Received: 10 January 2023 | Revised: 21 July 2023 | Accepted: 18 August 2023

DOI: 10.1002/qre.3436

SPECIAL ISSUE ARTICLE

WILEY

Comparison of threshold tuning methods for predictive monitoring

Paulina von Stackelberg | **Rob Goedhart** | **Ş. İlker Birbil** | **Ronald J. M. M. Does**

Amsterdam Business School, Business Analytics, University of Amsterdam, Netherlands

Correspondence
Paulina von Stackelberg, Amsterdam Business School, Business Analytics, University of Amsterdam, Netherlands.
Email: p.b.vonstackelberg@uva.nl

Abstract

Predictive monitoring techniques produce signals in case of a high predicted probability of an undesirable event, such as mortality, heart attacks, or machine failure. When using these predicted probabilities to classify the unknown outcome, a decision threshold needs to be chosen in statistical and machine learning models. In many cases, this is set to 0.5 by default. However, this may not lead to an acceptable model performance. To mitigate this issue, different threshold optimization approaches have been proposed in the literature. In this paper, we compare existing thresholding techniques to achieve a desired false alarm rate, and also evaluate the corresponding precision and recall performance metrics. A simulation study is conducted and a real-world example on a medical dataset is provided.

KEYWORDS

false alarm rate, predictive monitoring, threshold tuning

1 | INTRODUCTION

Predictive Process Monitoring (PPM) is aimed at detecting potential problems in processes before they occur. This involves modeling the outcome using statistical or Machine Learning (ML) models, which allow prediction of high risk situations across domains such as medicine[1,2] and manufacturing.[3–5] With the surge of available data, a wide variety of models have been proposed in the literature, including logistic regression, random forests, and neural networks.[6] Yet, when it comes to the practical value and performance of any PPM method, post-modeling decisions also need to be considered. One such choice is the probability threshold: after fitting the model and obtaining a vector of probabilities, a threshold has to be chosen to make a final classification. For binary outcomes, researchers and practitioners often rely on a threshold of 0.5. However, this may not be an optimal choice, since data characteristics such as class imbalance can skew the probabilities in a particular direction and therefore overpredict the majority class,[7,8] rendering a cutoff of 0.5 unsuitable.[9] While approaches involving resampling algorithms such as SMOTE[10] have been developed to address that challenge, their use may lead to issues such as poorly calibrated estimates.[11] Furthermore, certain applications require practitioners to limit the False Alarm Rate (FAR) to prevent alarm fatigue[12] as well as accommodate constrained resources. Therefore, there is a demand for methods that can be adapted to practitioner needs while being computationally efficient. To address this, different threshold tuning techniques have been proposed in the literature.

For instance, Huberts et al.[13] applied logistic hierarchical regression and XGBoost in a mental health setting and proposed a data-driven tuning procedure to achieve a desired FAR. Being based on a split-sample approach, this method however comes with a drawback: the model needs to be re-trained several times, and therefore this technique comes with a high computational cost. Alternative methods have been proposed that do not have this requirement. Two examples are the GHOST method[14] and Bootstrap approach.[15,16] Both of these techniques use the predicted probabilities in the training set to adjust the classification threshold, and therefore do not involve any re-fitting of the model in the thresholding procedure. Along this line of reasoning, Albers and Kallenberg[17] proposed a method that can be used to tune the probability threshold based on nonparametric theory of order statistics (OS). Due to not requiring any resampling or grid-search, this method[17] has a very low computational demand, and has been theoretically studied. However, to the best of our knowledge, its performance has thus far not been empirically contrasted with alternatives in a prediction setting.

In the current study, we compare these four existing techniques with a focus on two aspects: (1) how well a given method approximates a pre-set FAR, and (2) what the performances of the different techniques are in terms of precision and recall values. The thresholding approaches are tested in a simulation study and illustrated on a real-world medical dataset of ICU patients. This paper is structured as follows: first, we introduce the methods that we consider in our comparison (Section 2). Next, we describe the medical dataset as well as the mechanisms underlying the simulated datasets (Section 3). Finally, we discuss results (Section 4) as well as potential avenues for future research (Section 5).

2 | THRESHOLDING METHODS

We compare four different threshold tuning methods, which we describe in this section. Our goal is to use these methods to obtain thresholds that approximate a desired, pre-set FAR on the test set as closely as possible. The outcome in the data is binary, with $y = 0$ describing controls and $y = 1$ describing cases. Let FAR_{pre} denote the user-specified FAR, and FAR_{test} denote the FAR calculated on the test set when the obtained threshold is used to convert the predicted probabilities to classes.

The general set-up the four methods share in this study is as follows: (i) We split the dataset into a training (70%) and test set (30%). We use stratified sampling to preserve the class distribution. (ii) Next, we fit the model (e.g., logistic regression) on the training set. In the first three methods (i.e., 1. Order Statistics, 2. Bootstrap, 3. GHOST), this model is used for the thresholding procedure as well as to predict instances on the test set. In the last method (i.e., 4. Split Sample), the model fitted on the entire training set is not used for threshold tuning, but only for calculating the predictions on the test set: in this case, the entire training set is further split into an internal training and validation set for threshold tuning (see Section 2.4). (iii) Using one of the four methods, the threshold is tuned on the training set. (iv) Lastly, we use the model fitted on the entire training set to calculate probabilities on the test set. Then, we apply the estimated threshold to convert these probabilities to classes.

First, we introduce the first three methods mentioned above: OS, Bootstrap, and GHOST. Steps (i) and (ii) are applied first. Next, the model fitted on the training set is used to obtain the probabilities (denoted as p_i) on the training set for $i = 1, \dots, n_{train}$, where n_{train} denotes the sample size of the training set. Then, the sequence of steps for those three methods is as follows:

2.1 | Order statistics

The first method uses OS to obtain the threshold and is based on nonparametric theory.[17] In their paper, the authors introduced the Bias Criterion (BC) and the Exceedance Probability Criterion (EPC) for the design of univariate control charts. In this paper we focus on the BC, which aims to provide a pre-specified FAR in expectation. This can be implemented using the following procedure:

1. Order the probabilities as $p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(n_{train})}$ with $p_{(1)}$ the minimum and $p_{(n_{train})}$ the maximum OS of the estimated probabilities, respectively. Note that in this approach, only the in-control instances (with sample size $n_{IC}; y = 0$) in the training set are used to compute the threshold.
2. Calculate the threshold:

$$t_{OS} = (1 - \lambda)p_{(n_{IC}+1-k)} + \lambda p_{(n_{IC}-k)}, \tag{1}$$

where $\lambda = FAR_{pre} + \delta$, $\delta = n_{IC}FAR_{pre} - k$, and $k = \lfloor n_{IC}FAR_{pre} \rfloor$.

Note that in the version used in this study, weighing is used instead of random switches as suggested by Albers and Kallenberg.[17] This ensures that the threshold is a fixed constant for each sample, rather than a random variable. For more details on the derivation, we refer to Appendix A.

2.2 | Bootstrap

Another approach we examine in this study is based on a bootstrap approach (BST) to obtain a limit for the predicted probabilities.[15,16]

1. Draw B bootstrap samples of the predicted probabilities from the training set.
2. For each bootstrap sample, calculate the $100 \times (1 - FAR_{pre})th$ percentile.
3. Let the threshold t_{BST} then be the average of the $100 \times (1 - FAR_{pre})th$ percentiles in the B samples.

We set the number of bootstrap samples to B=1000. Note that while the model is trained on the entire training set, the control limits are obtained only using the in-control instances ($y = 0$) of the training set.[16]

2.3 | GHOST

This technique is based on the algorithm proposed by Esposito et al.[14] and altered to fit the current study. We define a fixed vector of thresholds $g \in G$ that are going to be tested. Let $FAR_{j,g}$ be the FAR obtained on the subset of the training set $j \in J$ for a particular threshold g. Let $|J|$ be the number of subsamples taken from the training data. The procedure for the GHOST algorithm is then as follows:

1. Convert the training probabilities to classes based on a vector of thresholds G that is pre-defined and fixed.
2. Draw $|J|$ stratified subsamples and calculate the metric of interest on each of them across thresholds $g \in G$. In this study, we are interested in the FAR.
3. Calculate the median FAR per threshold g over all $|J|$ subsamples, denoted as $FAR_{med,g}$.
4. Select the final threshold t_{GHOST} such that it yields the smallest absolute distance between FAR_{pre} and $FAR_{med,g}$, that is, $t_{GHOST} = min(g \in G : min(|FAR_{pre} - FAR_{med,g}|))$. Note that the first min argument is used to ensure a unique solution.

In this study, $|J|$ is set to 50 and the subsamples has a size of 20% of the original training data. The thresholds are chosen as $g \in G = \{0, 0.001, 0.002, \dots, 1\}$. We perform random sampling without replacement, based on the default setting chosen by the authors.[14] Originally, the authors implemented their method for Cohen's Kappa but noted that their algorithm can be adapted for the use of other metrics as well,[14] therefore making it suitable for the set-up adopted in this study.

2.4 | Split sample

The last method we test is the Split Sample method (SSM). This method is based on the idea of using a repeated split sample approach to obtain the classification threshold on the training set that can subsequently be used to classify probabilities in the test set. It therefore differs from methods 1-3 as with every split iteration, the model has to be re-estimated. After obtaining the threshold, a new model is fitted on the entire training set for prediction on the test set. The set-up is based on the work by Huberts et al.[13]; for changes made to the original algorithm, we refer to Appendix B.

Let FAR_{val} be defined as the FAR obtained on the internal validation set. By choosing a value u, a vector of thresholds $G = \{0, 1/u, 2/u, \dots, 1 - 2/u, 1 - 1/u, 1\}$ to be tested is defined. The procedure described in the steps 1, 2, and 3 below is repeated Q times and the thresholds t_q (with $q = 1, \dots, Q$) obtained in the Q repetitions are stored in a vector T. The threshold is computed in step 4.

1. Split the training set into an internal training and validation set. Set $w = 1/u$.
2. Fit the model on the internal training set. Predict on the internal validation set to obtain a vector of probabilities.
3. Use the following steps to find the threshold:

(i) Using the thresholds in G, convert the probabilities in the internal validation set to classes and calculate the FAR for each of those thresholds to obtain the vector for FAR_{val}, consisting of all values $FAR_{val,g}$ (i.e., the obtained FAR when threshold g is used for all $g \in G$).
(ii) Find the minimum threshold corresponding to the smallest absolute distance between FAR_{pre} and $FAR_{val,g}$, that is, $t_q = min(g \in G : min(|FAR_{pre} - FAR_{val,g}|))$.
(iii) Update w to $w = 2w/u$ and G to $\{t_q - uw/2, t_q - uw/2 + w, t_q - uw/2 + 2w, \dots, t_q - uw/2 + (u-1)w, t_q - uw/2 + uw\}$. If $w > w_{lim}$, go back to (i). If this is not the case, store t_q in T.

4. Let the final threshold be $t_{SSM} = max(T)$.

We fix the proportion of the internal training and validation set to 0.7 and 0.3, respectively. We repeat the splitting Q=5 times. To specify the range of thresholds, we set $w_{lim} = 10^{-5}$ and $u = 200$. We fix the smallest threshold to be tested to be the minimum probability as estimated on the internal validation set. Note that a higher threshold is associated with a lower FAR; therefore, by taking the maximum obtained threshold in the five splits, a different rationale is followed in this approach compared to taking the average in the other methods.

3 | PERFORMANCE EVALUATION

The thresholding methods are compared with respect to three metrics: the precision, recall, and the difference between the desired and obtained FAR. All metrics are calculated on the test set. Let TP denote the number of true positives, FP the number of false positives, FN the number of false negatives, TN the number of true negatives, and t the threshold. The FAR, precision and recall are then defined as follows:

	True class: control (y=0)	True class: case (y=1)
Predicted class: control (y=0)	True Negative (TN)	False Negative (FN)
Predicted class: case (y=1)	False Positive (FP)	True Positive (TP)

$$FAR(t) = \frac{FP(t)}{FP(t) + TN(t)} \tag{2}$$

$$Precision(t) = \frac{TP(t)}{TP(t) + FP(t)} \tag{3}$$

$$Recall(t) = \frac{TP(t)}{TP(t) + FN(t)} \tag{4}$$

As can be seen in the definitions above, precision describes the number of retrieved relevant items among the retrieved items, whereas recall is a measure of retrieved relevant items among the relevant items. Precision shows how effective a fitted model is at separating instances that are not relevant from the set of instances that were retrieved. A recall value of one can be achieved by simply classifying every instance as a case ($y = 1$).[18] Ideally, a model produces predictions that render both metrics to be as high as possible.

3.1 | ICU dataset

As a real-world illustration, we test the methods on the eICU Collaborative Research Database[19] which includes data collected at critical care units throughout the United States.[19,20] We use a range of predictors (see below) to predict the probability of mortality. To pre-process the data and choose variables, steps based on previous work are followed.[21,22,1] Hereby, individual patient time series are used to create seven segments (full time series, first 10%, last 10%, first 25%, last 25%, first 50%, last 50%) and summary statistics are calculated in each segment to create features (i.e., minimum, maximum, mean, SD, skew, number of measurements).[21] This results in a total of $m = 588$ predictors for $n = 30658$ patients,

[1] The script published by Ulmer et al.[21] (available on Github) is used as a basis and altered to generate the dataset used in this study.

with each row in the dataset corresponding to an individual patient. The final dataset contains 11.49% cases (i.e., patients who died; $y = 1$). The following variables are included:

- blood pH value
- respiratory rate
- body temperature
- heart rate
- diastolic blood pressure
- systolic blood pressure
- Glasgow coma scale (verbal)
- Glasgow coma scale (motor function)
- Glasgow coma scale (eyes)
- Glasgow coma scale (total)
- fraction of inspired oxygen
- blood glucose level
- blood oxygen saturation
- mean arterial pressure

Due to the high number of predictors, we fit logistic Ridge regression and LASSO regression models to deal with issues such as separation and collinearity.[23–25] The Ridge estimator uses the ℓ_2 penalty and is defined as follows:

$$\hat{\beta}(\lambda) = argmin||Y - X\beta||_2^2 + \lambda||\beta||_2^2. \tag{5}$$

The LASSO estimator uses the ℓ_1 penalty:

$$\hat{\beta}(\lambda) = argmin||Y - X\beta||_2^2 + \lambda||\beta||_1. \tag{6}$$

The penalty parameter in the regression models is tuned using 10-fold cross-validation (CV), with minimal deviance as a criterion.[26] We estimate the predictive performance of the final model (i.e., FAR, precision, recall) using 5-fold CV. To preserve the class distribution, we use stratified sampling. The desired FAR values are set to the following: $FAR_{pre} \in \{0.5, 0.2, 0.1, 0.05, 0.01\}$. The results are displayed and discussed in Section 4.

3.2 | Simulation

Covariates X are drawn from the multivariate normal distribution with mean vector $\mu = 0^T$ and variance-covariance matrix Σ with variances diag(Σ) = 1 for individuals (i = 1, \ldots , n). Coefficients are fixed as $\beta = (0.2, -0.5, 0.8)^T$. Values for the binary outcome $y = (y_1, y_2, y_3, \dots, y_n)^T$ are generated from the Bernoulli distribution with probability p_i for each individual i. Let β be defined as the vector of coefficients for the $m = 3$ predictors and β_0 be the intercept. Then the estimated model is of the form

$$log\left(\frac{p_i}{1 - p_i}\right) = \beta_0 + X_{i,1}\beta_1 + X_{i,2}\beta_2 + X_{i,3}\beta_3. \tag{7}$$

The influence of sample size, correlation between variables, and the proportion of cases ($y = 1$) is studied. The number of simulation iterations is set to $n_{sim} = 500$. The following conditions are varied:

1. **sample size**: $n \in \{200, 1000\}$
2. **correlation between covariates**: $r \in \{0, 0.3\}$
3. **approximate proportion of cases**: $prop_{case} \in \{0.2, 0.4\}$

The desired FAR values are set to the following: $FAR_{pre} \in \{0.5, 0.2, 0.1, 0.05\}$. The datasets generated are split into a training and a test set containing 70% and 30% of the observations, respectively. When splitting the data, we use stratified sampling such that the class distribution remains intact. Lower FAR_{pre} values are not tested to avoid issues related to lower case numbers in combination with a lower sample size.

4 | RESULTS

The current study is completed in R[27] (version 4.2.1). The packages `MASS`[28] (version 7.3-57) and `DescTools`[29] (version 0.99.46) are used to generate the simulated datasets. The regression models are fitted using `glm`[27] and `glmnet`[26] (version 4.1-4). `Caret`[30] (version 6.0-93) is used to split the data into training and test sets.

4.1 | ICU data

In our applied data example, there is little variation in the differences between the pre-set and approximated FAR across the OS, Bootstrap, and GHOST methods. The SSM as implemented in the current set-up mostly underperforms in comparison with the other methods. Differences between FAR_{pre} and FAR_{test} range from 0.0001 (condition: OS/Bootstrap, Ridge regression, $FAR_{pre} = 0.5$) to 0.0199 (condition: Split Sample, LASSO regression, $FAR_{pre} = 0.5$) (Table 1, Table E1/Appendix E).

TABLE 1 Pre-set FAR and obtained FAR values for the different thresholding methods on the ICU dataset.

Model	Pre-set FAR	FAR Order Statistics	FAR Bootstrap	FAR GHOST	FAR Split Sample
Ridge regression	0.5	0.5001	0.4999	0.4993	0.4811
	0.2	0.2027	0.2028	0.2025	0.1934
	0.1	0.1022	0.1022	0.1023	0.0972
	0.05	0.0523	0.0521	0.0521	0.0483
	0.01	0.0108	0.0108	0.0109	0.0092

Abbreviation: FAR, false alarm rate.

The range of precision and recall values depends on the pre-set FAR (Appendix C, Tables C1 and C2). In general, it can be seen that the Split Sample-based method performs best in terms of precision, and worst in terms of recall. The other methods performs roughly similarly, with no clear differences between the Ridge and LASSO models. Recall increases with higher pre-set FAR values, while precision decreases. Oppositely, precision increases in scenarios with a lower pre-set FAR. For readability, the results of Ridge regression are reported in the table below, and for the LASSO results we refer to Table E1/Appendix E.

4.2 | Simulation study

Overall, it can be seen that the results obtained in the simulation study fall in line with observations made in the case study. While the OS, Bootstrap, and GHOST based methods perform similarly in terms of obtained results, the SSM as set up in this study often performs significantly different in the three chosen evaluation measures. Across conditions, the main determining factors influencing results are (i) the desired FAR, and (ii) the sample size. The pre-set FAR naturally influences the obtained threshold and therefore has an effect on precision and recall; with a lower threshold, recall becomes higher and precision reduces. In scenarios with smaller sample sizes, the variance in results becomes more pronounced across conditions (Figures 1, 2). For the sake of brevity, we focus on scenarios with $prop_{case} = 0.2$ and $r = 0.3$ below. However, similar patterns are observed across most conditions.

In terms of obtained FAR values, we observe that the OS, Bootstrap, and GHOST based methods usually produce results close to each other, while the Split Sample based method tends to produce a FAR that is lower in comparison (Figures 1, 2). This difference becomes more pronounced in smaller samples. Especially results obtained from the Bootstrap and OS methods approximate each other, but which method has an edge in terms of mean differences between FAR_{pre} and FAR_{test} depends on the pre-set FAR: while the Bootstrap performs better in scenarios where $FAR_{pre} = 0.5$, the OS method usually outperforms it in scenarios where the pre-set FAR is lower (Table 2).

When inspecting the obtained values for precision, the Split Sample based method performs slightly better (Figures 1, 2). This indicates that a larger number of positive predictions are correct. However, this approach also yields the highest number of errors for precision: under certain conditions, no result can be obtained for this metric. Note that precision depends on the number of true positives and false positives (Equation 3): therefore, this can happen when for example, no cases ($y = 1$) are predicted, as the number of true positives as well as the number of false positives is 0. Those scenarios

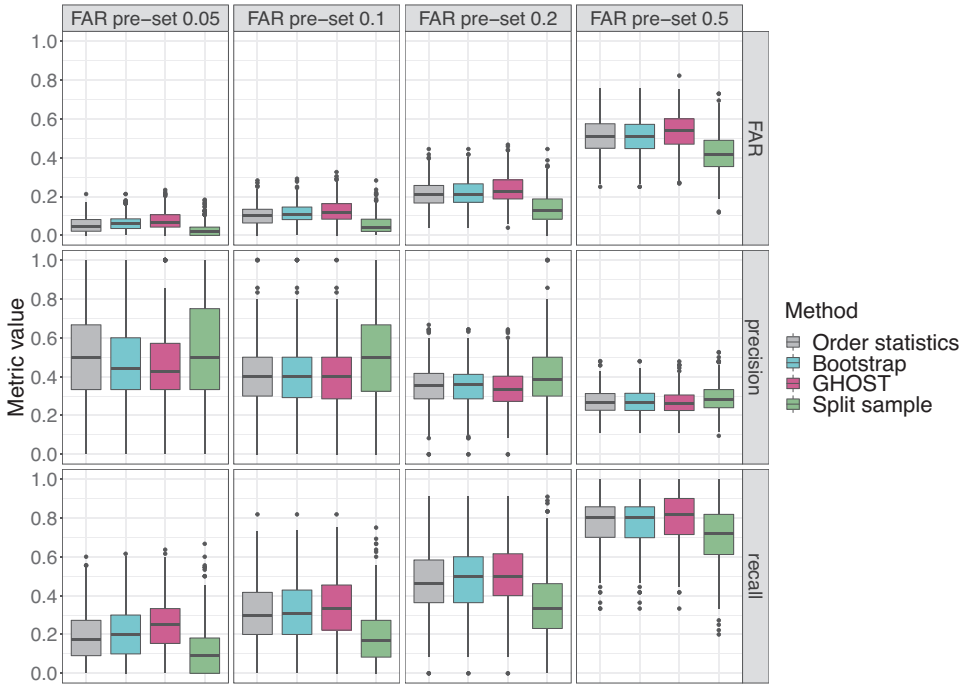

FIGURE 1 Obtained FAR, precision and recall for $n = 200$, $r = 0.3$, $prop_{case} = 0.2$.

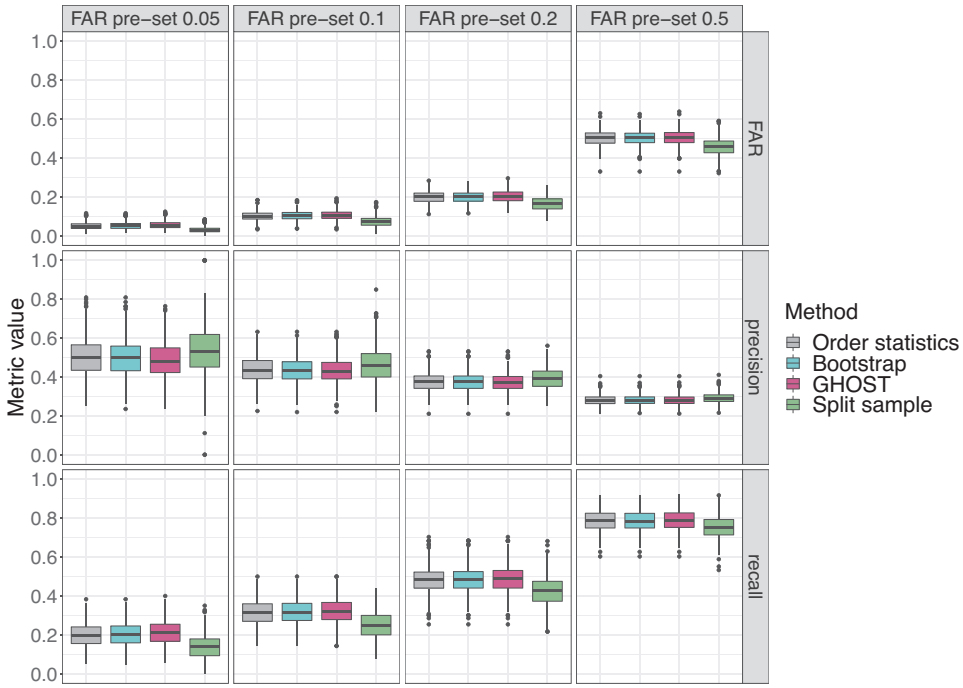

FIGURE 2 Obtained FAR, precision and recall for $n = 1000$, $r = 0.3$, $prop_{case} = 0.2$.

TABLE 2 Pre-set FAR and mean obtained FAR values for $n = 1000$, $prop_{case} = 0.2$, $r = 0.3$.

Pre-set FAR	Mean FAR Order Statistics	Mean FAR Bootstrap	Mean FAR GHOST	Mean FAR Split Sample
0.5	0.5031	0.5026	0.5056	0.4583
0.2	0.19999	0.2006	0.2036	0.1649
0.1	0.1014	0.1031	0.1057	0.0729
0.05	0.0501	0.0519	0.0565	0.0312

Abbreviation: FAR, false alarm rate.

are thus not included in the calculations (Appendix D). Concerning obtained recall values, a different pattern emerges: generally speaking, the SSM performs worst in most cases, and the difference once more generally increase in magnitude in smaller samples. The other three methods produce results that are very close to each other. In terms of the computational time, the OS based method generally was the quickest approach (Figure F1/Appendix F).

5 | DISCUSSION AND CONCLUSION

How to convert probabilities to classes is a crucial part of many predictive modeling efforts. While fitting the model itself (rightly) receives considerable attention in the literature, we argue that thresholding choices made post-modeling can influence the final classification substantially. Paying attention to the thresholding technique can save computational resources, allow practitioners to tune towards a metric of interest and in some situations eliminates the need for resampling in cases of class imbalance.

The key results show that methods based on OS or bootstrapping often performs best and fairly similar to each other. While bootstrapping works slightly better in situations where the desired FAR is 0.5, the OS based method has a slight edge in many scenarios where the desired FAR is lower. The OS based method furthermore provides the advantage of having an overall low computational cost - in contrast, methods based on techniques such as CV require a model to be re-estimated several times, and bootstrapping requires resampling from the set of probabilities B times.

As food for thought concerning future research, we believe that it is important to consider practical challenges when approaching predictive monitoring research. To ease implementations in real-world scenarios, several steps can be taken. First of all, the constant and at times changing influx of data in modern applications should be considered. When and how to update both the model and threshold to cope with data changes[31,32] therefore naturally forms the next step succeeding and complementing the current study. Secondly, the chosen use case determines which specific metric is the focus when tuning the threshold. While the FAR might be at the center in one situation, practitioners may need to focus on another metric, such as recall, or consider a range of thresholds instead in another scenario.[33,34] In their current state, only the Split Sample based method as well as GHOST offer the flexibility to tune towards a wide range of outcomes apart from the FAR. Based on the results obtained in the current study, we believe that adapting the OS method to tune to other outcomes may provide a viable, fast alternative to popular methods such as bootstrap and split sample approaches, and therefore opens ample opportunities for future work.

DATA AVAILABILITY STATEMENT

We used data from the eICU collaborative research database, as well as simulated data. Details on how to obtain the eICU data and the simulation procedure are included in the manuscript. Additionally, information and access to the eICU dataset can be obtained at https://eicu-crd.mit.edu/, whilst the code used to simulate data is available upon request.

ORCID

Paulina von Stackelberg https://orcid.org/0009-0009-4195-0997
Rob Goedhart https://orcid.org/0000-0001-9966-0284
Ş. İlker Birbil https://orcid.org/0000-0001-7472-7032
Ronald J. M. M. Does https://orcid.org/0000-0003-3452-6441

REFERENCES

1. Reifman J, Rajaraman S, Gribok A, Ward WK. Predictive monitoring for improved management of glucose levels. *J Diabetes Sci Technol*. 2007;1(4):478-486.
2. Randall Moorman J. The principles of whole-hospital predictive analytics monitoring for clinical medicine originated in the neonatal ICU. *NPJ Digital Medicine*. 2022;5(1):1-6.
3. Zonta T, Da Costa CA, da Rosa Righi R, de Lima MJ, da Trindade ES, Li GP. Predictive maintenance in the Industry 4.0: A systematic literature review. *Comput Ind Eng*. 2020;150:106889.
4. Carvalho TP, Soares FA, Vita R, Francisco RdP, Basto JP, Alcalá SG. A systematic literature review of machine learning methods applied to predictive maintenance. *Comput Ind Eng*. 2019;137:106024.
5. Zhou J, Li X, Andernroomer AJ, et al. Intelligent prediction monitoring system for predictive maintenance in manufacturing. In: *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005*. IEEE; 2005:6-10.
6. Di Francescomarino C, Ghidini C, Maggi FM, Milani F. Predictive process monitoring methods: Which one suits me best? In: *International Conference on Business Process Management*. Springer; 2018:462-479.
7. King G, Zeng L. Logistic regression in rare events data. *Polit Anal*. 2001;9(2):137-163.

8. Megahed FM, Chen YJ, Megahed A, Ong Y, Altman N, Krzywinski M. The class imbalance problem. *Nat Methods*. 2021;18(11):1270-1277.
9. Zou Q, Xie S, Lin Z, Wu M, Ju Y. Finding the best classification threshold in imbalanced classification. *Big Data Research*. 2016;5:2-8.
10. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. *J Artificial Intelligence Res*. 2002;16:321-357.
11. van den Goorbergh R, van Smeden M, Timmerman D, van Calster B. The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression. *J Am Med Inform Assoc*. 2022;29(9):1525-1534.
12. Cvach M. Monitor alarm fatigue: an integrative review. *Biomed Instrum Technol*. 2012;46(4):268-277.
13. Huberts LC, Does RJ, Ravesteijn B, Lokkerbol J. Predictive monitoring using machine learning algorithms and a real-life example on schizophrenia. *Qual Reliab Eng Int*. 2022;38(3):1302-1317.
14. Esposito C, Landrum GA, Schneider N, Stiefl N, Riniker S. GHOST: adjusting the decision threshold to handle imbalanced data in machine learning. *J Chem Inf Model*. 2021;61(6):2623-2640.
15. Sukchotrat T. *Data mining-driven approaches for process monitoring and diagnosis*. PhD thesis. The University of Texas at Arlington; 2008.
16. Chongfuangprinya P, Kim SB, Park SK, Sukchotrat T. Integration of support vector machines and control charts for multivariate process monitoring. *J Statist Comput Simulation*. 2011;81(9):1157-1173.
17. Albers W, Kallenberg WC. Empirical non-parametric control charts: estimation effects and corrections. *J Appl Statist*. 2004;31(3):345-360.
18. Buckland M, Gey F. The relationship between recall and precision. *J Am Soc Inf Sci*. 1994;45(1):12-19.
19. Pollard TJ, Johnson AE, Raffa JD, Celi LA, Mark RG, Badawi O. The eICU Collaborative Research Database, a freely available multi-center database for critical care research. *Sci Data*. 2018;5(1):1-13.
20. Sheikhalishahi S, Balaraman V, Osmani V. Benchmarking machine learning models on eICU critical care dataset. 2019. *arXiv preprint arXiv:1910.00964*.
21. Ulmer D, Meijerink L, Cinà G. Trust issues: uncertainty estimation does not enable reliable ODD detection on medical tabular data. In: *Machine Learning for Health*. PMLR; 2020:341-354.
22. Harutyunyan H, Khachatrian H, Kale DC, Ver Steeg G, Galstyan A. Multitask learning and benchmarking with clinical time series data. *Sci Data*. 2019;6(1):1-18.
23. Hoerl AE, Kennard RW. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*. 1970;12(1):55-67.
24. Tibshirani R. Regression shrinkage and selection via the lasso. *J R Stat Soc Series B Methodol*. 1996;58(1):267-288.
25. van Smeden M, Moons KG, de Groot JA, et al. Sample size for binary logistic prediction models: beyond events per variable criteria. *Stat Methods Med Res*. 2019;28(8):2455-2474.
26. Friedman J, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. *J Stat Softw*. 2010;33(1):1.
27. R Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing; 2022. https://www.R-project.org/
28. Venables WN, Ripley BD. *Modern Applied Statistics with S*. 4th ed. Springer; 2002. https://www.stats.ox.ac.uk/pub/MASS4/, ISBN 0-387-95457-0
29. Andri et mult al S. *DescTools: Tools for Descriptive Statistics*. 2022. https://cran.r-project.org/package=DescTools, R package version 0.99.47
30. Kuhn M. *caret: Classification and Regression Training*. 2022. https://CRAN.R-project.org/package=caret, R package version 6.0-93
31. Rizzi W, Di Francescomarino C, Ghidini C, Maggi FM. How do I update my model? On the resilience of Predictive Process Monitoring models to change. *Knowl Inf Syst*. 2022;64(5):1385-1416.
32. Márquez-Chamorro AE, Nepomuceno-Chamorro IA, Resinas M, Ruiz-Cortés A. Updating Prediction Models for Predictive Process Monitoring. In: *International Conference on Advanced Information Systems Engineering*. Springer; 2022:304-318.
33. Wynants L, van Smeden M, McLernon DJ, Timmerman D, Steyerberg EW, Calster vB. Three myths about risk thresholds for prediction models. *BMC Med*. 2019;17(1):1-7.
34. Wynants L, Collins GS, van Calster B. Key steps and common pitfalls in developing and validating risk models. *BJOG*. 2017;124(3):423-432.

How to cite this article: von Stackelberg P, Goedhart R, Birbil Şİ, Does RJMM. Comparison of threshold tuning methods for predictive monitoring. *Qual Reliab Eng Int*. 2024;40:499–512. https://doi.org/10.1002/qre.3436

APPENDIX A: BIAS CRITERION FOR OS APPROACH

Consider any random variable X_i for $i = 1, \dots, n_{IC}$. Then consider the Order Statistics (OS) $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n_{IC})}$ with $X_{(1)}$ the minimum and $X_{(n_{IC})}$ the maximum OS, respectively.

Let FAR_{pre} be the desired false alarm rate (FAR). Then, consider

$$k = \lfloor n_{IC} FAR_{pre} \rfloor \tag{A1}$$

with $\lfloor x \rfloor$ the floor function. First consider a one-sided upper control limit $\widehat{UCL} = X_{(n_{IC}-k)}$. This will have conditional (on the dataset) false alarm probability equal to $P_{n_{IC}} = P(X_{n_{IC}+1} > \widehat{UCL} | (X_1, \dots, X_{n_{IC}}))$. Using nonparametric theory based

on the uniform distribution, it can be shown (see Albers and Kallenberg[17]) that the expectation of this equals

$$E(P_{n_{IC}}) = \frac{k+1}{n_{IC}+1} > \frac{k+\delta}{n_{IC}} = FAR_{pre}, \tag{A2}$$

where $\delta = n_{IC}FAR_{pre} - k$. This can be approximately compensated (turned unbiased) by taking an adjusted limit according to

$$\widehat{UCL}^{*} = (1-\lambda)X_{(n_{IC}+1-k)} + \lambda X_{(n_{IC}-k)}, \tag{A3}$$

where $\lambda = FAR_{pre} + \delta$. Note that, for unbiasedness, Albers and Kallenberg[17] suggest using a random Bernoulli variable V with $P(V=1) = 1 - P(V=0) = \lambda$, such that the control limit also becomes a random variable for each sample. We chose to use a weighing between $X_{(n_{IC}-k)}$ and $X_{(n_{IC}+1-k)}$ instead. Note also that, for very small FAR_{pre} such that $k=0$, we require $X_{(n_{IC}+1)}$ which is not part of the sample. Since we are dealing with probabilities in this paper, this can be set equal to 1. Essentially, the reason for this is that using $X_{(n_{IC})}$ as limit would not yield the desired performance, hence extrapolation towards 1 is required.

APPENDIX B: ADAPTATIONS OF ORIGINAL CV ALGORITHM

As mentioned in the main text, the original algorithm proposed by Huberts et al.[13] is altered to accommodate our particular dataset characteristics and for comparison purposes. It should be noted that these alterations may not be suitable for other situations; in their study, the algorithm in its original set-up showed good performance. The following aspects are adapted:

- **Splitting the sample**: we use stratified sampling when splitting the training set into an internal training/validation set.
- **Choosing a threshold**: we choose the threshold in G by selecting the threshold giving the smallest absolute difference between FAR_{pre} and FAR_{val}.
- **Stopping condition**: the original algorithm included a set of stopping conditions related to the difference between the obtained FAR and $FAR_{val,g}$ (e.g., the threshold t_q was included in T if $|FAR - FAR_{val,g}| \leq 0.01FAR_{pre}$). We take this step out to ensure that a threshold can be found in more scenarios.
- **Calculation FAR**: we calculate $FAR_{val,g}$ as $FAR_{val,g} = FP/(FP+TN)$ for each threshold g.

APPENDIX C: FURTHER RESULTS

TABLE C1 Pre-set FAR and obtained precision values for the different thresholding methods on the ICU dataset.

Model	Pre-set FAR	Precision Order Statistics	Precision Bootstrap	Precision GHOST	Precision Split Sample
Ridge regression	0.5	0.1944	0.1945	0.1946	0.1997
	0.2	0.3276	0.3275	0.3278	0.3361
	0.1	0.447	0.4471	0.4469	0.4549
	0.05	0.5563	0.5568	0.5566	0.5694
	0.01	0.7703	0.7713	0.7694	0.7819
LASSO regression	0.5	0.1941	0.1941	0.1938	0.1998
	0.2	0.329	0.3291	0.3289	0.335
	0.1	0.443	0.4428	0.443	0.4497
	0.05	0.5543	0.5542	0.5547	0.5736
	0.01	0.7716	0.7707	0.7681	0.7791

Abbreviation: FAR, false alarm rate.

TABLE C2 Pre-set FAR and obtained recall values for the different thresholding methods on the ICU dataset.

Model	Pre-set FAR	Recall Order Statistics	Recall Bootstrap	Recall GHOST	Recall Split Sample
Ridge regression	0.5	0.9299	0.9299	0.9296	0.9248
	0.2	0.7612	0.7612	0.7612	0.7544
	0.1	0.6368	0.6368	0.6368	0.6258
	0.05	0.5048	0.5048	0.504	0.4923
	0.01	0.2802	0.2819	0.2817	0.2541
LASSO regression	0.5	0.9293	0.9293	0.9293	0.9233
	0.2	0.7643	0.7643	0.7641	0.7533
	0.1	0.6289	0.6289	0.6297	0.6178
	0.05	0.5051	0.5057	0.5057	0.4889
	0.01	0.2785	0.2791	0.2808	0.2592

Abbreviation: FAR, false alarm rate.

APPENDIX D: ERRORS

As described in section 4.2, in some cases, no result for precision can be obtained. This happens more often when the Split Sample approach is used, FAR_{pre} is lower, and the sample size is smaller ($n = 200$). The percentages are reported in the context of $n_{sim} = 500$ iterations (Table D1).

TABLE D1 Precision error percentages.

n	$prop_{case}$	FAR_{pre}	Method	r	Percentage error
200	0.2	0.05	Bootstrap	0.3	1
200	0.4	0.05	Bootstrap	0.3	0.2
200	0.2	0.05	GHOST	0.3	0.6
200	0.2	0.1	GHOST	0.3	0.2
200	0.2	0.05	Order statistics	0.3	3
200	0.4	0.05	Order statistics	0.3	1
200	0.2	0.1	Order statistics	0.3	0.4
200	0.2	0.05	Split sample	0.3	21.6
200	0.4	0.05	Split sample	0.3	18.6
200	0.2	0.1	Split sample	0.3	7.4
200	0.4	0.1	Split sample	0.3	3.6
200	0.2	0.2	Split sample	0.3	0.6
200	0.4	0.2	Split sample	0.3	0.8
200	0.2	0.05	Bootstrap	0	1.2
200	0.4	0.05	Bootstrap	0	0.4
200	0.2	0.05	GHOST	0	0.8
200	0.2	0.05	Order Statistics	0	2.2
200	0.4	0.05	Order Statistics	0	0.4
200	0.2	0.05	Split Sample	0	16.8
200	0.4	0.05	Split Sample	0	14.8
200	0.2	0.1	Split Sample	0	8.0
200	0.4	0.1	Split Sample	0	4.0
200	0.2	0.2	Split Sample	0	0.4
200	0.4	0.2	Split Sample	0	0.2

Abbreviation: FAR, false alarm rate.

APPENDIX E: LASSO RESULTS

TABLE E1 Pre-set FAR and obtained FAR values for the different thresholding methods on the ICU dataset.

Model	Pre-set FAR	FAR Order Statistics	FAR Bootstrap	FAR GHOST	FAR Split Sample
LASSO regression	0.5	0.5009	0.5009	0.5016	0.4801
	0.2	0.2024	0.2022	0.2024	0.1941
	0.1	0.1026	0.1026	0.1027	0.098
	0.05	0.0527	0.0528	0.0527	0.0472
	0.01	0.0107	0.0108	0.011	0.0095

Abbreviation: FAR, false alarm rate.

APPENDIX F: ADDITIONAL EXPERIMENTS

We ran a number of additional experiments to study the methods further. Results can be found below.

Computational demand

Differences between SSM, BST, OS, GHOST

We performed an experiment to compare the computational demand between methods. To study this, we generated 50 datasets (n_{sim}=50) in line with the mechanism described in section 3.2. We then recorded the time until the threshold was obtained per method, as described in Sections 2.1–2.4 (i.e., not including other steps of the procedure, such as dataset generation and fitting of the final model). The same aspects of the datasets were varied as in the main study, that is, correlation, sample size, as well as the average proportion of cases (Section 3.2). In general, we saw that tuning using the OS based method was the fastest. This trend could be seen across conditions. We further saw differences in terms of sample sizes, but no substantial differences with respect to the proportion of cases, pre-set FAR, or correlation. For readability, we therefore decided to average the results and mainly highlight the influence of the choice of thresholding method as well as sample size, as can be found below. It should be noted that some functions, for example, the bootstrap are very efficiently implemented in R, and that logistic regression is not a computationally intensive model. This should be taken into account when looking at the time comparison, especially when considering the results for the GHOST method as implemented for this paper. With computationally very intensive models, it can be expected that a method like GHOST would be faster than for example, the SSM as it does not require retraining of the model. For the authors' original implementation of GHOST,

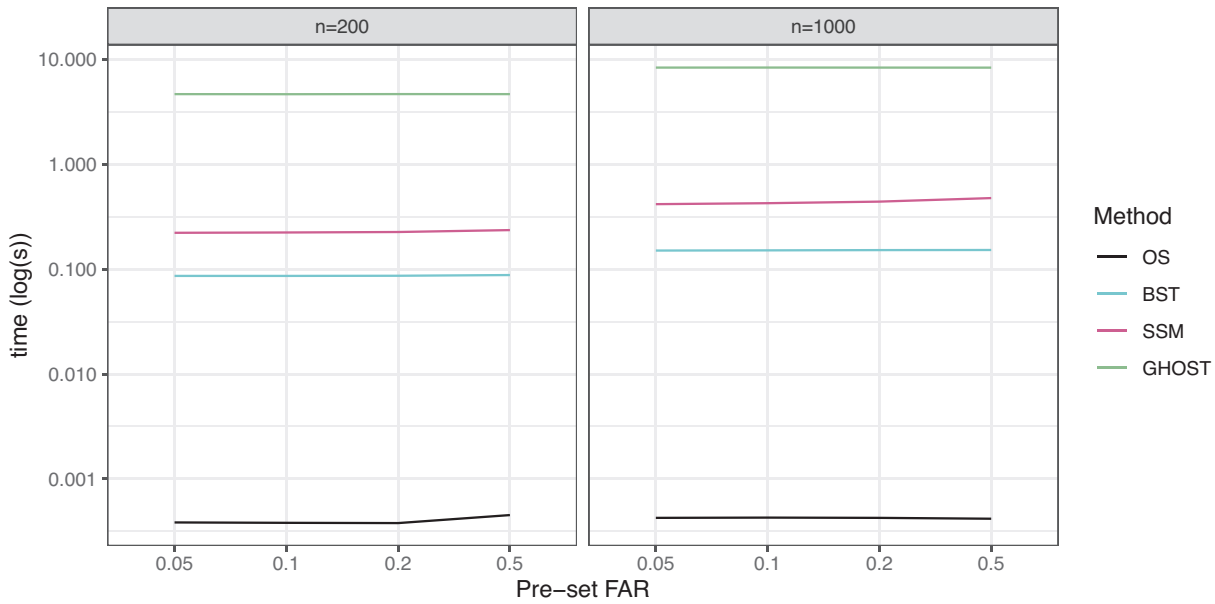

FIGURE F1 Time comparison (in seconds) between the four thresholding methods. Shown as the average per iteration computed over n_{sim}=50 iterations. *Note*: for readability, we used a logarithmic scale for the y-axis.

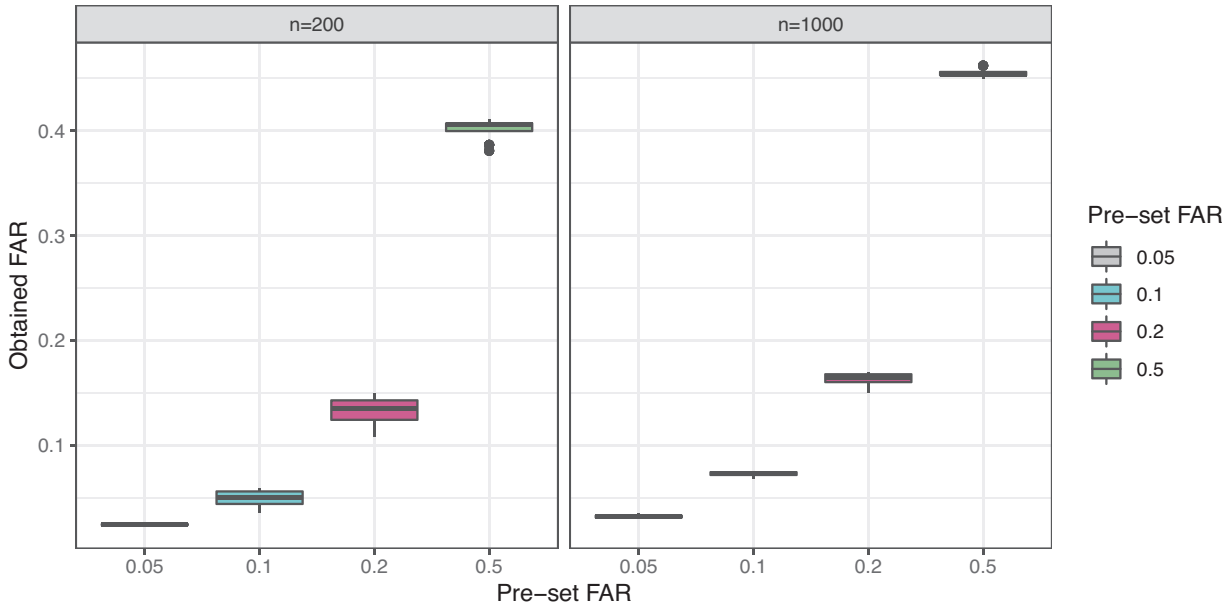

FIGURE F2 Obtained FAR when varying hyperparameters u and w_{lim}. FAR, false alarm rate.

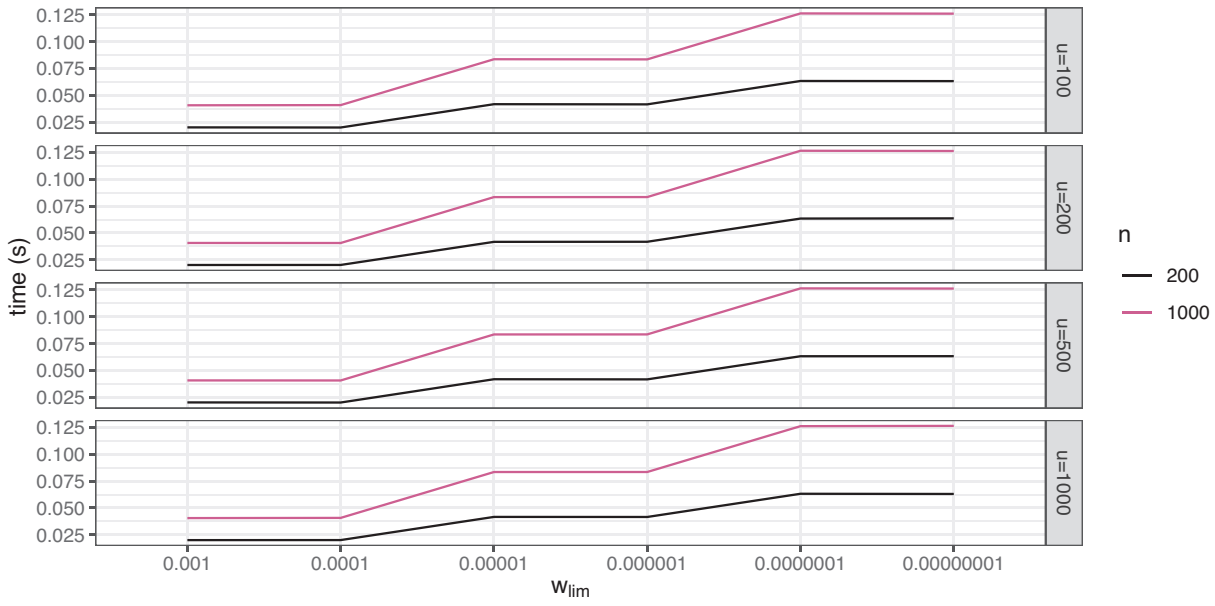

FIGURE F3 Time until final threshold was found when varying hyperparameters u and w_{lim}. Computed per split.

the reader is referred to the original article.[14] However, even with optimal implementation of all approaches, we believe that the order statistics method would still deliver the fastest results, as it does not require resampling or re-training of the model at all and because the speed does not depend on the sample size.

Influence of hyperparameter choices in SSM

We varied the hyperparameters u and w_{lim} to check their influence on the obtained results. We considered all combinations of $w_{lim} \in \{0.001, 0.0001, 0.00001, 0.000001, 0.0000001, 0.00000001\}$ and $u \in \{100, 200, 500, 1000\}$, leading to 24 different combinations. Overall, we can see that the choice of hyperparameter combination with the chosen settings does not show a consistent pattern in terms of influencing results. The parameter combinations lead to roughly similar obtained FAR, as can be seen due to the small variance in obtained FAR per condition (Figure F2). We also calculated the coefficient

of variation for the eight different combinations of pre-set FAR and sample size. It ranges from 0.0085 (condition: n=1000, pre-set FAR = 0.5) to 0.1761 (condition: n=200, pre-set FAR = 0.1).

Furthermore, we checked the computational demand of the 24 different hyperparameter combinations outlined in the beginning of this section. In general, practical differences were small, but it could be seen that with a decrease in w_{lim}, the computational time increased (Figure F3). Furthermore, with a larger sample size, the computational time also increased. Correlation between variables, pre-set FAR, as well as the proportion of cases did not have a significant influence. Therefore, for reasons associated with readability and brevity, we averaged the results across those conditions per w_{lim}/u combination in the figure below.

AUTHOR BIOGRAPHIES

Paulina von Stackelberg is a PhD student in the Department of Business Analytics at the University of Amsterdam, Netherlands. Her research focuses on predictive process monitoring.

Rob Goedhart is an Assistant Professor in the Department of Business Analytics at the University of Amsterdam as well as a senior consultant at the Institute for Business and Industrial Statistics at the University of Amsterdam. He obtained his MSc in Econometrics at the University of Amsterdam in 2014 and his PhD in Statistics at the same university in 2018. His current research revolves around estimation techniques in statistical and predictive process monitoring as well as explainable artificial intelligence (XAI).

Prof. Dr. Ş. İlker Birbil is a Professor of AI and Optimization techniques in the Department of Business Analytics at the University of Amsterdam. His research interests are focused on optimization methods in data science and decision making. He is also interested in data privacy and interpretable machine learning.

Prof. Dr. Ronald J.M.M. Does is Professor Emeritus of Industrial Statistics at the University of Amsterdam and independent statistical engineer. He is member of the editorial boards of the Journal of Quality Technology (since 2015), Quality Technology and Quantitative Management (since 2003), and Quality Engineering (since 2001). He is the recipient of the Hunter Award (2008), Shewhart Medal (2019), Box Medal (2019), and Lancaster Medal (2021). He is a Fellow of the American Society for Quality and American Statistical Association, an elected member of the International Statistical Institute, an Academician of the International Academy for Quality, and Secretary of the International Statistical Engineering Association.