


Large-scale automated forecasting for network safety and security monitoring

Roi Naveiro  | Simón Rodríguez | David Ríos Insua

Institute of Mathematical Sciences,
ICMAT-CSIC, Madrid, Spain

Correspondence

Roi Naveiro, Institute of Mathematical
Sciences, ICMAT-CSIC, 28049 Madrid,
Spain.

Email: roi.naveiro@icmat.es

Funding information

Spanish Ministry of Education,
Grant/Award Number: FPU 15/03636
PhD scholarship; Spanish Ministry of
Science, Grant/Award Number: FPI
SEV-2015-0554-16-4 PhD scholarship;
Spanish Ministry of Science, Grant/Award
Number: RTC-2017-6593-7 and
MTM2017-86875-C3-1-R; AXA-ICMAT
Chair on Adversarial Risk Analysis

Abstract

Large-scale real-time streaming data pose major challenges to forecasting, in particular, defying the presence of human experts to perform the required analysis. We present here a class of models and methods used to develop an automated, scalable, and versatile system for large-scale forecasting oriented toward network safety and security monitoring. Our system provides short- and long-term forecasts and uses them to detect issues, well in advance, that might take place in relation with multiple Internet-connected devices.

KEYWORDS

Bayesian methods, dynamic models, forecasting, network monitoring, real-time predictive analytics

1 | INTRODUCTION

As outlined in the work of Mortenson et al,¹ leveraging big data and real-time analytics constitute two main current research venues. Information and communication technologies (ICT) have experienced an exponential growth in the last few decades and most human activities, businesses, and devices strongly depend on them.² With the advent of the Internet of Things (IoT), this interrelation will become even more evident and change dramatically the way in which different components of business and service systems interact. In parallel, risks concerning the security of ICT systems are also growing, as pointed out, eg, in the work of The Geneva Association.³ Such cyber risks can be of natural origin or man-made, the latter potentially originating from human failure or intentional threats, as in cyber crime. These actually constitute a current major global trend, as reflected, eg, in the Global Risks Map.⁴ As an example, the technical report of McAfee and CSIS,⁵ catalogs around 70 new potential cyber threats per minute and estimates the annual cost to the global economy from cyber crime to be above 400 billion USD. This impact highlights the need for developing solid cybersecurity risk management frameworks and the central role that these will play in business and industrial security in the near future. In addition, the increasing significance and availability of data in every business-related activity emphasizes data-driven approaches to improve cybersecurity decision making. As an example, Sedgewick⁶ provides a set of standards and guidelines supporting such frameworks, covering key cybersecurity activities. Two of them refer to continuous safety monitoring of Internet-connected devices (ICDs) and anomaly detection. This has led to the development of tools that periodically collect high-frequency information from the ICDs of an organization to support their monitoring. Given the increasing relevance of ICT, we may need to face organizations with several hundred thousands of such devices, from

which we obtain tens of variables every few minutes. This poses tremendous challenges in processing such enormous amounts of data and making the relevant forecasts and decisions in real time to mitigate, sufficiently in advance, potential, or actual safety and security issues in a network. In particular, it is virtually impossible to analyze each individual device time series through people, creating the need for an automated framework and system. Moreover, within this context, many standard time series analysis models become useless either because they cannot tackle a huge amount of high-frequency data or because they cannot be used in an automated fashion because of the versatility of the series that need to be faced.

We introduce in this paper a framework for time series monitoring and anomaly detection, which serves as basis for a system for large-scale safety and security network monitoring. Functionally, we require the system to be:

- *Automatic.* Given the huge amount of series to be monitored, intervention of humans in the process should be kept to a minimum.
- *Versatile.* Time series may be of different nature and have different characteristics such as linear growth, seasonality, or outbursts. The approach should be able to deal with all these particularities in an automated fashion. In addition, the time series features may change over time and, thus, the framework should be able to adapt fastly and automatically.
- *Scalable.* The approach should scale both in time and memory space. It should be fast enough to be able to cope with many very high-frequency time series. In addition, because of the huge amount of time series to be monitored, it should be able to summarize each series with just a few parameters, avoiding storage of the whole series.
- *Accurate.* Besides, we would also like the system to fulfill the required good statistical properties in relation with the predictive accuracy of the provided forecasts.

Earlier work in network monitoring does not fully cover the above requirements. In his classic paper,⁷ Brutlag proposed a simple approach based on Holt-Winters forecasting; however, model parameters need to be set and tuned for each model to work well, complicating automation. In addition, his technique is not capable of tracking several seasonal periods, reducing its versatility. Taylor and Letham⁸ recently proposed an analyst-in-the-loop algorithm, which makes use of human and automated tasks, precluding full automation. Moreover, they essentially frame the monitoring problem as a curve-fitting exercise using Generalized Additive Models, not fully taking into account the temporal dependence structure in the data. This way, the dynamic nature of the algorithm and its adaptability to sudden changes are essentially lost. In the work of Vallis et al.,⁹ a tool is presented for anomaly detection based on a seasonal trend loess decomposition together with the Generalized Extreme Student Deviation (GESD) test to detect anomalies. This algorithm handles anomaly detection issues but does not accomplish other important monitoring tasks in relation with forecasting potentially dangerous future events. The main shortcoming is its rigidity in adapting to situations, in which the signal features change considerably. Classification and regression trees¹⁰ are also popular methods for anomaly detection. They may be used to provide point and interval forecasts as well as detect anomalies through GESD or Grubbs' tests; their main disadvantage is that, a growing number of features can impact computational performance, quickly jeopardizing scalability. Autoregressive integrated moving average (ARIMA) models have also been widely used in this area.¹¹ These models assume that the signal under study is stationary, possibly after differencing, which is not always the case in many network monitoring time series. In addition, numerous parameters such as the number of differences should be selected in advance, complicating automation. Finally, long short-term memory neural networks, if properly built, may perform successfully in anomaly detection tasks.¹² However, their main drawback stems from the complex work required to adjust them to reach a proper performance level, rendering automation virtually infeasible.

Our approach represents a step toward the achievement of a completely automated, scalable, and versatile system for large-scale time series monitoring and anomaly detection in the specific area of network safety and security monitoring. In this domain, aberrant behavior identification is often based on heuristic methods developed by analysts and usually lacks predictive capabilities. The framework we propose aims at providing such predictive power to the monitoring system in an automated way. For that purpose, we use a Bayesian approach differentiating between continuous and discrete series. For continuous ones, we use a modified version of dynamic linear models (DLMs)¹³ that incorporates the eventual existence of regular outbursts originated by physical processes such as backups or compressions, especially relevant in our application domain. The main advantage of DLMs is that they can be constructed using different blocks capturing each of the specific features of the time series typical of our domain. We provide an effective way to automatically identify the involved models. In addition, these summarize the relevant aspects of the series in a few parameters, facilitating space scalability. Moreover, computation of the predictive distributions is relatively fast, thus making the approach time scalable after appropriate tuning. For discrete series, we use discrete time Markov chains,¹⁴ which fulfill also our aforementioned desiderata. We emphasize that our aim with our approach is not that of providing a more accurate time series class of

models to the previously existing ones, but rather to mix existing solutions in this automated fashion to create a framework that meets the aforementioned requirements.

The structure of this paper is as follows. A description of the models and their goals are given in Section 2, together with how they are identified and how forecasts are made. This helps us in covering the versatility and automaticity requirements. We then discuss implementation details in Section 3, covering scalability requirements while in Section 4, we analyze the predictive accuracy performance requirements through empirical tests in different series. We end up with some remarks.

2 | PROBLEM FORMULATION AND MODEL DESCRIPTION

Consider, for the moment, the case of monitoring a single time series that, in our domain, will refer to some performance measure of an ICD, say the number of users connected to a device or the percentage of disk storage occupied. Associated with the series, there are two reference values, designated W (*warning*) and C (*critical*), linked with observation levels such that, if exceeded, should lead to issuing warning and critical signals, respectively. For example, for a storage disk, we could be monitoring its usage and set $C = 0.95$, meaning that when we reach a saturation of 95%, disk performance might degrade and even collapse, inducing a loss. Such thresholds will depend on the device and its overall relevance. Observe that we have established a two-level alarm system to try to provide richer information about potential failures of the monitored device. In the aforementioned example, setting $W = 0.9$ would allow us to raise awareness before it is too late.

As pointed out, several key network monitoring cybersecurity activities are related with safety monitoring and anomaly detection within time series, which, in turn, we shall base on:

- a. Making short term forecasts. These allow us to:
 - Identify anomalous behavior of the series when observed values do not lie within predictive intervals. This is related with security and could be useful in pointing critical issues in advance or detecting intruders in a system.
 - Identify unsafe behavior when the predictive intervals actually cover the W and/or C thresholds.
- b. Making long term forecasts. These allow us to foresee critical issues sufficiently in advance when the W and/or C thresholds appear in long-term predictive intervals: critical values that lie within them point to potentially problematic behavior in the distant future.

To accomplish such tasks, we need procedures to make point and interval predictions of the involved time series. The width of the intervals should be adaptable to control for false positives, as well as to take into account the value of the corresponding asset. Whenever the predictions reach either level W or level C , an alarm should be issued, being stronger in the latter case. In addition, we should emphasize alarms whenever several of them occur at consecutive time periods.

Depending on the specific nature of the series considered, a suitable model will be proposed. We first describe the models used for continuous time series and, then, those for discrete series. Their specific structure is based on extensive analysis performed with actual series in our domain, such as illustrated in Section 4.

2.1 | Continuous time series

Typical continuous time series examples in our domain include device load and disk storage. For a given organization, let Z_n be the n th observation of a relevant continuous variable monitored. If h is the monitoring period, as configured by the user, Z_n represents the observation at time $n \cdot h$. D_n represents the values observed before such time and is recursively defined through $D_n = D_{n-1} \cup \{z_n\}$.

2.1.1 | Model definition

We have identified the following relevant types of continuous time series in high-frequency network traffic monitoring:

- Series with a level or linear trend, possibly varying in time.
- Series with a level or linear trend together with 1 (or more) seasonal terms, typically describing daily and/or weekly patterns.

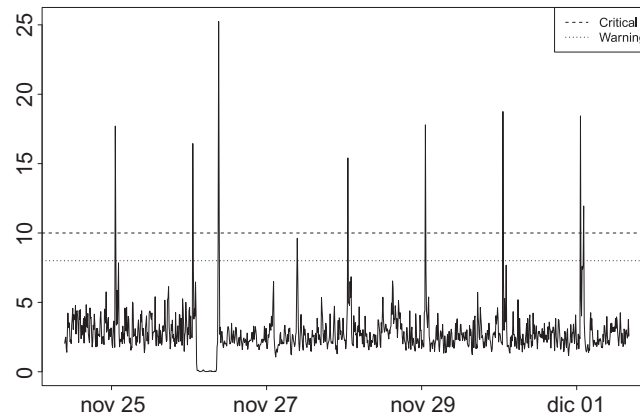


FIGURE 1 Continuous time series with linear and outburst terms

- Series with a level or linear trend together with several outburst terms, typically associated with compression or backup processes.
- Series with a level or linear trend together with 1 (or more) seasonal blocks as well as several outburst terms.

As an example, Figure 1 shows the average load of a device that goes through a backup process every night around 2:00 AM producing instantaneous load outbursts. This series would lie within the third type above.

As a consequence, the general expression that we shall adopt for our models will be

$$Z_n = Y_n + S_n + B_n + \epsilon_n,$$

where Y_n designates a linear trend block; S_n designates the seasonal block(s); B_n designates the outburst block(s); and, finally, ϵ_n designates a noise term. Note that not all three blocks will need to be included in all cases. We describe a procedure to automatically identify the required blocks in Section 2.1.2. We sketch first their definition.

The trend and seasonal terms are specifications of DLMs.¹³ We consider for them the general, normal DLM with univariate observation X_n , where X_n corresponds either to the linear trend or the seasonal term. They are characterized by the quadruple $\{F_n, G_n, V_n, W_n\}$: for each n , F_n is a known vector of dimension $m \times 1$, G_n is a known $m \times m$ matrix, V_n is a known variance, and W_n is a known $m \times m$ variance matrix. The model is then succinctly written as

$$\begin{aligned} \text{Observation: } & X_n | \theta_n \sim N(F_n' \theta_n, V_n), \\ \text{State: } & \theta_n | \theta_{n-1} \sim N(G_n \theta_{n-1}, W_n), \\ \text{Prior: } & \theta_0 | D_0 \sim N(m_0, C_0), \end{aligned}$$

where θ_n represents the state variable at time n . We specify now the general model for the required blocks, which we may combine using the superposition principle.^{13(p186-188)} The trend model is a specification of the DLM with constant F_n and G_n through

$$F = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

In turn, the basic seasonal model with period s is a specification of the DLM with constant F_n and G_n

$$F = \underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}}_{s-1}, \quad G = \underbrace{\begin{bmatrix} -1 & -1 & -1 & \dots & -1 \\ 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}}_{s-1}.$$

Typical periods that we incorporate are $s = 288$ for 5-minute data, $s = 144$ for 10-minute data and $s = 48$ for 30-minute data. For larger periods, we might use a Fourier decomposition and work with a few of the Fourier components.^{15(p102-109)} For both models, we use $V_n = 1$; however, this parameter could be assessed using, eg, maximum likelihood estimation. The variance matrix W_n is defined through the discount principle with discount factor of 0.95.^{13(p193-200)}

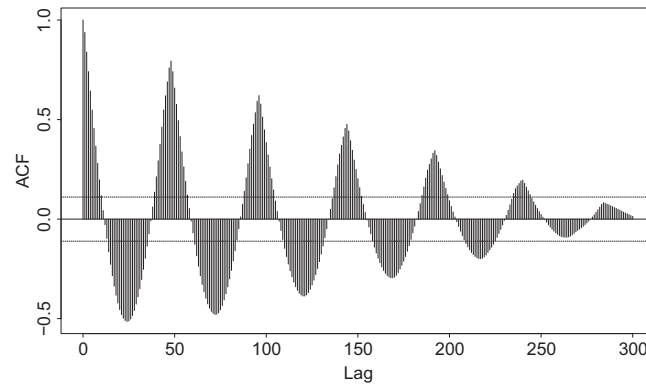


FIGURE 2 Sample autocorrelation function of a network monitoring series with seasonal component. The period is 48, corresponding to measuring the series twice every hour for a full day

In case some outburst processes are detected, then at the corresponding times, the DLM model is *turned off* and the analysis of these points is made separately. To this end we use a normal model for each particular outburst process

$$B_{n_p} \sim \mathcal{N}(\mu, \sigma^2),$$

with mean μ and variance σ^2 . Here, n_p would represent the index of the p th outburst of a given type.

2.1.2 | Model identification

By default, we use as baseline a linear trend block on top of which we add seasonal and outburst blocks when such effects seem relevant. These are identified as follows:

- **Seasonal component:** We store the lags t_i at which sign changes in the sample autocorrelation function of the series take place. We then calculate the difference between the closest nonconsecutive lags, $(t_{i+2} - t_i)$ for $i = \{1, \dots, j-2\}$, assuming that there are j changes, and compute the mean (m) and variance (s^2) of differences. If $s/m < r$, where r is an adjustable threshold, we include a seasonal component with nearest integer to m as the estimated seasonality. Figure 2 represents the autocorrelation of a series of period 48 in a specific example.
- **Outbursts:** We first search through the dataset looking for times in which the data lie outside γ standard deviations from the estimated mean of the series, with γ an adjustable threshold. For these times we record how many instances b_j of the same type of outburst occur; for example, when analyzing daily-regular outbursts, we take a certain number of days and count in how many of them does a peak appear at the same hour. After a sufficiently long time, we declare that this is an identified regular outburst process if

$$\frac{b_j}{b} > q, \quad (1)$$

where b is the number of periodical time intervals in the data used for model identification purposes, eg, the total number of days in our previous example. q is a *repetition threshold* that can be tuned to make peak selection stricter. Condition (1) suggests that peaks need to appear, at least, $q \times b$ times throughout the data timespan to be considered part of a regular outburst process.

We have also considered the use of Holder exponents to characterize the appearance of outbursts in our time series, a fairly popular approach for this problem as well.¹⁶ As they are constructed by comparing the data with an approximating (smooth) polynomial function, they could represent a general method for recognizing outbursts. However, we deemed that this was not necessary for the series we have analyzed so far. In our case, we knew that the underlying process generating our peaks was substantially different from the process that ruled the rest of the series; thus, a simple estimate as the one presented above is more than enough to tell these peaks apart from the rest of the data. Moreover, the peaks appear regularly at fixed times in each series and, consequently, they are relatively simple to identify. The analysis of these outbursts with Holder exponents could however still be performed but to do so, we would have to make sure that this approach also fulfills the requirements we have made (low running time and memory-efficient calculations), which may not be trivial since the corresponding estimates may depend on much heavier calculations.

For this identification process, we use a large enough amount of data, able to capture the main features of the time series under study. In our application domain, where daily and weekly effects are the most relevant, five weeks have been usually sufficient.

2.1.3 | Priors

It could be the case that we have available prior information for specific monitored series. This will typically entail improved performance in terms of faster fitting. However, to automate the approach, we need to be generic and have adopted specific priors as follows:

Linear term. The adopted prior is $\mathcal{N}(m_0, C_0)$, where

$$m_0 = [0 \ 0], \quad C_0 = \begin{bmatrix} 10^7 & 0 \\ 0 & 10^7 \end{bmatrix}.$$

Seasonal term. We use again a $\mathcal{N}(m_0, C_0)$ prior. s being the period, we adopt as m_0 a vector of 0's of dimension $s - 1$. The covariance matrix, of dimension $(s - 1) \times (s - 1)$, takes the form

$$C_0 = \begin{bmatrix} A & B & \dots & B \\ B & \ddots & & \vdots \\ \vdots & & \ddots & B \\ B & \dots & B & A \end{bmatrix},$$

where A is a large positive value and B is a negative value defined so that the sum of every row and column is zero. Thus, we set $B = -\frac{A}{s-2}$.

Outburst term. We use a noninformative prior

$$\Pi(\mu, \sigma^2) \propto \frac{1}{\sigma^2}.$$

2.1.4 | Model forecasting

We describe how we make forecasts with the aforementioned models. Petris et al^{15(p53-55)} provide closed forms of the one-step-ahead predictive distribution $\mathcal{N}(f_n, Q_n)$ of $X_n|D_{n-1}$, affecting both the trend and seasonal terms, and their superposition thereof. As a consequence:

- The pointwise forecast at the next period n is $E[X_n|D_{n-1}] = f_n$.
- If u_n and l_n represent, respectively, the upper and lower bounds of the expected predictive interval with probability content α , they are defined through

$$\begin{aligned} u_n &= f_n + z_{1-\alpha/2} Q_n^{1/2}, \\ l_n &= f_n - z_{1-\alpha/2} Q_n^{1/2}, \end{aligned} \quad (2)$$

where $z_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile of the standard normal distribution, being α the desired probability level of the predictive interval, which may be chosen by design, eg, depending on the asset value of the corresponding device. By default, we use $\alpha = 0.95$.

k -step ahead forecasts are also relevant in our context. If we use $a_n(k) = E[\theta_{n+k}|D_n]$, $f_n(k) = E(Y_{n+k}|D_n)$, and $Q_n(k) = \text{Var}(Y_{n+k}|D_n)$, Petris et al^{15(p70-71)} provide closed forms for such means and variances. As a consequence:

- The k -step ahead point forecast for the trend term at time n is

$$f_n(k) = a_n(0) + a_n(1)k. \quad (3)$$

- The k -step point forecast for the seasonal term at time n is

$$f_n(k) = a_n(0)_{k \bmod s}. \quad (4)$$

Prediction intervals take a form similar to (2).

For the outbursts, for simplicity, we index the peaks of a particular type with p , the number of observed outbursts of such type until the corresponding time. Then,

$$B_{p+1}|D_p \sim \mu_p + t_{p-1} \sqrt{\left(1 + \frac{1}{p}\right) \sigma_p^2},$$

where μ_p is the sample mean after observing p outbursts of the corresponding type, and σ_p^2 is the sample variance. We update recursively the parameters through

$$\mu_{p+1} = \frac{p\mu_p + X_{p+1}}{p+1}, \quad \sigma_{p+1}^2 = \frac{\sigma_p^2(p-1) + k\mu_p^2 + X_p^2 - (p+1)\mu_{p+1}^2}{p},$$

where X_p is the value of the series at the time corresponding to the p th outburst of the given type. Then,

- The *point* forecast is $E[B_{p+1}|D_p] = \mu_p$.
- The *interval* forecast is

$$\begin{aligned} u_{p+1} &= \mu_p + t_{1-\frac{\alpha}{2}, p+1} \sqrt{\left(1 + \frac{1}{p}\right) \sigma_p^2}, \\ l_{p+1} &= \mu_p - t_{1-\frac{\alpha}{2}, p+1} \sqrt{\left(1 + \frac{1}{p}\right) \sigma_p^2}, \end{aligned} \tag{5}$$

where $t_{1-(\alpha/2), p+1}$ is the $1 - \alpha/2$ quantile of the t -distribution with $p - 1$ degree-of-freedom, with α as above.

Finally, for general forecasts, we first distinguish between regular points and outbursts. For regular points, the prediction is based on the superposition of the involved DLM components. However, if the prediction refers to an outburst, the DLM model is switched off and the outburst forecast is used.

2.2 | Discrete time series

We briefly sketch the approach with this type of series. For further details see the work of Insua et al.¹⁴ We use finite, time homogeneous Markov chains $\{X_n\}$, with states $\{1, \dots, K\}$, where $K = C + c$ is the stated critical level plus a small integer c . We write the transition matrix as $\mathbf{P} = (p_{ij})$, where $p_{ij} = P(X_n = j | X_{n-1} = i)$, for $i, j \in \{1, \dots, K\}$. Should it exist, the stationary distribution π is the unique solution of $\pi = \pi\mathbf{P}$, $\pi_i \geq 0$, $\sum \pi_i = 1$.

We assimilate the monitored data D_n to observing n successive transitions of the Markov chain, say $X_1 = x_1, \dots, X_n = x_n$, given the known initial state $X_0 = x_0$. A natural prior for \mathbf{P} is defined by letting $\mathbf{p}_i = (p_{i1}, \dots, p_{iK})$ have a Dirichlet distribution $\mathbf{p}_i \sim \text{Dir}(\boldsymbol{\alpha}_i)$, where $\boldsymbol{\alpha}_i = (\alpha_{i1}, \dots, \alpha_{iK})$ for $i = 1, \dots, K$. Then, the posterior is $\mathbf{p}_i | D_n \sim \text{Dir}(\boldsymbol{\alpha}'_i)$ where $\alpha'_{ij} = \alpha_{ij} + n_{ij}$ for $i, j = 1, \dots, K$; being $n_{ij} \geq 0$ the number of observed transitions from state i to state j . Specifically, the prior adopted is given by the coefficients of the corresponding Dirichlet distributions presented in the following $K \times K$ matrix:

$$\begin{bmatrix} \alpha & \beta & \gamma & \gamma & \dots & \gamma \\ \beta & \alpha & \beta & & \dots & \gamma \\ \gamma & \beta & \ddots & & & \vdots \\ \vdots & \vdots & & & & \beta \\ \gamma & \gamma & \dots & & \beta & \alpha \end{bmatrix}.$$

In our case, because of the high frequency of the time series involved, the most-likely behavior for a certain state is to remain in its position, followed by one-state transitions. The rest of transitions are less likely. We model this behavior by setting $\alpha > \beta > \gamma$. In particular, our default values are $\alpha = 10$, $\beta = 8$, and $\gamma = 2$. As before, the proposed prior is general, flexible, and useful to deal with the cases we have found in network monitoring, facilitating a generic automated approach.

We predict the next value of the Markov chain, at time $n + 1$, using our estimates of the transition probabilities

$$P(X_{n+1} = j | D_n) = \int p_{x_n j} f(\mathbf{P} | D_n) d\mathbf{P} = \frac{\alpha_{x_n j} + n_{x_n j}}{\alpha_{x_n \bullet} + n_{x_n \bullet}} \equiv \hat{p}_{nj},$$

where $\alpha_{i \bullet} = \sum_{j=1}^K \alpha_{ij}$ and $n_{i \bullet} = \sum_{j=1}^K n_{ij}$. The pointwise prediction will then be $\sum_{j=1}^K j \cdot \hat{p}_{nj}$. Once the prediction of the next value is performed, we can also calculate the prediction interval around such estimation. If i is the last visited state and α is the probability of the one-step-ahead predictive interval, we get its upper and lower bounds u_{n+1} and l_{n+1} , using Algorithm 1. Note that discrete time series have both *maximum* and *minimum* states, here, K and 1 , respectively.

Algorithm 1 Predictive interval calculation for discrete time series

Data: Last visited state: i . Transition probabilities: $\hat{p}_{i,j}$ for $j = 1, 2, \dots, K$.

Initialization: $\text{sum} = p_{i,i}$, $u_{n+1} = i$, $l_{n+1} = i$;

while $\text{sum} < \alpha$ **do**

if $u_{n+1} = K$ **then**

 | continue, interval stops growing upwards;

else

 | $u_{n+1} = u_{n+1} + 1$;

 | $\text{sum} = \text{sum} + \hat{p}_{i,u_{n+1}}$;

end

if $l_{n+1} = 1$ **then**

 | continue, interval stops growing downwards;

else

 | $l_{n+1} = l_{n+1} - 1$;

 | $\text{sum} = \text{sum} + \hat{p}_{i,l_{n+1}}$;

end

end

Result: u_{n+1} , l_{n+1}

$k > 1$ -step-ahead predictions are more complex. For small k , we can use

$$P(X_{n+k} = j | D_n) = \int (\mathbf{P}^k)_{x_n j} f(\mathbf{P} | D_n) d\mathbf{P},$$

giving a sum of Dirichlet expectation terms. However, as k increases, the evaluation of this expression becomes computationally infeasible in our domain. Thus, we perform approximately by calculating the k th power \hat{P}^k of the matrix of expected values of the probabilities and using

$$P(X_{n+k} = j | D_n) \approx (\hat{P}^k)_{nj}.$$

Our interest also lies in the stationary distribution of the chain. For high-dimensional chains as the ones we need to deal with, we use the approximation

$$\begin{aligned} \hat{\pi} &= \hat{\pi} \hat{\mathbf{P}}, \\ \sum_i \hat{\pi}_i &= 1, \\ \hat{\pi}_i &\geq 0. \end{aligned} \tag{6}$$

Once this equation is solved, we can use $\hat{\pi}_i \geq 0$, the approximate stationary distribution, to produce long-term forecasts.

2.3 | General scheme

The general strategy followed for each time series is described in Figure 3 (note that this whole procedure is applied in an automated fashion to each new time series inputted to the system). Once the series is in, a first distinction is made between whether it is continuous or discrete. In this last case, the system uses the Markov model in Section 2.2. Otherwise, the model identification process presented in Section 2.1.2 is applied. Once the blocks have been identified, the appropriate dynamic model in Section 2.1.1 is used. After fitting the corresponding models based on a sufficient amount of data,

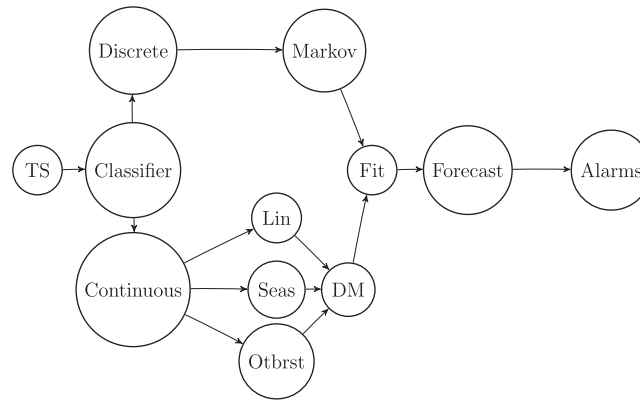


FIGURE 3 General scheme

forecasts may be made and alarms raised in case an anomalous behavior is detected or reaching critical values is predicted, both in the short or medium terms.

3 | IMPLEMENTATION

We now describe the implementation and use of our approach. We present here issues in relation with having to deal with several hundred thousands of monitored series sampled over periods ranging from 1 to 30 min, depending on the criticality of the corresponding device. We thus cover the aforementioned speed and space scalability requirements.

The implementation has been carried out through the creation of an object-oriented python package linked with the core monitoring system. Each monitored time series will be associated with an object that includes all methods required to accomplish predictive monitoring tasks. The package is structured as in Figure 4. The main code blocks are the classes *gen* and *classifier*. The main code receives the data and timestamps of the training period of each series and generates a *classifier* object. Its attributes contain the characteristics needed to identify the class of models to be used in the analysis of that particular time series. Once this object is constructed, using it together with the data and timestamps, a *gen* object is created. This is the general class whose methods depend on the model relevant for the given time series, which will ultimately be used to undertake the forecasting, anomaly, and critical value detection tasks. Among these, we highlight the *update* method, which incorporates new data to the current model and updates the parameters in a Bayesian manner, and the *predict* method, which performs *k*-step ahead predictions, returning point, and interval forecasts, providing the basis to construct the specific critical and anomaly detection functions that raise the alarms.

Short-term forecasts are needed for safety and security purposes. Given D_n , we produce forecasts using the *predict* method for future observations $x_{n+1}, x_{n+2}, x_{n+3}, \dots$ up to a certain time $(n + k) \cdot h$, with *k* defined by the user. $k = 3$ is the default value. Forecasts are delivered as predictive intervals $[l_{n+i}, u_{n+i}]$ with probability level configurable by the user.

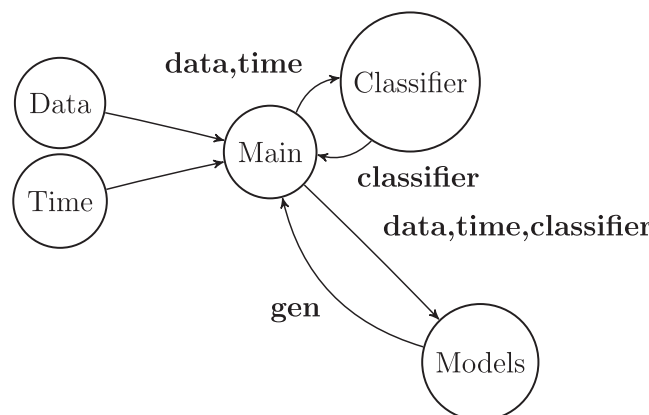


FIGURE 4 Structure of the python package for time series monitoring and anomaly detection

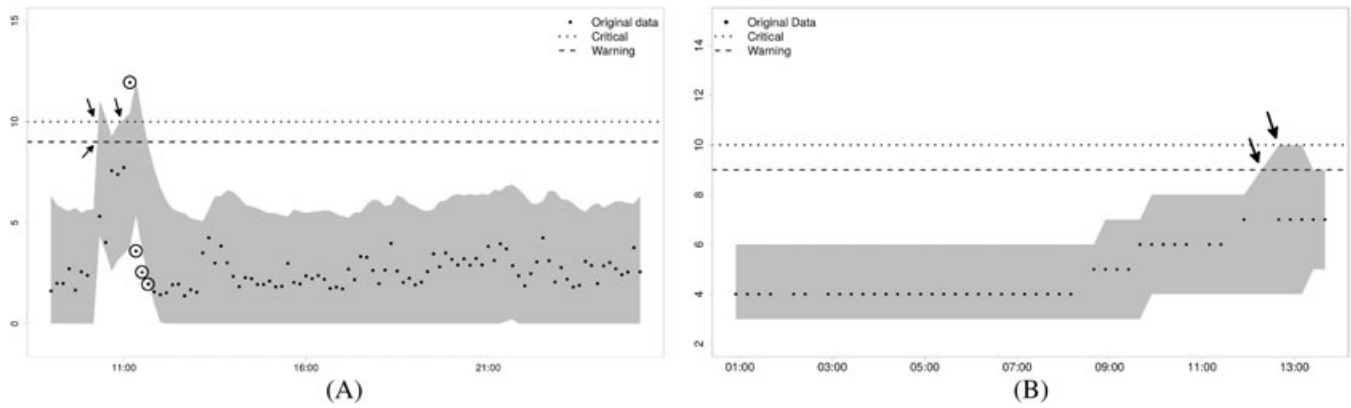


FIGURE 5 Short-term forecasting using 95% one-step-ahead predictive intervals. A, Continuous time series; B, Discrete time series

These could be used to detect aberrant behavior within the time series: when the next observation x_{n+1} does not lie in the predictive interval $[l_{n+1}, u_{n+1}]$, a warning concerning unexpected behavior is displayed pointing to a potential security issue. Our implementation modulates the warning depending on the number of consecutive intervals in which it needs to be launched. Therefore, as to control for false positives, alarms could be issued just when the number of violations exceeds certain threshold within a moving window with fixed number of time steps, as suggested in the work of Brutlag.⁷ Short-term predictions also serve for safety monitoring tasks. Specifically, when W or C lie within a predictive interval $[l_{n+j}, u_{n+j}]$ for $j \in \{1, 2, \dots, k\}$, an alarm pointing to potentially high values of x should be issued. The higher the number of intervals covering the particular level, the more intense would be the alarm. We illustrate both functionalities through a practical example in Figure 5A. In this case, an intense unexpected behavior alarm would be issued around 2:00 AM as there are four measurements outside the corresponding predictive intervals. In addition, an alarm concerning a potentially high value of the series would be issued at 1:40 AM, as the predictive intervals exceed the warning and critical levels at 1:50 AM and 2:00 AM, respectively. A similar example using discrete time series is displayed in Figure 5B.

Long-term forecasts are used to accomplish safety monitoring tasks: we try to ascertain whether critical levels will be reached with sufficiently high probability in the long term. In principle, it could be done using predictive intervals as we do with short-term predictions and illustrated in Section 2.1.4. However, since the system must monitor so many high-frequency time series in real time and, consequently, must perform under a very narrow time window, we need to make a compromise: although the calculation of predictive intervals is doable as in Section 2.1.4, here, we will just focus on point forecasts. This allows for a drastic improvement in the computational costs of running the involved algorithms, as we can use explicit expressions to make the forecasts obviating the costly computation of predictive variances. We use as point forecasts z_{n+j} , the midpoint of intervals $[l_{n+j}, u_{n+j}]$, and try to find out the first j_1 such that $z_{n+j_1} > W$ and the first j_2 such that $z_{n+j_2} > C$. This allows us to identify, well in advance, time instants, in which critical values might be reached, as we may provide explicit expressions of such inequalities. Indeed, in the linear case, from (3), we try to find the first $j_i, i = 1, 2$, such that

$$j_1 > \frac{W - a_n(0)}{a_n(1)}, \quad j_2 > \frac{C - a_n(0)}{a_n(1)},$$

assuming that $a_n(1) > 0$. Similarly, with a seasonal block, from (4), we try to find the first $j_i, i = 1, 2$ such that

$$a_n(0)_{j_1 \bmod s} > W, \quad a_n(0)_{j_2 \bmod s} > C,$$

if any. Finally, when the model contains linear and seasonal blocks, we compute j_1 (and similarly j_2) through

```

while  $a_n(0) + a_n(1)k + a'_n(0)_{k \bmod s} < W$  do
  |  $k = k + 1$ ;
  |  $j_1 = k$ ;
end

```

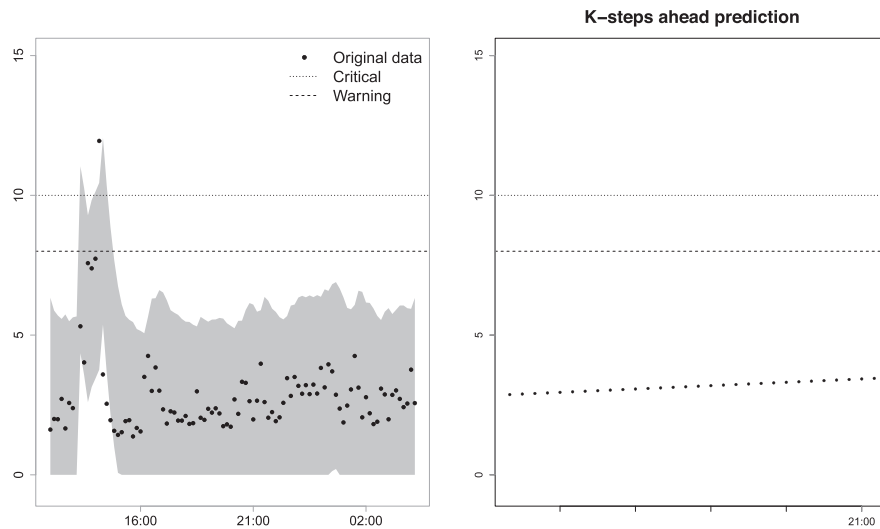


FIGURE 6 Long-term forecast for continuous time series

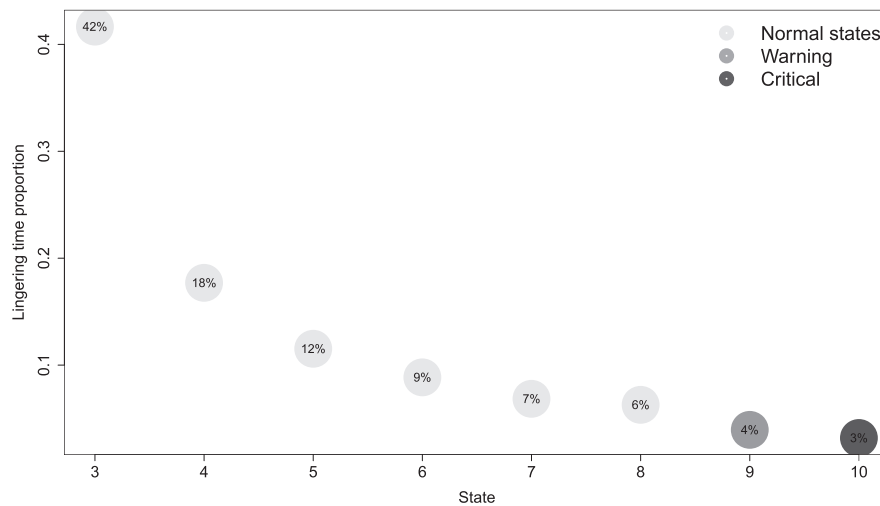


FIGURE 7 Long-term forecast for discrete time series

where $a'_n(0)$ are the parameters of the pointwise k -step ahead forecasts for the seasonal term, and $a_n(0)$ and $a_n(1)$ are those of the linear trend.

A practical example of this type of forecast is illustrated in Figure 6. There, the time series is not expected to cross neither warning nor critical levels in the next five hours. However, if the linear growth in the main trend continues, both levels could possibly be reached in the future. To control for false positives, we could define a *relevance time window*, only raising an alarm when anomalous behavior occurs repeatedly inside it and neglecting them otherwise.

For long-term forecasting with discrete time series, we use the stationary distribution in (6). If the sum of probabilities assigned to the states that lie above the warning (critical) level is higher than a certain preestablished threshold, an alarm would be raised. This is illustrated in Figure 7. In this case, an alarm should be raised if the warning (critical) threshold is at 0.07 or lower.

We assessed the time performance of our approach with stress tests. For different models, we have sequentially added 15 000 new data points through the *update* method, performing a short-term prediction (30 min) and a long-term prediction (5 hours) at each iteration. In particular, we tested four models: one with a linear trend; one with a linear trend and a seasonal block with period 144; a model with a linear trend and an outburst block; and, finally, a Markov chain model with 50 states. In Table 1, we show the mean, median, min, and max times of the whole operation (update + short-term prediction + long-term prediction) for each of the models used. Time calculations have been performed in an Intel Core i7-3630UM, 2.40GHz × 8 machine.

TABLE 1 Mean, median, min, and max times in seconds of update, short-term and long-term predictions for different models

	Mean	Median	Min	Max
Linear	$4.03 \cdot 10^{-4}$	$3.97 \cdot 10^{-4}$	$3.76 \cdot 10^{-4}$	$9.12 \cdot 10^{-4}$
Linear + Seasonal	$2.50 \cdot 10^{-2}$	$2.49 \cdot 10^{-2}$	$2.43 \cdot 10^{-2}$	$4.60 \cdot 10^{-2}$
Linear + Outburst	$4.42 \cdot 10^{-4}$	$4.35 \cdot 10^{-4}$	$2.73 \cdot 10^{-4}$	$8.38 \cdot 10^{-4}$
Markov Chain	$8.81 \cdot 10^{-4}$	$8.06 \cdot 10^{-4}$	$7.70 \cdot 10^{-4}$	$1.96 \cdot 10^{-3}$

The algorithm is fast enough, and consequently, able to cope with the typical very high-frequency data in the network monitoring domain. Note though that the model including the seasonal trend is remarkably slower than the rest of the models. This is to be expected since it includes a seasonal term with long period, which involves performing several operations with high-dimensional matrices. Should better performance be required, we could use a Fourier decomposition of the seasonal trend, and work with a few of the most relevant Fourier components,^{15(p102–109)} although this would require a method to automate identification of such components.

In terms of memory, the whole approach is constructed so that a fixed number of parameters are stored at any step of the calculations to further improve its performance. In the case of continuous time series, the linear part of the DLM stores two 2-component vectors, the vector F , and the mean of the state distribution m ; three 2×2 matrices, the matrix G , the covariance matrix C of the state distribution, and the system covariance; and a parameter corresponding to the observation variance. Similarly, the seasonal part stores two $(s - 1)$ -component vectors, three $(s - 1) \times (s - 1)$ matrices, and one parameter. The outburst term holds only two parameters corresponding to the mean and variance of the corresponding distribution, for each recurrent peak detected. Finally, the Markov model for discrete time series employs a $k \times k$ matrix that is updated at each step, where k corresponds to the number of states considered in the chain. This all means that in the worst case scenario, the model needs to store in memory $\mathcal{O}(s^2)$ parameters for the continuous case and $\mathcal{O}(k^2)$ for the discrete case, which is feasible within the architecture developed.

These remarks, together with the time performance measures previously presented, suggest that the algorithm fulfills the time and space scalability requirements.

4 | EMPIRICAL TEST FOR ACCURACY

This section provides empirical support for the accuracy of the proposed framework for network monitoring, fulfilling our fourth requirement. It also illustrates how the approach adopted was designed based on modeling numerous series in our domain to obtain the relevant features.

To that end, we use as benchmark the four time series plotted in Figure 8, which are representative of the aforementioned characteristics in network traffic monitoring (linear trends, seasonal behaviors, outbursts, and discrete processes). In this way, we show that our automated framework meets the aforementioned properties of scalability and versatility, being able to deal with time series of very different nature.

To assess the overall accuracy of our approach, we use some of the traditional performance metrics for point forecasts. In particular, we shall use the mean absolute error, the mean square error, and the mean absolute percentage error. We compare our modeling framework with some of the traditional methodologies mentioned in the introduction, specifically ARIMA and exponential smoothing (ES), as there are open source automated implementations available. Specifically, we shall use the *forecast* R package.^{17,18} In our case, for the comparison, it is important to use monitoring methods that are able to deal in a completely automated fashion with large amounts of different time series.

Note first that our forecast, in contrast to these provided by the other methods, is a full predictive distribution rather than a single-point forecast. Then, if we want to compute the aforementioned performance metrics, we need to summarize our predictive distribution with a single point. To that end, we shall choose the point that minimizes the corresponding expected loss under the predictive distribution.*

Figure 9 plots the different performance metrics versus the forecast horizon from 1- to 10-step-ahead for the first three time series in Figure 8. Notice that the accuracy of our method is deemed competitive. In particular, our approach notably

*It is well known that the predictive median is optimal under absolute loss; the mean, under squared loss, and the (-1) -median under the absolute percentage loss, that is the median of the p.d.f. $g(y) \propto p(y)/y$, where $p(y)$ is the forecast distribution.

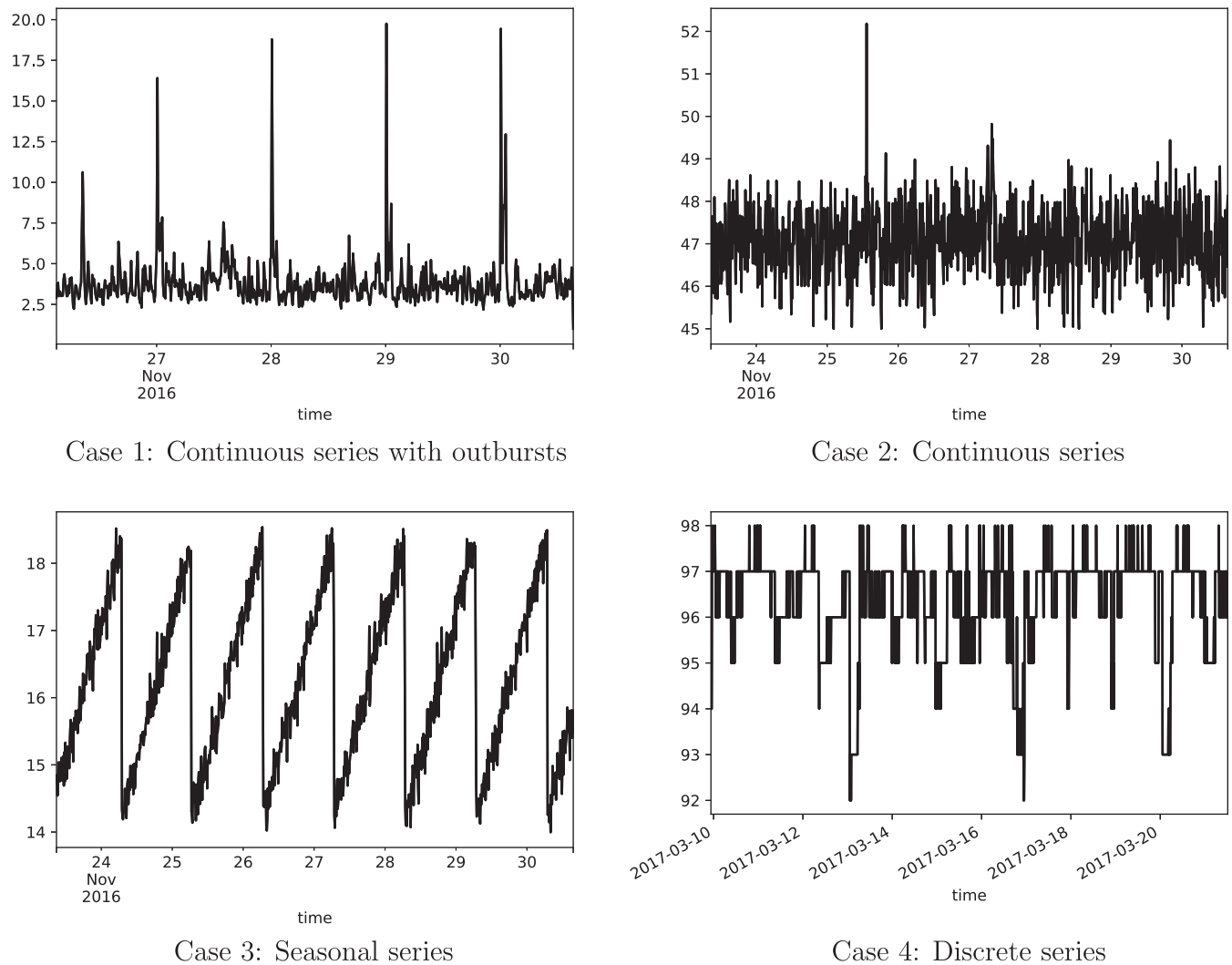


FIGURE 8 Time series used for the comparisons

outperforms ARIMA and exponential smoothing in cases 1 and 3. In the first case, this is because of the fact that we are capable of accounting for the outburst process in such sample. Modeling these regular outbursts separately makes all the difference here. For sample 2, all methods have a similar performance in all three metrics, due to its simpler nature. However, for sample 3, the automated approach implemented in the *forecast* R package for ARIMA and exponential smoothing was not capable of recognizing the seasonal component in such data. Therefore, the difference in performance is because of the fact that our approach is correctly identifying the seasonal component and the error remains more stable through the number of steps ahead of the predictions.

To further study sample 3 results, we also computed these performance metrics for ARIMA and ES, tuning them manually to explicitly account for the corresponding seasonal behavior. Results are presented in Figure 10. Here, we see that once ARIMA and ES are given this information, their performance in the long-term predictions improves sharply. However, even in this case our approach, still being completely automatic, remains competitive. Moreover, if we also consider the scalability requirement, both ES and ARIMA have much higher running times, being especially high in the case of ARIMA. Therefore, even though we had to intervene the ARIMA and ES models to correct them and gave them much more running time to produce the forecasts, our method still performs well enough while fulfilling the requirements of automaticity and scalability.

Finally, we also calculated the performance metrics for the fourth sample in Figure 11. However, neither ARIMA nor ES are prepared to deal with discrete time series. Therefore, here, we only present the results for our method. As expected, our method performs better in short-term predictions than in the long-term ones. As mentioned, when dealing with long-term forecasts in the discrete case, it may be more sensible to work with the stationary distribution described in (6).

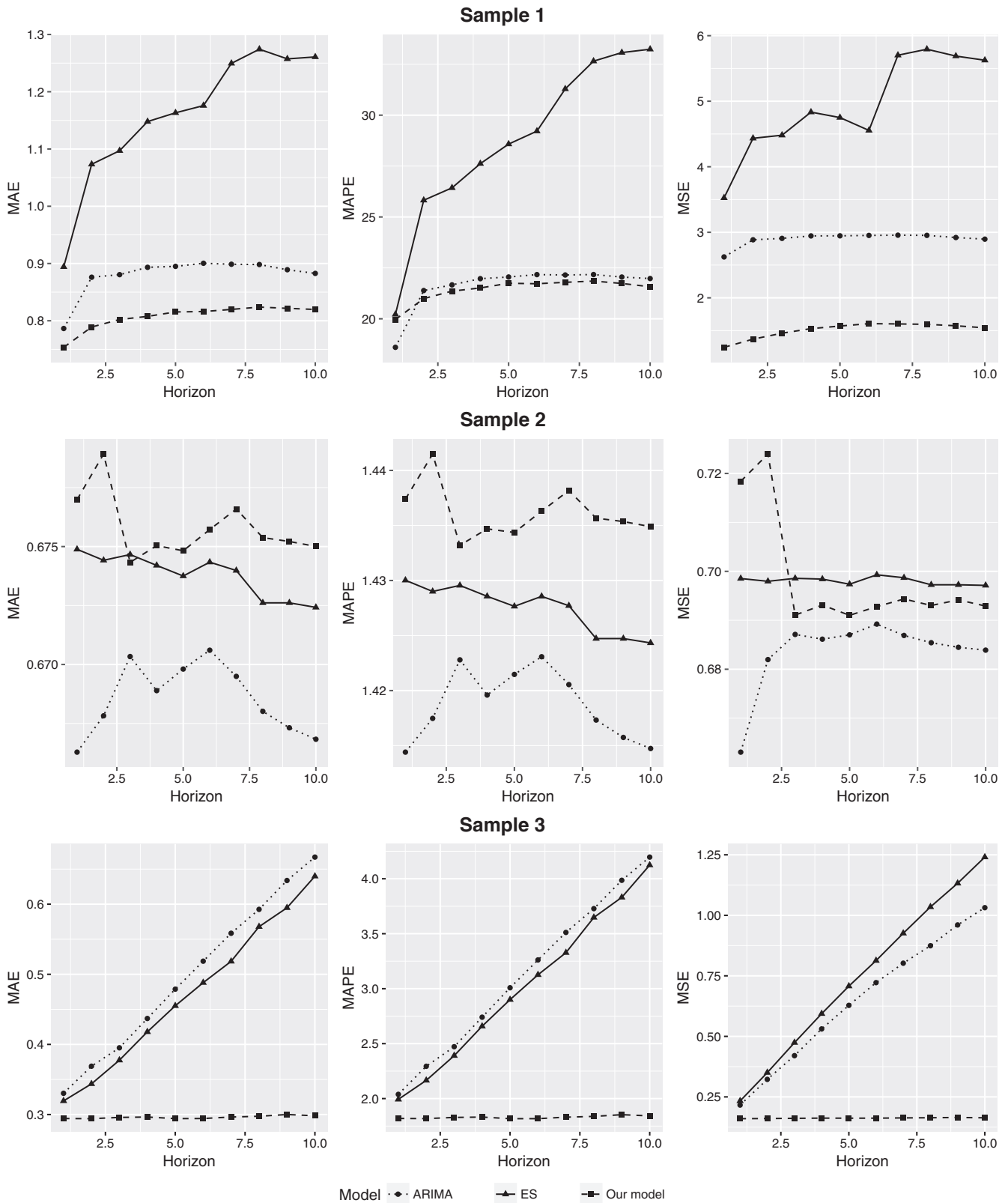


FIGURE 9 Point forecast performance comparison for the continuous time series. ARIMA, autoregressive integrated moving average; ES, exponential smoothing; MAE, mean absolute error; MAPE, mean absolute percentage error; MSE, mean square error

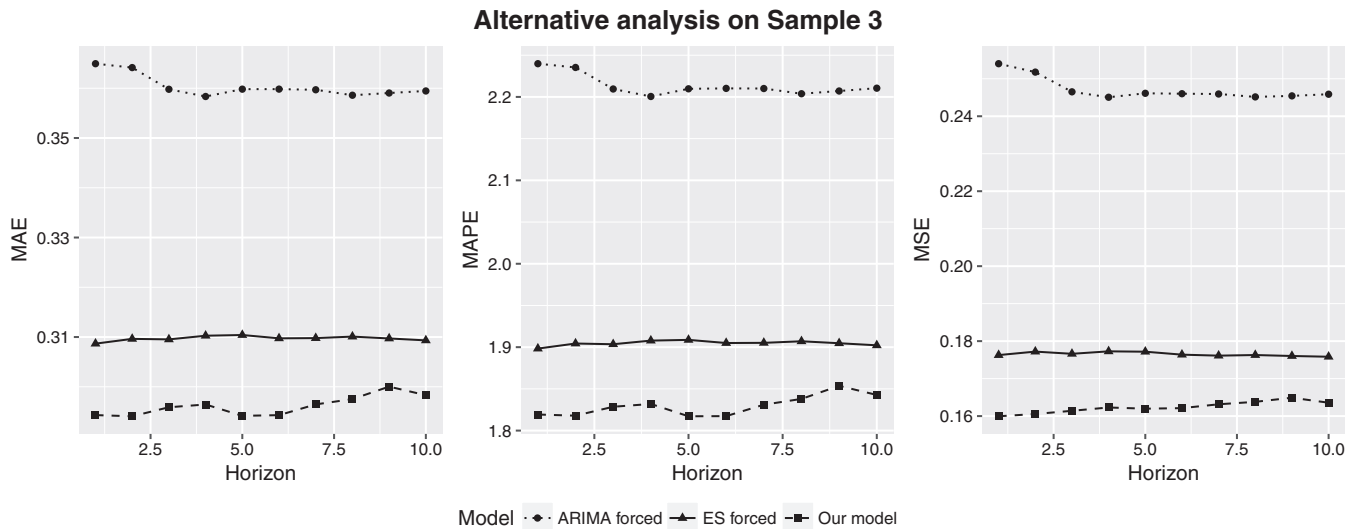


FIGURE 10 Point forecast performance comparison for sample 3 using *hand-crafted* features for autoregressive integrated moving average (ARIMA) and exponential smoothing (ES) models versus the automatic fitting of our approach (forcing seasonality and giving the period of this seasonal component). MAE, mean absolute error; MAPE, mean absolute percentage error; MSE, mean square error

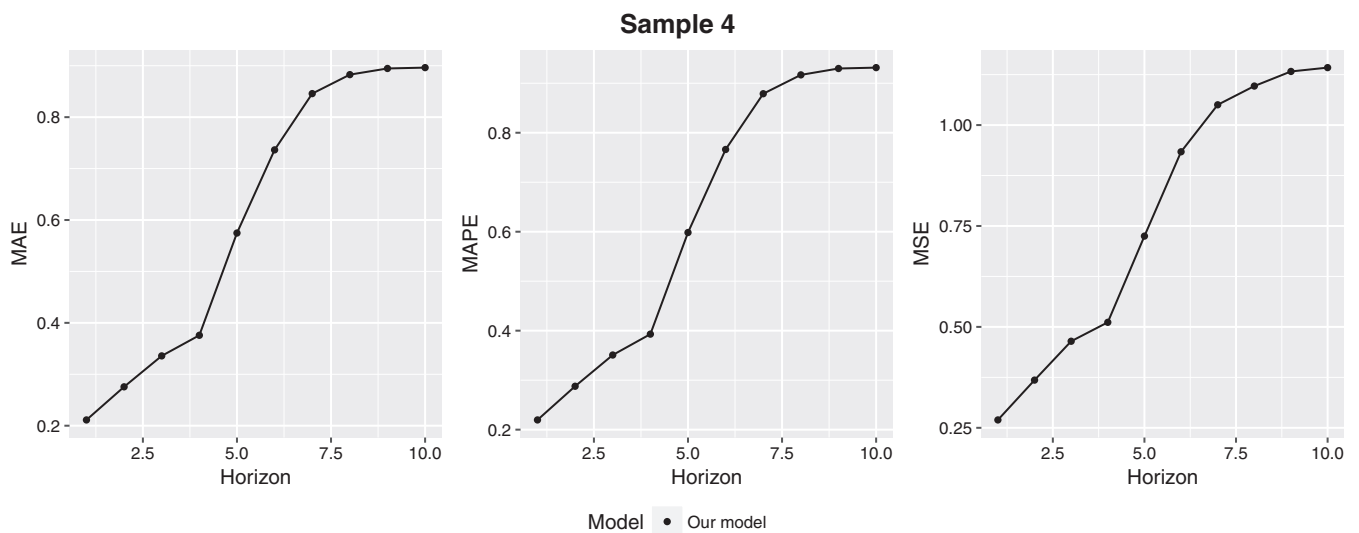


FIGURE 11 Point forecast performance comparison for the discrete time series. MAE, mean absolute error; MAPE, mean absolute percentage error; MSE, mean square error

A key aspect that differentiates our methodology from the rest is that, it produces full predictive distributions, with reasonable uncertainty quantification, rather than point forecasts. Such distributions play an important role in anomaly detection as pointed out in Section 2. To perform probabilistic forecast evaluations, we plot in Figure 12 the empirical coverage against the width of the predictive distributions, for 1-, 5-, and 10-step-ahead predictions in each of the cases studied. As can be seen, our approach is working properly in the continuous case as the coverage curve is very close to the 45-degree line (ideal coverage). In the discrete time series case, we observe overcoverage especially for 5- and 10-step-ahead forecasts. This is a consequence of how the credible intervals are constructed in such case (Algorithm 1). If we want to obtain an α probability predictive interval, we will keep on adding states to the interval until the probability of the system staying in any of those states is at least α . However, we have point probability masses because of the discrete nature of the series and, therefore, the addition of a discrete number of these will make the real probability of the predictive interval bigger than α since we only stop adding states once α is surpassed to make sure that we cover *at least* such level.

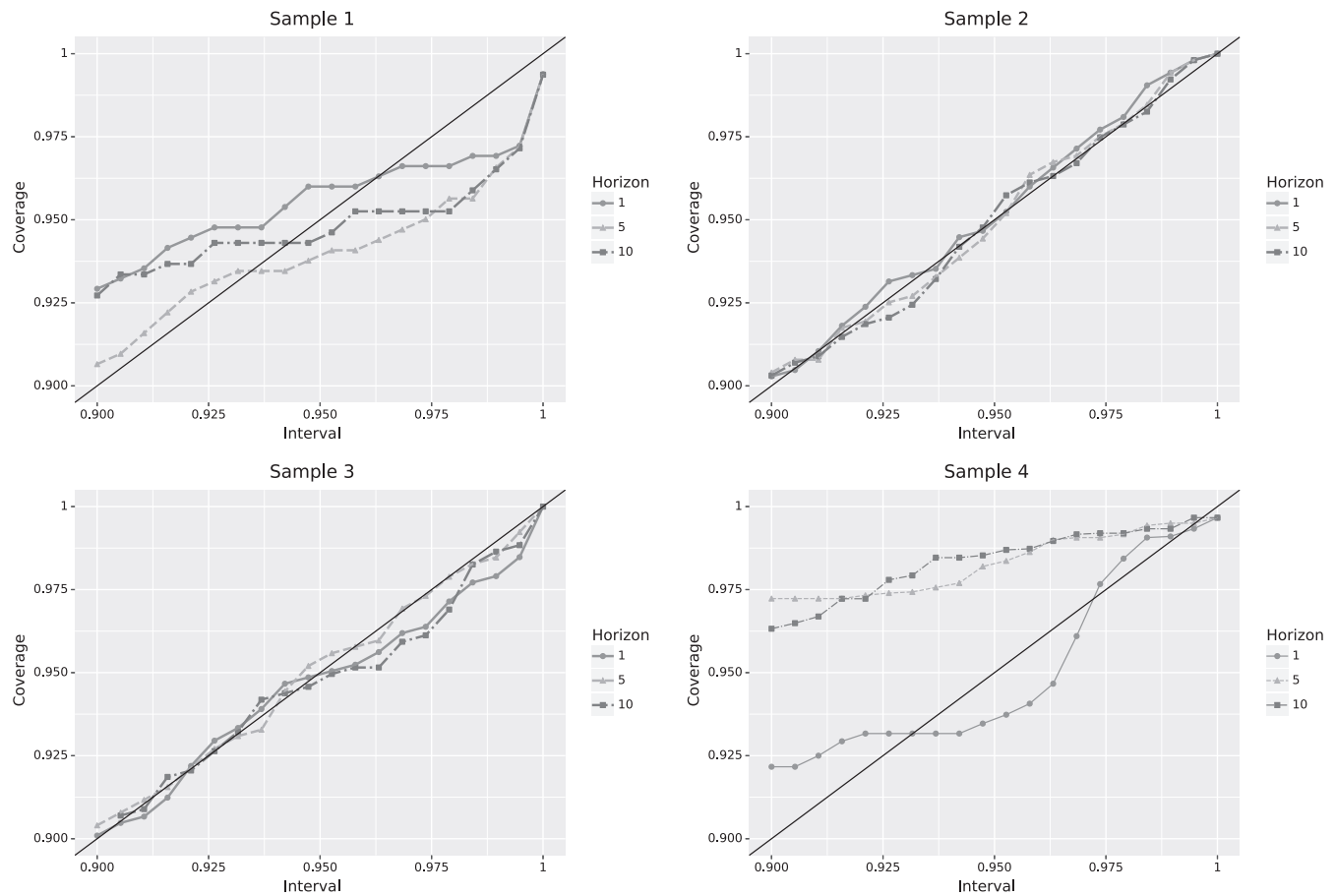


FIGURE 12 Empirical coverage for 1-, 5-, and 10-step-ahead predictive distributions [Colour figure can be viewed at wileyonlinelibrary.com]

5 | DISCUSSION

Based on the features typical of network monitoring time series, we have provided a framework to identify safety and security issues within a large number of ICDs. The framework has been implemented and operates as part of a system controlling a network with more than three hundred thousand devices, even taking into account that each of them provides several very high-frequency time series.

The framework presented is very flexible when it comes to identify and model different behaviors that can be described in terms of basic components, as it happens in our application domain. This would allow for wide use through different environments beyond it, as we may use the same approach to monitor very distinct time series with intrinsic varying nature. Moreover, since the procedure just needs to store a few parameters for each time series, we can say that it is scalable, both in terms of memory and runtime. The framework is amenable of parallel processing, making possible to monitor different batches of series in different cores, reinforcing this way the scalability request. In addition, thanks to our model identification procedure; our approach is able to work automatically without human supervision. Another interesting advantage refers to updating series with missing data, which happens relatively frequently in our domain. In presence of such missing values, the state carries no information and, therefore, the filtering distribution at such time is just the one-step-ahead predictive distribution of the previous step.

Improvements in the algorithm can still be performed, especially if we take into account specifics of the cases monitored. To this extent, using the eventual hierarchy between monitored devices, model identification could be optimized, for example, allowing for a faster regular outburst detection. Moreover, should there exist correlated time series in the database, such correlations could be used similarly to our advantage. For this, we would need coupling-decoupling strategies of the type described in, eg, the work of Berry and West.¹⁹ It is also possible to carry out an automatic performance evaluation of the algorithm. This could be done by computing the autocorrelation function of the residuals within a

certain time window. If, for instance, a strong correlation in the first few lags is encountered, an autoregressive term could be added to the model. In other cases, when model performance deteriorates too much, an alarm could be issued, finally demanding the intervention of a human analyst. It may be also interesting to adjust in real time the width of the predictive probability intervals taking into account the particular potential economical losses of false positives and negatives of the system. From this point of view, we may also find interesting to rebuild the structure of the algorithm to try to produce long-term forecasts including the probability intervals as well, moving beyond the point-wise predictions here proposed for practical implementation.

ACKNOWLEDGEMENTS

Roi Naveiro acknowledges the Spanish Ministry of Education for the FPU 15/03636 PhD scholarship. Simón Rodríguez acknowledges the Spanish Ministry of Science for the FPI SEV-2015-0554-16-4 PhD scholarship. The work of David Ríos Insua is supported by the Spanish Ministry of Science programs RTC-2017-6593-7 and MTM2017-86875-C3-1-R, and the AXA-ICMAT Chair on Adversarial Risk Analysis. We are very grateful for the comments and suggestions of the referees.

ORCID

Roi Naveiro  <https://orcid.org/0000-0001-9032-2465>

REFERENCES

1. Mortenson MJ, Doherty NF, Robinson S. Operational research from Taylorism to Terabytes: a research agenda for the analytics age. *Eur J Oper Res*. 2015;241(3):583-595.
2. Dimelis SP, Papaioannou SK. ICT growth effects at the industry level: a comparison between the US and the EU. *Inf Econ Policy*. 2011;23(1):37-50.
3. The Geneva Association. *Ten Key Questions on Cyber Risk and Cyber Risk Insurance*. Zurich, Switzerland: The Geneva Association; 2016. Technical Report.
4. World Economic Forum. *The Global Risks Report*. Cologne, Switzerland: World Economic Forum; 2018. Technical Report.
5. McAfee, CSIS. *Net Losses: Estimating the Global Cost of Cybercrime*. McAfee Intel Security; 2014. Technical Report.
6. Sedgewick A. Framework for improving critical infrastructure cybersecurity. Gaithersburg, MD: NIST; 2014. Technical Report.
7. Brutlag JD. Aberrant behavior detection in time series for network monitoring. Paper presented at: 14th Systems Administration Conference (LISA); 2000; New Orleans, LA.
8. Taylor SJ, Letham B. Forecasting at scale. *Am Stat*. 2018;72(1):37-45.
9. Vallis O, Hochenbaum J, Kejariwal A. A novel technique for long-term anomaly detection in the cloud. In: Proceedings of the 6th USENIX conference on Hot Topics in Cloud Computing (HotCloud); 2014; Philadelphia, PA.
10. Breiman L, Friedman J, Stone C, Olshen R. *Classification and Regression Trees*. Boca Raton, FL: Taylor & Francis Group; 1984.
11. Bianco AM, García Ben M, Martínez EJ, Yohai VJ. Outlier detection in regression models with ARIMA errors using robust estimates. *J Forecast*. 2001;20(8):565-579.
12. Malhotra P, Vig L, Shroff G, Agarwal P. Long short term memory networks for anomaly detection in time series. In: *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. Louvain-la-Neuve, Belgium: Presses universitaires de Louvain; 2015:89-94.
13. West M, Harrison J. *Bayesian Forecasting and Dynamic Models*. New York, NY: Springer-Verlag New York; 1999.
14. Insua DR, Ruggeri F, Wiper M. *Bayesian Analysis of Stochastic Process Models*. Chichester, UK: John Wiley & Sons; 2012.
15. Petris G, Petrone S, Campagnoli P. *Dynamic Linear Models with R (Use R!)*. New York, NY: Springer-Verlag; 2009.
16. Sohn H, Robertson AN, Farrar CR. *Singularity Detection Using Holder Exponent*. Los Alamos National Laboratory; 2002. Technical Report.
17. Hyndman RJ, Khandakar Y. Automatic time series forecasting: the forecast package for R. *J Stat Softw*. 2008;27(3):1-22.
18. Hyndman R, Athanasopoulos G, Bergmeir C, et al. forecast: forecasting functions for time series and linear models. R package version 8.4. 2018.
19. Berry L, West M. Bayesian forecasting of many count-valued time series. arXiv preprint arXiv:1805.05232. 2018.

How to cite this article: Naveiro R, Rodríguez S, Ríos Insua D. Large-scale automated forecasting for network safety and security monitoring. *Appl Stochastic Models Bus Ind*. 2019;35:431-447. <https://doi.org/10.1002/asmb.2436>