Multivariate Automatic Tuning of Isolation Forest for Anomaly Detection in Critical Infrastructures: A Solution for Intelligent Information Systems

David Saavedra Pastor[1][a], José Vicente Berná Martínez[1][b], Lucia Arnau Muñoz[1][c] and Carlos Calatayud Asensi[2][d]

[1]*Department of Computer Science and Technology, University of Alicante, Spain*

[2]*Head of Reverse Osmosis, Aguas de Valencia S.A., Spain*

Keywords: Isolation Forest, Anomaly Detection, Auto-Tuning, Hyperparameter Tuning, Critical Infrastructure, Real-Time Monitoring.

Abstract: The Isolation Forest (IF) algorithm is effective in detecting anomalies in critical infrastructure, but its performance depends on the proper setting of five hyperparameters: sample size, number of trees, maximum tree depth, maximum number of features and detection threshold. Static tuning of these parameters is inefficient and poorly adaptable to dynamic environments. This paper proposes a multivariate autotuning method that automatically optimises these hyperparameters by: (1) adaptive adjustment of the sample size based on the standard deviation of the anomaly scores, (2) selection of the number of trees according to F1-score stabilisation, (3) control of the maximum depth based on the average isolation rate, (4) adjustment of the maximum number of features according to the variance of the data, and (5) optimisation of the detection threshold by minimisation of a cost function. The auto-tuning procedure has been validated in the detection of anomalies in drinking water networks, showing an F1-score improvement of 7.5% and a reduction of the execution time by 22.55% compared to static configurations, demonstrating its feasibility for real-time systems.

1 INTRODUCTION

Critical infrastructures such as water, energy and transport distribution networks rely on real-time monitoring systems to ensure their safety and efficiency. These systems generate large volumes of sensor data, where early detection of anomalies, such as leaks or outages, is crucial to optimise resources and prevent damage and losses. The Isolation Forest (IF) algorithm (Liu et al., 2008) has established itself as an efficient technique to identify anomalies in complex datasets by randomly partitioning the data space. Its simplicity and low computational cost make it ideal for real-time applications. However, its performance depends on the proper setting of the five hyperparameters that define it: sample size (S), number of trees (T), maximum number of features (F), maximum depth of trees (D) and detection threshold (Th). The static adjustment of these parameters leads to a lack of adaptability of the algorithm to the evolution of the information sources, which can cause instability to changes in the data, false positives (unnecessary alarms) or false negatives (undetected anomalies), compromising the efficiency and reliability of the model.

This paper proposes a method for multivariate dynamic self-adjustment of the IF hyperparameters without human intervention, improving their accuracy, adaptability and robustness. The approach integrates: (1) adaptive adjustment of S based on the standard deviation of the anomaly scores, (2) optimisation of T by F1-score stabilisation, (3) selection of F according to the variance of the

[a] https://orcid.org/0009-0006-2132-1307
[b] https://orcid.org/0000-0002-9007-6054
[c] https://orcid.org/0009-0008-4555-4409
[d] https://orcid.org/0009-0007-5744-631X

Pastor, D. S., Martínez, J. V. B., Muñoz, L. A. and Asensi, C. C.
Multivariate Automatic Tuning of Isolation Forest for Anomaly Detection in Critical Infrastructures: A Solution for Intelligent Information Systems.
DOI: 10.5220/0013710700004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 2: KEOD and KMIS*, pages 393-400
ISBN: 978-989-758-769-6; ISSN: 2184-3228

features, (4) control of *D* based on the average isolation rate and (5) adjustment of *Th* by minimisation of a cost function. To validate this proposal, the method has been tested on real drinking water infrastructures, on which the IF algorithm is evaluated after the dynamic self-adjustment processes. The results have shown an improvement of the F1-score by 7.50% (from 0.80 to 0.86) and a reduction of the execution time of the IF algorithm by 22.55% (from 5.10s to 3.95s) as it optimises the size of the datasets to the minimum necessary to maximise accuracy, demonstrating its potential for dynamic environments.

The paper is organised in five sections: section 2 reviews the state of the art of the IF algorithm and its variants, section 3 develops the proposed methodology, section 4 presents the experimental results and section 5 presents the conclusions and future research lines.

2 STATE OF THE ART

Isolation Forest (Liu et al., 2008) is a technique for anomaly detection that uses random partitions of the data space by means of decision trees. Its computational efficiency and simplicity have made it a popular tool for identifying anomalies in complex datasets. However, its performance depends on the proper configuration of its multiple hyperparameters and the deviations generated in high dimensionality datasets have motivated the development of variants to improve its accuracy and adaptability.

Extended Isolation Forest (EIF) (Hariri et al., 2019) replaces axis-parallel splits with random hyperplanes, reducing geometric deviations and improving detection in non-aligned distributions. However, EIF does not address dynamic hyperparameter optimisation, leaving this task to humans. *Cluster-Based Improved Isolation Forest* (CIIF) (Karczmarek et al., 2020) integrates 'k-Means' clustering to adapt partitions to spatio-temporal data, but introduces a new hyperparameter that makes automatic adjustment difficult, the number of clusters. *SCiForest* (Liu et al., 2010) optimises partitions using information gain criteria, speeding up anomaly isolation in multivariate systems, although it still relies on manual settings for parameters such as maximum depth or number of features.

Other more recent approaches such as the *Bilateral-Weighted Online Adaptive Isolation Forest* (Hannák et al., 2023) adjusts the weights on tree paths according to dynamic patterns and improves detection in data streams but also does not automatically optimise its parameters. The *Deep Isolation Forest* (Xu et al., 2023) incorporates neural networks to combine scores from different trees, increasing robustness in complex datasets, although at the cost of higher computational complexity and without an approach for multivariate adjustment. Finally, other hybrid methods, such as *Hybrid Isolation Forest* (Nalini et al., 2024), combine IF with clustering techniques, but their effectiveness still depends on static manual configuration. Some variants adjust parameters such as the number of trees or the detection threshold by means of rules of thumb, but these rules again generate a truly static configuration.

There is no comprehensive solution to the algorithm's hyperparameter tuning problem as some proposals optimise certain parameters while leaving others untuned (Bischl et al., 2024). The parameter tuning problem generates such a high impact on model accuracy that approaches can now be found that address the issue by combining techniques that make up for the tuning shortcoming, such as (Priyanto and Purnomo, 2021) where IF is combined with Long Short-Term Memory so as not to require continuous fine-grained hyperparameter tuning. Although the proposal manages to generate an adequate fit, low ROC curve values are still observed because the anomaly detection method can only detect anomalous samples with a huge False Positive and False Negative value. In (Dhouib et al., 2023), a method for generalisation of the score function motivated by information theory is proposed and used to aggregate the scores of tree estimators so that the anomaly detection threshold can be optimised, although no systematic improvement was achieved and even in some datasets poor performance was generated. On the other hand, in (Lee et al., 2020) a sequential model-based optimisation (SMBO) method is used, which includes Bayesian optimisation and tree-structured parameter estimators (TPE), the problem it has is that it uses a random search so that areas of the search space may be left unexplored, falling into local optima.

In this work we propose a multivariable autotuning system that automatically optimises its hyperparameters. By obtaining a procedure that can be executed autonomously, when necessary, IF can dynamically self-configure optimising its efficiency and validity in real-time environments. When data evolves, e.g. changes between day/night, holiday/work, the autotuning process can be run to optimise IF and restore its accuracy.

3 METHODOLOGY

Table 1 summarises the techniques used in each hyperparameter (HP) for automatic tuning together with the initial default value.

Table 1: Auto-tuning process and default values of the HP.

HP	Process Adjustment	Defalt
S	Standard deviation (σ).	256
T	Metric stabilisation (F1-score).	100
F	Variance (σ^2) .	1.0
D	Average isolation rate (*R*).	$Log_{(2)}(S)$
Th	Cost function (*FP*, *FN*)	1.0

During the tuning process, default values shall be used for those hyperparameters that have not been calculated. When the appropriate value of a hyperparameter is set, the optimal value shall be used for the calculation of the next hyperparameter.

3.1 Sample Size Adjustment

The sample size (*S*), which defaults to 256, defines the subset of data used to construct each tree, directly affecting the stability of the anomaly scores. A too small *S* can generate unrepresentative trees, while an excessive *S* increases the computational cost without significant improvement, especially in datasets with high variability. To optimise *S* automatically, we propose a method based on the variability of the anomaly scores, measuring their standard deviation (σ). This approach adjusts *S* according to its dispersion, ensuring a trade-off between accuracy and efficiency. First, the standard deviation σ_o of the original set *Dat* is calculated. The adjustment process will start from *S=256* and search for a minimum *S* sufficient to achieve a minimum set standard deviation σ_{min} similar to the original set standard deviation with a maximum error e_σ, formally defined:

$$\min(S) \rightarrow \sigma_{min} \leq \sigma_o \pm e_\sigma \quad (1)$$

Input: *Dat* (original dataset*), S=S_initial* = 256, e_σ*=0.05*, Inc_{Dat}= 0.1
Output: Adjusted *S*

σ_o = standard_deviation(Dat)
σ_{min} = standard_deviation(Dat(*S*))
While *S* < card(*Dat*) **AND** $\sigma_{min} > \sigma_o \pm e_\sigma$
  S = S*(*1*+Inc_{Dat})
  σ_{min} = standard_deviation (Dat(*S*))
EndWhile
Return *S*

Pseudocode 1: Adaptive adjustment of the sample size (*S*).

The use of standard deviation (σ) is due to its sensitivity to dispersion and its low computational cost (*O(n)*), compared to alternatives such as entropy or interquartile range. This process will be iterative, showed in Pseudocode 1, incrementing S by a percentage $Inc_{(Dat)}$ (e.g. 10%) until the condition is met or S=card(*Dat*).

3.2 Number of Trees Adjustment

In the IF algorithm, the number of trees (*T*), set by default to 100, determines the number of random partitions used to compute the anomaly scores, and influences the robustness and computational cost of the algorithm. An insufficient *T* generates inaccurate results, while an excessive *T* increases the execution time without significant improvements in accuracy.

To adjust *T* we propose a method based on the stabilisation of the F1-score performance metric. The process will use values of *T_val*=[5,100] with *step* jumps of *step=5*, starting from value 5 until reaching the appropriate level, and maximum 100. To evaluate the performance, we generate an artificially contaminated subset of the original dataset (*Dat*), using the previously calculated *S*, introducing 1% of anomalies by a random increase of 50% in a randomly selected feature, simulating a peak value. The artificial 1% contamination ensures that the anomalies are rare but detectable. The procedure, Pseudocode 2, iterates by running IF with values in *T* of *T_val*, identify as anomalies the 1% of the data with the highest anomaly scores and calculate the F1-score by comparing these identified anomalies with those actually contaminated.

Input: *T_val = [5,100]*, step=*5*, *N = 3*, $F1_{sta}$*= 0.01*, *S*=calculated in section 3.1, *T=0,* Dat'(S) = Contaminate 1% of Dat(S)
Output: Adjusted *T*

F1-score-list = [] // List to store F1-scores, empty
While
  T =T + step
  Run IF with *T* trees on *Dat'*(*S*) -> *scores*
  Calculate F1-score (*scores*)
  F1-score-list.stack(F1-score)
  If (len(*F1-score-list)<N)* **Continue**
  meet=0
  For *i*= len(*F1-score-list*) - N to len(*F1-score-list*)
   If *F1-score-list[i]*≤ $F1_{sta}$ meet=+1
  If meet=*N* **Return** *T*
Endwhile

Pseudocode 2: Calculation of the number of trees (*T*).

F1-score will be considered stable when in *N* iterations $F1\text{-}score \leq F1_{sta}$ is obtained, accepting as

the value of *T* that of the last iteration. In our proposal *N=3* to avoid local stabilisations and $F1_{sta}=0.01$ to require high accuracy. If F1-score does not stabilise *T*=100.

3.3 Maximum Features Adjustment

The maximum number of features (*F*) defines the proportion of features used in each random partition. An *F* that is too high may incorporate noise, while an *F* that is too low reduces the ability to detect anomalies in features that are not being considered. To optimise *F* we use the variance (σ^2) of the features.

The process, Pseudocode 3, starts with all features *F=1*.0 of the original dataset (*Dat(S)*). The first step is to normalise the values of *Dat* by scaling the values of each feature to a common range, e.g. [0,1]. Subsequently, the variance of a subset of features is calculated according to the ratio *F* and the average variance (*V_prom*) is obtained by averaging the variances of each feature. The boundaries of quartile 1 (B_{Q1F}) and quartile 4 (B_{Q4F}) of the set of variances generated by:

$$B_{Q1F} = \frac{\max\_value(Q1)+\min\_value(Q2)}{2} \quad (2)$$

$$B_{Q4F} = \frac{\max\_value(Q3)+\min\_value(Q4)}{2} \quad (3)$$

Input: *Dat* (original dataset), *F=F_initial* = 1.0, *α_reduction* = 0.5, *α_increase* = 1.5, *F_min* = 1/num_feat(*Dat*), *F_max* = 1.0
Output: *F* adjusted

Normalise *Dat* by scaling each characteristic to [0,1].
num_features = num_feat(*Dat*)
While *true*:
  selected_features = SelectRandom(*F* * num_features, *Dat*).
  σ^2= CalculateVariance(*selected_features*)
  V_prom= Average(σ^2)
  Q1, *Q2*, *Q3*, *Q4*= CalculateQuartiles(σ^2)
  B_{Q1F}= (max_value(*Q1*) + min_value(*Q2*)) / 2
  B_{Q4F}= (max_value(*Q3*) + min_value(*Q4*)) / 2
  If *V_prom*> $_{B(Q4F)}$:
    If *F* * *α_reduction*< F_min:
      Return *F*
    F= F * *α_reduction*
  Else If *V_prom* < $_{B(Q1F)}$:
    If *F* * *α_increase* > F_max:
      Return F
    F = F * *α_increase*
  Else:
    Return *F*
  UpdateModelIF(*F*)
EndWhile
Return *F*

Pseudocode 3: Setting the maximum number of features (*F*).

If *V_prom* exceeds B_{Q4F}, it indicates high dispersion and possible noise so we would want to decrease the value of F as long as it allows to select at least 1 feature. If *V_prom* is less than B_{Q1F}, it indicates loss of information, so we would seek to increase *F*, selecting more features as long as $F \leq 1$.

The value of increasing or decreasing *F* is done by multiplying *F* by a constant *α*. The values of *F* are restricted to the range [F_min=1/num_feat(*Dat*), *F_max=1*.0]. After each adjustment, the variance is recalculated with the new *F* and the IF model is updated with the new *F* value.

3.4 Setting the Maximum Depth

The maximum depth (*D*), usually defined in IF as $log_{(2)\ (}S)$, limits the growth of trees. Too low a *D* prevents complex anomalies from being detected, while too much *D* can generate unnecessarily deep trees, increasing the computational cost. To optimise *D*, Pseudocode 4, we propose a method based on the average isolation rate (*R*), defined as the average depth at which anomalies are isolated.

Input: *Dat* (original dataset), *R_75*, *R_25*, D_min = 1, *D=D_max* = $log_{(2)\ (}S)$, *β* = 0.2
Output: *D* appropriate

Contaminate 5% of *Dat* to 50% identify anomalous records
While
  Execute IF with *D* on *Dat*
  Identify top 5% of points with highest score
  Create anomaly set
  Calculate *R* as the average of the elements of C_R
  If *R* < *R_75* **AND** *D* > D_min:
    D = max(*D**(1-β), *D_min*) // Reduce *D*
  Else if *R* > *R_25* **AND** *D* < D_max:
    D= min(*D**(1+β), *D_max*) // Increase *D*
  Else:
    Return *D*
EndWhile
Return *D*

Pseudocode 4: Setting the maximum depth (*D*).

The process starts by contaminating 5% of the dataset (*Dat(S)*) by incrementing a random feature by 50% to identify anomalous records. We initialise *D* as *D_max= log₂(S) and D_min=1 (minimum height of 1 node)*, where *S* is the sample size. We run the IF algorithm on *Dat(S)*, create a set $C_R$with the isolation depths of the anomalies and calculate *R* as the average of the elements of C_R. We compare *R* with the thresholds *R_25* (quartile Q1) and R_75 (quartile Q3) of the isolation depths. If *R* is less than R_75 and *D>D_min*, it indicates slow isolation and possible

overfitting, so we reduce D by $D=max(D\times(1-\beta),D\_min)$. If R is below R_25 and $D<D\_max$, it indicates that anomalies require more partitions, so we increase D by $D=min(D\times(1+\beta), D\_max)$. If R is between R_25 and R_75, D is returned. We restrict D to the range $[D\_min=1, D\_max= log_{(2)}(S)]$ to balance accuracy and efficiency, and β will be a value less than 1 and greater than 0, whose size will depend on the willingness with which we wish to make the adjustment, empirically we use $\beta=0.2$.

This fitting is done iteratively, updating the IF model with the new D at each iteration to adapt to changes in the complexity of the data.

This method is based on the fact that anomalies tend to be isolated in fewer partitions than normal points, a principle of the IF algorithm. The adjustment rate $\beta=0.2$ produces gradual changes, avoiding large oscillations. The thresholds R_25 (quartile Q1) and R_75 (quartile Q4) are calculated from the isolation depths of the contaminated anomalies, representing the expected distribution of the isolation rate in the dataset. The R_25 and R_75 thresholds are obtained in an initial analysis by running IF with $D=log_{(2)}(S)$ on a contaminated dataset (5% anomalies), calculating the isolation depths of the anomalies and determining the 25 and 75 quartiles of these depths, reflecting the expected distribution of the isolation rate. This approach ensures that the adjustment of D is sensitive to the complexity of the anomalies, optimising detection without unnecessary computational costs.

3.5 Detection Threshold Adjustment

The detection threshold (Th) determines at what score an element is classified as anomalous. This threshold is crucial to control the false positive (FP) and false negative (FN) rate. A too low Th may generate unnecessary alarms by marking records as anomalies when they are not, while a too high Th may not detect critical anomalies.

To optimise Th we propose a method based on the minimisation of a cost function that weights the classification errors according to their impact on the application context, Pseudocode 5. The cost function $FC(Th)$ assigns a weight to errors due to false positives and false negatives, so that it can give more importance to one type of failure or the other.

$$FC(Th) = \delta \cdot FP(Th) + (1-\delta) \cdot FN(Th) \quad (4)$$

Where FP and FN represent the false positive and false negative rates, and δ is a weight reflecting the relative importance of each type of error. If $\delta=0.5$, both errors have the same weight. This value is selected empirically after a preliminary analysis of the impact of the errors.

Before evaluating FC(Th), we generate a dataset from the original dataset of size S, $Dat(S)$, contaminating it with 5% anomalies, increasing by 50% a random feature in the selected records to simulate anomalous events and identify exactly the contaminated records. Next, we apply a binary search on the range of Th between $Th\_min=0$ and $Th\_max=1.0$, initialising $Th=0.5$. The binary search evaluates the cost function FC at the intermediate points $(Th+Th\_min)/2$ and $(Th\_max+Th)/2$. If $FC((Th+Th\_min)/2)$ is less than $FC((Th\_max+Th)/2)$, $Th\_max=Th$ and $Th=(Th+Th\_min)/2$ are updated; otherwise, $Th\_min=Th$ and $Th=(Th\_max+Th)/2$ are updated. This process continues as long as $Th\_max-Th\_min \geq grad$ with $grad=0.01$, ensuring accuracy with low computational cost.

Input: *Dat* (original dataset), δ = 0.2, *Th_min* = 0, *Th_max* = 1.0, *grad* = 0.01, *Th* = 0.5, FC (cost function: *FC(Th)=δ-FP(Th)+ (1-δ) -FN(Th))*
Output: *Th* appropriate

Contaminate_5% of *Dat* (increase 50% of a characteristic)
While *Th_max - Th_min ≥ grad*:
  mid1 = (*Th* + *Th_min*)/2
  mid2 = (*Th_max* + *Th*)/2
  If FC(*mid1*) < FC(*mid2*):
    Th_max = *Th*
    Th = *mid1*
  Else:
    Th_min = *Th*
    Th = *mid2*
EndWhile
Return *Th*

Pseudocode 5: Setting the detection threshold (Th).

The binary search is selected for its efficiency ($O(log\ n)$). The 5% contamination level reflects a realistic estimation of anomalies in critical systems, and allows to accurately identify anomalous records to calculate FP and FN in the cost function. The initial value of $Th=0.5$ is selected as the midpoint of the range [0, 1.0] for the binary search. The weight $\delta=value$ can be adjusted according to the domain, in this case $\delta=0.2$ is empirically selected to prioritise false negatives (FN), relevant in critical systems where anomaly detection is a priority. For example, in contexts where FNs are more critical, values such as $\delta=0.6$ could be used. The $grad=0.01$ ensures accurate Th by limiting the difference between Th_max and Th_min, avoiding unnecessary iterations.

4 EXPERIMENTATION

In this section we will compare the auto-tuned model with static configurations, evaluating its adaptability, accuracy and computational efficiency. The implementation was done in Python 3.10 using the 'scikit-learn' library (version 1.6). The experiments were run on a computer with an Intel Core i9-13900H (2.50 GHz) processor, 32 GB of RAM, an NVIDIA GTX 4060 GPU and the Windows 11 operating system in its 24H2 version. We repeated each test three times and averaged the results to reduce variability and ensure statistical stability.

We use a real dataset from the year 2024, from a drinking water distribution network in a city in southeastern Spain. It contains 33096 anonymised records with measurements every 15 minutes of 10 features, including the following data: water level, flow (inflow and outflow), discharge and pressure in 4 geographical zones (Z1, Z2, Z3, Z4). A realistic scenario with heterogeneous data is presented, whose units of measurement and accuracy are detailed in Table 2.

Table 2: Identifiers, accuracy and units of the dataset.

Characteristics	ID	Accuracy	Units
Date	F	1	day
Time	H	15	minutes
Level Z4	N Z4	0.01	m height
Level Z1	N Z1	0.01	height
Inlet flow Z4	F-E Z4	0.01	m3/h
Output flow Z4	F-S Z4	0.01	m3/h
Flow UII Z3	F-UII Z3	0.01	m3/h
Discharge UII Z3	I-UII Z3	0.01	m3/h
Discharge Z2	I Z2	0.01	m3/h
Discharge UII Z1	I-UII Z1	0.01	m3/h
Pressure UII Z3	P-UII Z3	0.01	mca
Pressure Z2	P Z2	0.01	mca

4.1 Sample Size (S)

Following the process described in section 3.1, we start with $S=256$ and adjust S iteratively, increasing it by 10% (S=S×($1+Inc_{Dat}$), with Inc_{Dat}=0.1) if σ_{min} exceeds the threshold $\sigma_o \pm e_\sigma$, where σ_o is the standard deviation of the original set *Dat* and e_σ=0.03 is the maximum allowed error. A small error has been adjusted as a large similarity in deviation is sought. The adjustment stops when $\sigma_{min} \leq \sigma_{(o)} \pm e\sigma$ or S≥card(*Dat*).

Table 3: Evaluation of sample size (S).

Iteration	S	σ_{min}
Start	**256**	**0.56**
Autotuned It. 1	282	0.56
Self-adjusted It. 2	310	0.56
Self-adjusted It. 3...	341	0.55
Self-adjusted It. 34	**6540**	**0.48**

In Table 3, the behaviour of the method is presented, showing how S adapts to the variability of the data compared to a static setting of $S=256$. The data set shows a $\sigma_o=0.45$, so we will look for an o_{min} that meets the given constraint. The autotuning method manages to stabilise σ_{min}, bringing it close to σ_o with an error within the allowed range (±0.03) at iteration 34, where S is set to 6540. This demonstrates that the adaptive adjustment of S achieves an effective trade-off between accuracy and computational efficiency. The working dataset will consist of 19% of the original.

4.2 Number of Trees (T)

Following the process described in section 3.2, an artificially contaminated subset (1% anomalies) is created, using the calculated value of S ($S=6540$). The fit iterates T in the range [5, 100] with steps of 5, stopping when the F1-score variance is less than 0.01 for $N=3$ consecutive iterations, with a maximum of $T\_max=100$.

Table 4 shows the F1-score value with the different values of T and how with $T=40$ a value similar to the static default value of 100 is obtained.

Table 4: Evaluation of the number of trees (T).

Iteration	T	F1-score
Static	**100**	**0.82**
Auto-tuned It. 1	5	0.84
Self-adjusted It. 2	10	0.86
Self-adjusted It. 3...	15	0.87
Self-adjusted It. 8	**40**	**0.86**

This T saves up to 60% of the computation time regarding the number of trees.

4.3 Number of Features (F)

In Section 3.3, a method was proposed that adapts F according to the average variance (V_prom) of the features, reducing F to $F×0.5$ if V_prom exceeds B_{Q4F} (75th quartile), or increasing it to $F×1.5$ if it falls below B_{Q1F} (25th quartile), within the range [F_min=0.1, F_max= 1.0]. The process uses $S=6540$ and $T=40$, calculated previously.

Table 5 shows how *F* adjusts to the variability of the data and its impact on the F1-score. The autotuning method adjusts F to a value of 0.84, reducing the average variance (*V_prom*) so that it remains between the thresholds B_{Q1F} and B_{Q4F}.

Table 5: Evaluation of the maximum number of features (*F*).

Iteration	*F*	B_{Q1F}	B_{Q4F}	*V_prom*
Static	**1.0**	**0.14**	**0.23**	**0.33**
Autotuned It. 1	0.5	0.16	0.23	0.12
Self-adjusted It. 2	0.75	0.16	0.23	0.28
Self-adjusted It. 3	0.38	0.16	0.23	0.12
Autotuned It. 4	0.56	0.16	0.23	0.15
Autotuned It. 5	**0.84**	**0.16**	**0.23**	**0.18**

4.4 Maximum Depth (D)

In Section 3.4, a method was proposed that adapts *D* according to the average isolation rate (R), increasing *D* by 20% ($D=min(D\times(1+\beta), D\_max)$) if *R* exceeds R_75 (75 quartile), or reducing it by 20% ($D=max(D\times(1-\beta), D\_min)$) if it falls below R_25 (25 quartile), with $\beta=0.2$, within the range $[D\_min=1, D\_max=log_{(2)}(283)=8.15]$. The *R_25* and *R_75* thresholds were initially calculated by running IF with $D= log_2(6540)$ on a 5% contaminated dataset, determining the *25th* and *75th* quartiles of the anomaly isolation depths. The process uses *S=6540*, *T=40*, and *F=0.84*, calculated previously. Table 6 shows how *D* fits the complexity of the data, compared to a static configuration with $D=log_2(6540) =12.31$.

Table 6: Evaluation of the maximum depth (*D*).

Iteration	*D*	*R*
Static	**12.31**	**6.5**
Autotuned It. 1	9.84	6.4
Self-adjusted It. 2	7.87	6.3
Self-adjusted It. 3	**6.30**	**6.3**

The autotuning method adapts *D* to an average value of 6, adjusting to the isolation rate (*R*=6.3). The adjustments based on *R_25* and *R_75* avoid over-fitting and optimise the detection of subtle anomalies, while maintaining computational efficiency.

4.5 Threshold (Th)

Following section 3.5, Th is optimised by minimising the cost function $FC(Th)=\delta\text{-}FP(Th)+(1\text{-}\delta)\text{-}FN(Th)$, with $\delta=0.2$, using binary search on a contaminated subset (5% of anomalies) within the range [*Th_min=0*, *Th_max=1*], with a gradient of 0.10. The value of $\delta=0.2$ was selected to prioritise false negative (*FN*) detection, crucial for identifying critical anomalies. The process uses *S=6540* and *T=40*, *F=0.*84 and *D=6*, calculated previously.

Table 7 shows how Th converges and its effect on false positive (*FP*), false negative (*FN*) and cost (*FC*) rates, compared to the fixed static setting of *Th=1.*0.

The autotuning method optimises *Th* to an average value of 0.88, with an average FC of 0.07. The prioritisation of *FN* ($1\text{-}\delta=0.8$) ensures high sensitivity in anomaly detection,

Table 7: Evaluation of the detection threshold (*Th*).

Iteration	*Th*	*FP*	*FN*	FC
Static	**1.0**	**0.15**	**0.9**	**0.30**
Autotuned It. 1	0.50	0.45	0.08	0.37
Self-adjusted It. 2	0.75	0.32	0.12	0.28
Self-adjusted It. 3	**0.88**	**0.08**	**0.06**	**0.07**

4.6 Comparison and Limitations

The auto-tuning method dynamically optimises the five hyperparameters of the IF algorithm (*S, T, F, D, Th*) for real-time anomaly detection. The performance of the auto-tuned model, based on the values obtained in sections 4.1 to 4.5 (*S*= 6540, T=40, *F=0.*84, D=6, *Th=0.*88), is then compared with the usual static configuration *(S=33096, T=100, F=1.0, $D=log_2(33096)= 15.01$, Th= to be determined)*, using the water dataset of a drinking water distribution network. The auto-fit model was evaluated on the basis of F1-score and runtime. Table 8 presents the comparative results.

Table 8: Comparative analysis static vs auto-tuned.

Configuration	*S*	*T*	*F*	*D*	*Th*	F1-score	Time (s)
Static	33096	100	1.0	15	0.89	0.80	5.10
Autotuned	6540	40	0.84	6	0.88	0.86	3.95

For this example, after several tests it was determined that in the static model, a *Th=0.89* provided acceptable detection. The auto-tuned model improves the F1-score by 7.50% (from 0.80 to 0.86) and reduces the run time by 22.55% (from 5.10s to 3.95s) versus the static configuration. These gains reflect the method's ability to adapt to the heterogeneity of the water dataset, optimising the detection of anomalies such as leaks with greater sensitivity and efficiency. The elimination of manual intervention ensures adaptability in dynamic environments, overcoming the limitations of static configurations as the model can self-configure when accuracy drops.

However, the proposed method has some limitations. The quality of the thresholds B_{Q1F}, B_{Q4F}, *R_25* and *R_75* depends on an initial analysis, which could be problematic in incomplete or noisy datasets. Furthermore, the method assumes that anomalies are

rare and detectable by significant increments, which may not apply to datasets with subtle anomalies or highly skewed distributions.

5 CONCLUSIONS

This paper presents a multivariate autotuning method that dynamically optimises the five hyperparameters of the IF algorithm (*S*, *T*, *F*, *D, Th*), validated on a real dataset of a water distribution network in 2024. The proposed procedure allows the model to self-adjust and also to do so obtaining better F1-score results and processing time than using the default values that are traditionally configured.

As an automatic procedure, it can be launched periodically, ensuring that the IF model will be adjusted to the data conditions at that instant. The method provides a robust, adaptive and scalable solution for real-time systems. Above all, it allows independence from the skill and knowledge of the expert fitting the model, providing a completely independent and objective methodology. Compared to approaches such as *Extended IF* or *Deep IF*, it stands out for its simplicity and simultaneous optimisation of all hyperparameters. It does not require the introduction of new parameters such as *CIIF*, which make the process more complex, nor does it add high computational complexity such as *LSTM*, since self-adjustment can be performed while IF performs detection. Finally, the proposed method allows the hyperparameters to be adjusted continuously, unlike *HIF* or *SCiForest*, which ensures that the algorithm will maximize its accuracy.

In the short term, future research will address the sensitivity of the method to noisy or incomplete data, incorporating advanced data quality assurance techniques. In addition, we will also validate this procedure using other critical infrastructure datasets, such as power grids, to extend its applicability. The objective is to study whether there is similarity in the behaviour of the hyperparameters. In the medium term, the aim is to incorporate techniques that detect the ideal moment to recalculate the hyperparameters, for example by detecting a degradation in accuracy due to changes in data trends.

ACKNOWLEDGEMENTS

This work has been supported by Project PID2023-152566OB-I00 "Preventive Maintenance of digital infrastructures through the application of Artificial Intelligence for the diagnosis and prediction of anomalies (PreMAI)" funded by MICIU/AEI /10.13039/501100011033 and by ERDF, EU; project UAIND22-01B - "Adaptive control of urban supply systems" of the Vice-rectorate for Research of the University of Alicante; and project co-financed by the Valencian Institute for Competitiveness and Innovation (IVACE+i) and is eligible for co-financing by the European Union (Exp. INNTA3/ 2022/3).

REFERENCES

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., ... & Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *13*(2), e1484.

Dhouib, H., Wilms, A., & Boes, P. (2023). Distribution and volume based scoring for Isolation Forests. arXiv preprint arXiv:2309.11450.

Hannák, G., Horváth, G., Kádár, A., & Szalai, M. D. (2023). "Bilateral-Weighted Online Adaptive Isolation Forest for Anomaly Detection in Streaming Data." Statistical Analysis and Data Mining: The ASA Data Science Journal, 16(3), 215-223.

Hariri, S., Kind, M. C., & Brunner, R. J. (2019). "Extended Isolation Forest." IEEE Transactions on Knowledge and Data Engineering, 33(4), 1479-1489.

Karczmarek, P., Kiersztyn, A., Pedrycz, W., & Al, E. (2020). "K-Means-Based Isolation Forest." Knowledge-Based Systems, 195, 105659.

Lee, C. H., Lu, X., Lin, X., Tao, H., Xue, Y., & Wu, C. (2020, April). Anomaly detection of storage battery based on isolation forest and hyperparameter tuning. In *Proceedings of the 2020 5th I.Conference on Mathematics and Artificial Intelligence* (pp. 229-233).

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). "Isolation Forest." In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2010). "On Detecting Clustered Anomalies Using SCiForest." In J. European Conf. on Machine Learning and Knowledge Discovery in Databases (pp. 274-290). Springer.

Nalini, M. yamini, B., Ambhika, C., & Siva Subramanian, R. (2024). "Enhancing Early Attack Detection: Novel Hybrid Density-Based Isolation Forest for Improved Anomaly Detection." International Journal of Machine Learning and Cybernetics, 1-19.

Priyanto, C. Y., & Purnomo, H. D. (2021, September). Combination of isolation forest and LSTM autoencoder for anomaly detection. In 2021 2nd International Conference on Innovative and Creative Information Technology (ICITech) (pp. 35-38). IEEE.

Xu, H., Pang, G., Wang y., & Wang y. (2023). "Deep Isolation Forest for Anomaly Detection." IEEE Transactions on Knowledge and Data Engineering, 35(12), 12591-12604.