decision is done selecting an attribute a to split according to its kurtosis score and a split value randomly selected from its support. By setting the parameter max height h, each RHT produces at most $\eta = 2^h$ leafs corresponding to η bins in which all the instances are grouped. Finally, the anomaly score of each instance $x \in X$ is the *information-content* (aka *self-entropy*) of its terminating *leaf Q* aggregated over all the trees.

Random Histogram Tree: Given a dataset $X \in \mathbb{R}^{n \times d}$, where n is the number of instances and d the number of *features* or *attributes*, a *RHT* is a binary tree in which a node Q_i is either an *internal* node with exactly two children or a *leaf* node with no children. In the former case, the node splits the dataset into a left branch X_{QL} and a right one X_{QR}, according to the *kurtosis Split*. A *RHT* contains at most $\eta = 2^h$ leafs, where h is the maximum height of the tree.

kurtosis Split: a kurtosis split chooses according to which attribute $a \in d$ to split the dataset X by assigning higher probability to features whose kurtosis scores are higher. Given a random variable $\mathcal{X}$, the kurtosis score is defined as:

$$K[\mathcal{X}] = E\left[\left(\frac{\mathcal{X}-\mu}{\sigma}\right)^4\right] = \frac{E\left[(\mathcal{X}-\mu)^4\right]}{E\left[(\mathcal{X}-\mu)^2\right]^2} = \frac{\mu_4}{\sigma^4}, \quad (1)$$

where μ_4 is the fourth central moment and σ the standard deviation, measures the *tailedness* of $\mathcal{X}$. Equation (1) denotes the standardized data raised to the fourth power. As a result, instances within the region of the peak have negligible contribution to the kurtosis score, while instances outside the region of the peak (e.g. *outliers*) contribute the most. In [16], Moors defined it as a measure of dispersion, while he concluded that high values of K are due to either i) occasional values far from the mean in a distribution whose probability mass is concentrated around the mean or ii) probability mass concentrated in the tails of the distribution. The kurtosis score measures the heaviness of the tails and it is therefore an indicator of *outliers*' existence in the tail. Consider, for example, Fig. 1 representing 4 features extracted from datasets *Annthyroid* (top) and *Mulcross* (bottom) depicting both *normal* and *anomalous* probability density functions. It is easily observable that features with heavier tails and consequently higher kurtosis score (e.g. X_1-top and X_2/X_3-bottom) are more likely to contain anomalous instances than the remaining ones (e.g. X_0/X_4-top and X_0/X_1-bottom in which anomalies are clearly not separable: the probability functions overlap).

We compute the logarithm $y = log(x + 1)$ of the kurtosis score so as to focus on its order of magnitude, while preventing our approach from being sensitive to small changes on such score. Notice that we add 1 to all the scores such that constant features (kurtosis 0) obtain a transformed score equal to 0 and thus not selectable as a splitting attribute.

A *kurtosis Split* therefore:

i computes the kurtosis scores (1) of all the attributes $a \in d$ and defines the *kurtosis Sum* K_s:

$$K_s = \sum_{a=0}^{d} log\left[K(X_a) + 1\right] \tag{2}$$

ii picks a random value from $\mathcal{X} \sim U[0, K_s]$:

$$r = \mathcal{X} \sim U[0, K_s] \tag{3}$$

iii designates the attribute to split a_s according to the choice r in K_s:

$$a_s = argmin\left(i \middle| \sum_{a=0}^{i} log\left[K(X_i) + 1\right] > r\right) \tag{4}$$

Internal Node: Given an internal node Q, the latter splits the data in exactly two subsets X_{QL} and X_{QR} (if possible) by selecting an attribute $a_s \in d$ according to the kurtosis Split criterion and a splitting value $a_v \in (min_{a_s}, max_{a_s})$ until:

i the maximum height h of the tree is reached.

ii there are $|X_Q| = 1$ instances in Q.

iii there is only one distinct instance in Q. All the remaining instances are thus duplicates.

Leaf: A leaf is a node with no children which is populated by at least one instance. Given a leaf Q, we denote with $S(Q)$ the set of distinct instances associated to Q. We also define $P_Q := \frac{|S(Q)|}{n}$, which can be seen as the probability that an instance is associated to the leaf Q. We recall that the total number of leaves is bounded by 2^h, where h is the maximum height of the tree (input parameter).

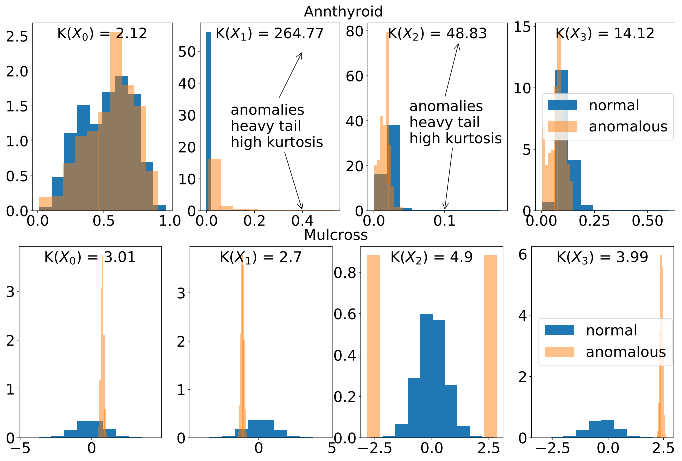

Fig. 1. Probability Density Function of 4 features depicting both *normal* and *anomalous* class extracted from datasets *Annthyroid* and *Mulcross* respectively. It is easily observable that features with heavier tails and consequently higher *kurtosis* score (e.g. X_1-top and X_2/X_3-bottom) are more likely to contain separable *anomalous* instances than remaining ones (e.g. X_0/X_4-top and X_0/X_1-bottom in which *anomalies* are clearly not separable).

Anomaly Score: Given an instance $p \in X$ and a *Random Histogram Tree* $RHT_i \in RHF$, we define the anomaly score of p w.r.t. RHT_i as:

$$RHT_i(p) = \log\left[\frac{1}{P_{Q_j}}\right], \quad p \in S(Q_j), \tag{5}$$

which can be seen as the *Information Content* (also called Shannon's information [17]) of p in RHT_i. The *Information Content* measures the level of "surprise" of an event, with rare events being more surprising than relatively common events. As a result, the smaller the cardinality of $S(Q_j)$ the more likely p is considered to be an outlier. We adopt the convention that $-log(0) = +\infty$. The anomaly score of p w.r.t. to RHF is obtained by summing the scores over all RHT_i's as follows.

$$RHF(p) = \sum_{i=0}^{t} RHT_i(p) \tag{6}$$

The pseudocode of the algorithm to construct a random histogram forest is provided in Algorithm 1, while Algorithm 2 shows how to compute the anomaly score for a point p.

The running time of Algorithm 1 is $\mathcal{O}(ntdh)$, where h denotes the maximum height, t denotes the number of trees, n the number of instances while d denotes the number of dimensions d.

IV. Experimental Evaluation

A. Settings.

Libraries: Our experimental evaluation is conducted on a Linux Fedora 31 server equipped with Intel(R) Xeon(R) CPU E5-2665 @ 2.40GHz - 32 CPUs and 48 GB

Algorithm 1: RHF(X, nd, h)

```
    Input: dataset X, node depth nd and max height h
    Output: RHF
 1: if nd $\geq$ h or $|X|$ == 1 then
 2:     return Leaf{S(Q)}
 3: else
 4:     let A be the set of attributes
 5:     select the attribute to split $a_s$ according to kurtosis
        Split described in (2), (3), (4)
 6:     select a random split value $a_v$ in the min and max
        support of $a_s$ in X
 7:     $X_l$ = filter($X|\ a_s < a_v$)
 8:     $X_r$ = filter($X|\ a_s \geq a_v$)
 9:     return Node {
            value = $a_v$,
            attribute = $a_s$,
            left = RHT($X_l$, nd+1, h),
            right = RHT($X_r$, nd+1, h)
        }
10: end if
```

Algorithm 2: Score(x, node)

```
    Input: instance $x \in X$ and an RHT node
    Output: anomaly score given RHT
 1: if node is Leaf then
 2:     $P = node\{S(Q)\}/|X|$
 3:     return $log(1/P)$
 4: else
 5:     a = node.attribute
 6:     v = node.value
 7:     if $x_a < v$ then
 8:         score(x, node.left)
 9:     else
10:         score(x, node.right)
11:     end if
12: end if
```

RAM. Our code is written in *Python 3* [18] while it uses $NumPy == 1.17.4$ *[19] and* $Pandas == 0.25.3$ *[20]* for data preprocessing. The implementations of the algorithms described in Section II belong to either $PyOD == 0.7.9$ [21] (HBOS, PPCA, K-NN, K^{th}-NN, OCSVM) or $Scikit-learn == 0.23.1$ [22] (iForest and LOF) packages. Our *RHF*'s *Python 3* implementation is available at [23].

Parameters: We set the parameters of each approach, considered in our experimental evaluation, while following the directions of the corresponding authors. In particular, we run *HBOS* selecting the input parameter *number of bins* using the rule of thumb $K = \sqrt{n}$ as suggested by the authors. Similarly, we set $n\_components = mle$ and $svd\_solver = full$ which finds the best number of PPCA's components. We run 10 times the proximity based methods *K-NN*, *K^{th}-NN* and *LOF* by increasing the number of nearest-neighbors $K - nn \in [20, 30]$ and aggregate the scores. As already successfully done in [10], we use *OCSVM*'s default parameters *kernel=rbf*, *degree=3* with regularization parameter $\nu = 0.5$. Finally, *iForest* uses its default parameters $t = 100$ and sample size $\psi = 256$. RHF uses $t = 100$ and $h = 5$ corresponding to at most 32 *leafs*. Both *Isolation Forest* and *RHF* are run 10 times.

Metrics: We evaluate the performance of the algorithms by measuring the *Average Precision (AP)* of the *Precision-Recall Area Under the Curve* (without interpolation): $AP := \sum_n (R_n - R_{n-1}) P_n$ where $P_n = \frac{tp}{tp+fp}$ and $R_n = \frac{tp}{tp+fn}$ are the Precision and Recall at the n^{th} threshold. We observe that the Receiver Operating Characteristic (ROC) is often employed to evaluate anomaly detection methods [9] - [3]. However, it has been shown in [8] that when the classes are not balanced (e.g. anomalies) the *AP* curves better reflect the efficacy of an algorithm. [8] shows moreover that a curve dominates in ROC if and only if it dominates in AP space.

Datasets: We put a major effort in providing an extensive experimental evaluation. In particular, we include all datasets that have been considered in the literature for anomaly detection, to the best of our knowledge. This is crucial to ensure a fair comparison, in that, the overall results might change dramatically depending on the selection of the datasets, as pointed out in [15]. We use 38 publicly available benchmark datasets ranging from 240 to 623091 instances and from 3 to 274 features. Each of them is available either at the UCI [24] or at the ODDS [25] repositories. We furthermore consider the recently released *WikiQOE* [26] dataset, which consists of a Wikipedia large measurements campaign of WebQOE metrics.

The KDD'99 Cup dataset is one of the most widely used benchmark for anomaly detection. The dataset contains information about network connections as exchanged bytes ("source bytes", "destination bytes", etc.) and service type ("http", "smtp", "ftp", etc.). It consists of 4,898,431 instances and 41 attributes. Similarly to the filtering technique used by [27] [9] we extract 5 subsets according to the values of the *service* attribute (*http*, *smtp*, *ftp*, *finger* and *other*). Out of the 41 available attributes, we select, as already done in [27] only 3 of them namely "duration", "source bytes", and "destination bytes" as they are thought to be the most relevant ones [27]. We obtain in this way the datasets we call *kdd_http* (623091 instances), *kdd_smtp* (95554 instances), *kdd_ftp* (5214 instances), *kdd_finger* (1033 instances) and *kdd_other* (12844 instances). While [9] filtered the dataset according to the *service* attribute only, [27] filters them also by the positive *logged_in* attribute as they are successful attacks. We also consider this additional filter by further reducing the *kdd_http* dataset into the *http_logged* (567498 instances) one by excluding the negative values of *logged_in* attribute.

In order to determine to which extent the presence of duplicates might affect the overall results, we consider also a smaller version in which duplicates have been filtered out:

	n	d	anomalous - duplicates	HBOS	PPCA	OCSVM	KNN	KthNN	LOF	ISO	RHF_K	RHF_R
musk	3060	166	3.1% - 0%	0.904	**1.0**	**1.0**	0.432 ± 0.023	0.626 ± 0.027	0.239 ± 0.013	0.980 ± 0.021	**0.994 ± 0.007**	0.990 ± 0.008
http_logged	567498	3	0.4% - 97%	0.242	0.769	0.492	0.009 ± 0.001	0.009 ± 0.001	0.022 ± 0.003	0.947 ± 0.033	**0.982 ± 0.002**	0.990 ± 0.001
kdd_smtp29	96554	3	0.03% - 0%	0.980	0.773	0.405	0.090 ± 0.002	0.104 ± 0.002	0.014 ± 0.001	**0.989 ± 0.001**	0.954 ± 0.008	0.970 ± 0.005
breastcancer	683	9	34.9% - 1.2%	0.878	0.958	0.918	0.933 ± 0.001	0.942 ± 0.002	0.294 ± 0.007	**0.967 ± 0.004**	0.952 ± 0.005	0.962 ± 0.003
shuttle	49097	9	7% - 0%	0.911	0.915	0.907	0.182 ± 0.001	0.188 ± 0.001	0.118 ± 0.002	**0.976 ± 0.005**	0.933 ± 0.006	0.951 ± 0.003
satimages	5803	36	1.2% - 0.03%	0.732	0.872	0.965	0.443 ± 0.013	0.554 ± 0.033	0.028 ± 0.002	0.923 ± 0.006	**0.926 ± 0.008**	0.918 ± 0.010
kdd_ftp	5214	3	26.7% - 79.7%	0.391	0.846	0.596	0.259 ± 0.001	0.261 ± 0.001	0.284 ± 0.002	0.428 ± 0.014	**0.922 ± 0.008**	0.909 ± 0.007
ionosphere	351	33	35.8% - 0.7%	0.28	0.747	0.839	**0.922 ± 0.003**	0.866 ± 0.009	0.851 ± 0.004	0.812 ± 0.005	**0.819 ± 0.006**	0.800 ± 0.006
kdd99G	620098	29	0.17% - 1.33%	0.585	0.683	0.325	0.177 ± 0.003	0.190 ± 0.003	0.004 ± 0.001	0.531 ± 0.002	**0.774 ± 0.056**	0.577 ± 0.033
kdd_http29	623091	29	0.64% - 31.2%	0.536	0.758	0.499	0.096 ± 0.002	0.108 ± 0.001	0.017 ± 0.003	0.537 ± 0.012	**0.770 ± 0.076**	0.501 ± 0.015
kdd_http_distinct	222027	3	0.03% - 0%	0.049	0.637	0.373	0.352 ± 0.010	0.375 ± 0.007	0.027 ± 0.001	0.017 ± 0.005	**0.743 ± 0.042**	0.795 ± 0.015
mulcross	262144	4	10% - 0%	0.064	**0.979**	0.643	0.052 ± 0.001	0.052 ± 0.001	0.171 ± 0.001	0.565 ± 0.034	**0.733 ± 0.032**	0.730 ± 0.041
satellite	5100	36	1.4% - 0%	0.500	0.583	0.622	0.563 ± 0.012	0.612 ± 0.006	0.187 ± 0.001	0.639 ± 0.014	**0.651 ± 0.015**	0.650 ± 0.015
magicgamma	19020	10	35.1% - 1.7%	0.467	0.586	0.626	**0.735 ± 0.001**	0.728 ± 0.001	0.540 ± 0.003	**0.648 ± 0.008**	0.624 ± 0.010	0.626 ± 0.012
wbc	378	10	5.5% - 4.7%	**0.699**	0.556	0.529	0.546 ± 0.001	0.554 ± 0.002	0.573 ± 0.011	**0.591 ± 0.026**	0.577 ± 0.013	0.612 ± 0.011
cardio	1831	31	9.6% - 0.56%	0.416	**0.612**	0.533	0.363 ± 0.003	0.384 ± 0.002	0.156 ± 0.001	0.557 ± 0.027	**0.567 ± 0.023**	0.553 ± 0.023
penglobal	809	16	11.1% - 0%	0.237	0.301	0.569	**0.897 ± 0.003**	0.864 ± 0.044	0.566 ± 0.018	**0.612 ± 0.027**	0.556 ± 0.039	0.553 ± 0.043
kdd_http	623091	3	0.64% - 98.1%	0.204	**0.550**	0.369	0.010 ± 0.001	0.010 ± 0.001	0.017 ± 0.002	0.488 ± 0.049	**0.550 ± 0.009**	0.572 ± 0.002
thyroid	3772	6	2.4% - 0%	**0.718**	0.362	0.318	0.344 ± 0.006	0.376 ± 0.005	0.088 ± 0.005	0.515 ± 0.031	**0.550 ± 0.024**	0.376 ± 0.026
kdd_other	12844	3	3.72% - 0%	0.076	0.052	0.092	0.095 ± 0.001	0.089 ± 0.001	0.095 ± 0.002	0.493 ± 0.012	**0.539 ± 0.062**	0.477 ± 0.046
pima	768	8	34.8% - 0%	0.517	0.454	0.464	**0.521 ± 0.001**	**0.521 ± 0.001**	0.423 ± 0.002	**0.505 ± 0.012**	0.489 ± 0.008	0.496 ± 0.008
arrhytmia	452	274	14.6% - 0%	0.384	0.395	0.391	0.390 ± 0.001	0.395 ± 0.001	0.340 ± 0.004	**0.478 ± 0.012**	0.431 ± 0.019	0.460 ± 0.021
spambase	4601	57	39.4% - 7.4%	**0.541**	0.404	0.399	0.408 ± 0.001	0.422 ± 0.001	0.345 ± 0.004	**0.465 ± 0.017**	0.413 ± 0.014	0.410 ± 0.023
kdd_ftp_distinct	2876	3	9.8% - 0%	0.304	0.259	0.190	0.255 ± 0.002	0.264 ± 0.002	0.138 ± 0.010	0.391 ± 0.011	**0.399 ± 0.025**	0.347 ± 0.017
kdd_smtp	96554	3	1.22% - 97.8%	0.286	0.353	0.323	0.015 ± 0.001	0.015 ± 0.001	0.020 ± 0.001	0.251 ± 0.013	**0.389 ± 0.087**	0.478 ± 0.122
abalone	1920	7	1.5% - 0%	0.265	0.298	0.308	0.189 ± 0.015	0.329 ± 0.004	0.194 ± 0.022	**0.458 ± 0.058**	0.377 ± 0.027	0.340 ± 0.035
mnist	7603	100	9.2% - 0%	0.095	0.383	0.385	0.405 ± 0.001	**0.411 ± 0.001**	0.266 ± 0.007	0.272 ± 0.018	**0.348 ± 0.028**	0.300 ± 0.029
annthyroid	7200	6	2.3% - 0%	**0.411**	0.190	0.186	0.229 ± 0.001	0.225 ± 0.001	0.175 ± 0.002	0.303 ± 0.014	**0.309 ± 0.008**	0.220 ± 0.017
kdd_finger	1033	3	2.42% - 0%	0.266	0.191	0.299	0.287 ± 0.006	**0.312 ± 0.001**	0.163 ± 0.005	0.244 ± 0.022	**0.269 ± 0.019**	0.374 ± 0.020
yeast	1191	31	4.6% - 0%	0.116	0.241	0.202	0.221 ± 0.005	**0.278 ± 0.001**	0.242 ± 0.005	0.220 ± 0.013	**0.229 ± 0.012**	0.214 ± 0.006
mammography	11183	6	2.3% - 2.3%	0.086	**0.205**	0.183	0.170 ± 0.001	0.174 ± 0.001	0.098 ± 0.003	**0.187** ± 0.025	0.134 ± 0.011	0.129 ± 0.023
vowels	1456	12	3.4% - 8%	0.118	0.068	0.195	**0.548 ± 0.007**	0.464 ± 0.005	0.346 ± 0.010	**0.152 ± 0.029**	0.131 ± 0.035	0.103 ± 0.021
wikiqoe	55932	17	8.0% - 0%	0.088	0.120	0.122	0.135 ± 0.001	0.134 ± 0.001	0.091 ± 0.001	**0.168 ± 0.001**	0.118 ± 0.011	0.122 ± 0.010
vertebral	240	6	12.5% - 0%	0.090	0.104	0.103	0.091 ± 0.001	0.089 ± 0.001	**0.105 ± 0.002**	**0.095 ± 0.002**	0.094 ± 0.003	0.095 ± 0.004
kdd_smtp_distinct	71257	3	0.03% - 0%	0.162	0.085	0.058	0.136 ± 0.006	**0.171 ± 0.013**	0.047 ± 0.001	0.049 ± 0.003	**0.078 ± 0.005**	0.074 ± 0.004
cover	286048	10	0.9% - 0%	0.026	0.078	**0.096**	0.041 ± 0.001	0.041 ± 0.001	0.013 ± 0.001	0.059 ± 0.018	**0.077 ± 0.017**	0.085 ± 0.022
wine	4898	11	0.4% - 0%	0.030	0.062	0.065	**0.087 ± 0.001**	0.079 ± 0.001	0.080 ± 0.001	0.041 ± 0.009	**0.064 ± 0.006**	0.076 ± 0.006
aloi	50000	27	3.0% - 0%	0.029	0.037	0.040	0.058 ± 0.001	0.050 ± 0.001	**0.092 ± 0.002**	0.033 ± 0.001	**0.036 ± 0.001**	0.038 ± 0.001
average	//	//	//	0.360 ± 0.093	0.460 ± 0.100	0.411 ± 0.088	0.308 ± 0.084	0.323 ± 0.084	0.197 ± 0.063	0.463 ± 0.098	**0.513 ± 0.100**	0.497 ± 0.100

TABLE I

AP SCORES OF ALL APPROACHES ON ALL OUR DATASETS. THE RESULTS ARE SORTED IN DECREASING ORDER OF *iForest* SCORES. IN THE CASE OF PROBABILISTIC APPROACHES, EACH VALUE IS AN AVERAGE OVER 10 RUNS WHICH IS COMPLEMENTED WITH A 0.95 CONFIDENCE INTERVAL. THE BEST RESULTS FOR EACH DATASET ARE REPRESENTED IN BOLD.

we will refer to them as *kdd_http_distinct*, *kdd_smtp_distinct* and *kdd_ftp_distinct*. We include in our comparison also the full version *kdd_http29* and *kdd_smtp29* in which all the 29 continuous attributes are used. All the continuous features are used also by [9] in which the authors tackle also the duplicates problem by limiting the number of attacks and present to the community their kdd99 dataset (composed by 620098 instances with 0.17% *anomalous* instances). We will refer to this dataset as *kdd99G* by author's name.

B. Comparison

We evaluate all the algorithms discussed in Section II, which have proven to be most effective according to previous experimental evaluations [9], [10], [15]. We also consider two variants of *RHF*: one variant where *Kurtosis Split* is used (RHF_K), and one where random splits are used (RHF_R).

We report in Table I a full comparison of all approaches and all datasets considered in our paper. Our results confirm some of the results provided in [10]. In particular, *iForest* (0.463 ± 0.098), *PPCA* (0.460 ± 0.010), *OCSVM* (0.411 ± 0.088) and *RHF* (0.513 ± 0.100) are indeed the most effective algorithms for *anomaly detection*, while *proximity-based* methods such as *K-NN* and *LOF* appear to be less effective. Moreover, we can see in Table I that both variants of *RHF* outperform the other approaches. RHF_K achieves best results in terms of AP, with the most significant differences being observed in the *mulcross* dataset and those extracted from *kdd99*. We observe that RHF_K achieves an AP which is roughly 10% larger than *iForest* and *PPCA* on average and up to a factor of 2 in some datasets.

One of the main reasons why *RHF* outperforms *iForest* is probably that it computes a forest based on all input instances, while *iForest* only focuses on a random sample of the input data. As a result, there is a non-negligible probability of neglecting some of the anomalies.

RHF and *iForest* clearly outperform PPCA on kdd_http, penglobal, thyroid, kdd_ftp_distinct, while PPCA is the clear winner on mulcross dataset. This is not surprising, considering that the latter dataset consists of two dense anomalous clusters, whose density functions can be accurately estimated by PPCA. For such a dataset, *iForest* exhibits the worst results.

As also stated in [15], we observe that, most of the approaches achieve similar performances in most of the datasets, with the most prominent differences being found in a few

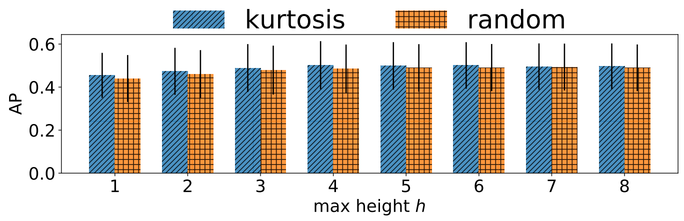

Fig. 2. Parameters tuning: Average Precision score for increasing maximum tree height *h* comparing both *Kurtosis Split* as well as *Random Split* criterion.

datasets such as *http_logged, penglobal, thyroid, mulcross, kdd99G, kdd_other, kdd_fttp, kdd_http_distinct*.

C. Parameters Tuning

RHF uses two input parameters: the max height *h* which determines the η number of *leafs* and the number of trees *t*. As in most of the ensemble methods, we use *t = 100* trees while empirically study the behavior of *h*. Our method is somehow linked to histograms, as we split the data into η leaves. Therefore, we employ the widely used rules of thumb in determining the number of leaves (i.e. bins in histograms). In particular Sturge's [28] rule of thumb $k = \lceil 1 + log_2 n \rceil$ suggests that the number of bins should increase logarithmically in the number of instances. We study *RHF*'s performance for increasing $h \in [1, 8]$ which defines $\eta \in [2, 256]$ comparing the results obtained using both *Random Split* and *Kurtosis Split*. The results depicted in Fig. 2 show two takeaways: i) *RHF*'s performance smoothly vary over *h* and ii) on average, *Kurtosis Split* consistently outperforms the *Random Split*.

Regarding the maximum height *h* parameter, we observe from Fig. 2 that: i) the AP benefits from increasing *h*, however, ii) the AP reaches its maximum value already when *h = 4/5*; iii) the number of leaves $\eta = 16/32$ defined by *h=4/5* is consistent with Sturge's rule of thumb which would recommend to use $K = 14$ bins for smaller datasets (e.g. 5000 instance) while $K = 21$ for bigger ones (e.g. 620000 instance). We set and recommend thus *h = 5*, as it produces the best results and should handle properly also larger datasets.

V. Conclusions

We present a novel anomaly detection method called *Random Histogram Forest (RHF)*, which builds a random forest while using the *Kurtosis* score as splitting criterion. The anomaly score of each instance is computed as the information content of the *leaf* it belongs to. We provide an extensive experimental evaluation on 38 datasets, including all datasets used as benchmarks for anomaly detection, to the best of our knowledge. Our experimental evaluation shows that our approach outperforms the other approaches in terms of average precision, while being simple and intuitive. Moreover, the running time of our algorithm grows linearly with the size of the input dataset.

Acknowledgment

This work has been carried out at LINCS (http://www.lincs.fr), in the frame of a cooperation between Huawei Technologies France SASU and Telecom Paris (Grant no. YBN2018125164). Any opinion, findings or recommendations expressed in this material are those of the author(s).

References

[1] D. Hawkins, *Identification of Outliers*. Chapman and Hall, 1980.
[2] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study", *Intelligent Data Analysis*, pp. 429–449, 2002.
[3] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest", in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.
[4] B. Schölkopf, R. Williamson *et al.*, "Support vector method for novelty detection", in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99, 1999.
[5] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers", in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD 00.
[6] A. Liaw and M. Wiener, "Classification and regression by randomforest", *R News*, vol. 2, no. 3, pp. 18–22, 2002.
[7] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders", in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD 17.
[8] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves", in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML 06, 2006.
[9] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data", *PLOS ONE*, 2016.
[10] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses", *Pattern Recognition*, vol. 74, pp. 406 – 421, 2018.
[11] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis", *Journal of the Royal Statistical Society*, 1999.
[12] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm", 2012.
[13] F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces", in *Principles of Data Mining and Knowledge Discovery*, 2002.
[14] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets", *SIGMOD Rec.*, 2000.
[15] A. Emmott, S. Das *et al.*, "A meta-analysis of the anomaly detection problem", *arXiv: Artificial Intelligence*, 2015.
[16] J. J. A. Moors, "The meaning of kurtosis: Darlington reexamined", *The American Statistician*, vol. 40, no. 4, pp. 283–284, 1986.
[17] C. E. Shannon, "A mathematical theory of communication." *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.
[18] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, 2009.
[19] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006.
[20] W. McKinney *et al.*, "Data structures for statistical computing in python", in *Proceedings of the 9th Python in Science Conference*, 2010.
[21] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection", *Journal of Machine Learning Research*, 2019.
[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel *et al.*, "Scikit-learn: Machine learning in python", *Journal of machine learning research*.
[23] [Online]. Available: https://github.com/anrputina/rhf
[24] D. Dua and C. Graff, "UCI machine learning repository", 2017.
[25] S. Rayana, "ODDS library [http://odds.cs.stonybrook.edu]", 2016.
[26] F. Salutari, D. Da Hora, G. Dubuc, and D. Rossi, "Analyzing wikipedia users perceived quality of experience: A large-scale study", *IEEE Transactions on Network and Service Management*, 2020.
[27] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms", in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.
[28] H. A. Sturges, "The choice of a class interval", *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.