



# A comparison of proximity-based methods for detecting temporal anomalies in business processes

Ioannis Mavroudopoulos<sup>1</sup> · Anastasios Gounaris<sup>1</sup>

Received: 1 March 2021 / Revised: 9 January 2022 / Accepted: 19 February 2022 /  
Published online: 21 March 2022

© The Author(s), under exclusive licence to Springer Science+Business Media LLC, part of Springer Nature 2022

## Abstract

Outlier detection in process mining refers to aspects such as infrequent behavior in relation to the underlying business process models or to anomalous latencies of task execution, termed as temporal anomalies. In this work, we focus on the latter form of anomalies and we aim at investigating in depth the behavior of several proximity-based variants, which are shown to outperform simple statistical ones. We investigate multiple distance functions and approaches to establishing the outlierness of traces or individual tasks, and we explain the superiority of our proposals over existing proximity and probability distribution fitting-based techniques yielding up to 2.05X higher F1 score. We also provide guidelines as to which variant to be chosen based on the type of anomalies targeted and the dataset characteristics.

**Keywords** Outlier detection · Anomaly detection · Business processes · Process mining

## 1 Introduction

Nowadays a lot of businesses turn to Business Process Management (BPM) in order to improve their processes and become more efficient. BPM is the art and science of overseeing how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities (Marlon et al. 2013). Thus the main focus of BPM is to improve the processes in a business. A business process is represented as a set of tasks and their flows, which are orchestrated to achieve a common business goal (Kueng and Kawalek 1997). Since BPM execution is supported by a breadth of software tools, automated log collection is typically enabled. The logs cover a variety of aspects of the execution of a business process instance at a fine level of granularity, e.g., when a specific

---

Editors: Annalisa Appice, Grigorios Tsoumakas.

✉ Anastasios Gounaris  
gounaria@csd.auth.gr

Ioannis Mavroudopoulos  
mavroudo@csd.auth.gr

<sup>1</sup> Aristotle University of Thessaloniki, Thessaloniki, Greece

task of the process was triggered. Each logged event corresponds to a specific trace according to the process instance from which it was generated and the logs contain events from a collection of traces.

These data can be processed using data mining techniques to provide more knowledge about the business than just monitoring (van der Aalst 2016).

In this work, we focus on the broader area of finding anomalies (or equivalently, outliers) in the monitored data of business processes. An outlier is “*an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism*” (Hawkins 1980). The outlying data often contains useful information about the abnormal behavior, which is the reason that is used in applications such as intrusion detection (Jiang et al. 2006), credit card fraud (Tom and Fawcett 2002), and so on Aggarwal (2017).

Detecting deviations from the normal execution of business process based on historical data in a event log is also termed as business process deviance mining (Francesco and Luigi 2018). The targeted abnormalities can be related to a variety of issues, such as delay in a task execution across several traces, or wrong sequences of tasks in a trace.

Most studies in the field focus on finding outlier patterns in the sequence of events in a trace, e.g., (Satyal et al. 2019; Conforti et al. 2017) are proposals that detect such structural anomalies. This is arguably useful as it reveals and gives rise to several issues, e.g., the fact that the process may be executed in multiple not fully aligned manners.

A complementary and less explored approach to performing meaningful outlier detection in the domain of business processes is to identify anomalies in the logged behavior of the activity latencies in terms of non-typical runtimes (durations). The detection can be applied to both individual activities and traces as a whole. More specifically, an outlier event refers to a logged instance of a specific activity, the execution time of which is non-typical compared to other instances of the same activity. Similarly, an outlier trace comprises activity instances, which are characterized by an abnormal combination of execution times that differentiates it from the rest of the traces. These anomalies are termed here as *temporal* ones, as discussed in Rogge-Solti and Kasneci (2014).

In the literature, there are multiple approaches to detecting outliers, regardless of the application domain, such as probabilistic and statistical, linear modeling, proximity-based and clustering ones (Aggarwal 2017; Boukerche et al. 2020).

Proximity-based techniques heavily rely on the notion of  $k$ -nearest neighbors (Goldstein and Uchida 2016). The contribution of this work is that it systematically investigates the performance of several variants of such nearest-neighbor-based outlier detection techniques to deal with temporal anomalies in business processes. We experimentally show that small changes in the rationale of each technique may have a tangible impact on their effectiveness. We also derive conclusions as to which variant to be selected under different settings.

More specifically, we build upon our recent work in Mavroudopoulos and Gounaris (2020), which showed the potential of distance-based outlier detection (Knorr and Ng 1999, 1998) over the statistical approach in Rogge-Solti and Kasneci (2014). Distance-based outlier detection is the most prominent representative of proximity-based methods (Aggarwal 2017) and its main function is the computation of the number of neighbors for each instance. To the contrary, statistical approaches rely on computations of values, such as mean execution times and standard deviations, whereas clustering-based methods first group points in clusters and then detect points not belonging to any cluster and/or post-process the clusters formed using statistical techniques.

The main extension in this work compared to Mavroudopoulos and Gounaris (2020) is that we explore a wider range of alternatives to perform proximity-based outlier detection

that differ in their rationale, the criteria employed and the distance functions although they all leverage the computation of the  $k$  nearest neighbors.

Through thorough evaluation from scratch, we derive insightful conclusions regarding these dimensions when operating both at a trace and a task event level.

In our experiments, we mainly deal with performance in terms of precision and recall in a wide range of parameters and the results show that the proximity-based variants presented in this work are robust and perform significantly better than existing solutions for temporal outlier detection in business process logs.

The proximity-based variants that we propose achieve a higher F1 score up to 1.84X over the proximity-based technique in Hsu et al. (2017) and up to 2.05X compared to the probability distribution fitting technique in Rogge-Solti and Kasneci (2014); the proposals in Hsu et al. (2017), Rogge-Solti and Kasneci (2014) are our direct competitors since they directly target temporal anomalies.

Based on the exact type of temporal anomalies and the amount of outliers in the dataset, we provide guidelines as to which variant to be chosen, since there is not a globally dominant one, as we show in our evaluation results. The complete implementation is provided in open source and our experiments are reproducible; the implementation contains an extensive re-engineering in a more efficient manner than in our previous work in Mavroudpoulos and Gounaris (2020).<sup>1</sup>

The remainder of this work is structured as follows. In Sect 2 we give the formal definitions. In Sect. 3, we present the different methods for detecting trace and event outliers, respectively. In Sect. 4, we present the experimental evaluation. We discuss additional issues in Sect. 5. Next, we present the related work and we conclude in Sect. 7.

## 2 Preliminaries

We follow the same terminology and event log definition as in many other works in business process mining, e.g., (Conforti et al. 2017; Rogge-Solti and Kasneci 2014). More specifically, we assume that businesses have a mechanism to record the corresponding event logs in place, in order to analyze their processes. An *event log* is composed of a set of traces. Each *trace*, or equivalently *case*, corresponds to a specific process instance execution and is identified by a unique *case* identifier. The instance execution is manifested as a recorded sequence of events. Each *event* records the execution of an *activity* in a particular trace (case).<sup>2</sup>

**Definition 1** Event Log: let  $A = \{a_1, a_2, \dots, a_m\}$  be a finite set of activities (tasks) of size  $m$  and  $E = \{e_1, e_2, \dots, e_n\}$  be a finite set of events of size  $n$ . A log  $L$  is defined as  $L = (E, C, \gamma, \delta, ts, \leq)$  where  $C$  is the finite set of Case (Trace) identifiers,  $\gamma : E \rightarrow C$  is a surjective function assigning events to Cases,  $\delta : E \rightarrow A$  is a surjective function assigning events to activities,  $ts$  records the timestamp denoting the finish of task execution, and  $\leq$  is a strict total ordering over events, normally based on execution timestamps.

<sup>1</sup> [https://github.com/mavroudo/Proximity-based\\_OutlierDetection\\_BPM](https://github.com/mavroudo/Proximity-based_OutlierDetection_BPM).

<sup>2</sup> Note that we use the term *event* differently from standards in business processes such as BPMN2.0, where the notion of event has a different meaning.

From the definition above, it is straightforward to calculate the latency of each task based on the difference between its timestamp and the timestamp of its immediate predecessor; process instance start and termination events are included in the logs. The latency includes both the task duration and any waiting time.<sup>3</sup>

In order to detect outlier traces, i.e., traces that deviate significantly from other traces in terms of the latency of their activities,

we first group the  $L$  entries by trace, i.e., case identifier and then convert every trace to a vector of size  $m$ , where each position to this vector corresponds to a distinct activity.

**Definition 2** Trace Vector: given a trace  $t$ , the *Trace Vector* of this trace, is a vector  $(t_1, \dots, t_m)$  where  $m$  is the number of different activities in the process to which  $L$  refers and  $t_i$  is the mean execution time of all events of type  $a_i$  in  $t$ .

In the definition above<sup>4</sup>, execution time should be interpreted as equivalent to task latency.

Given that we employ a nearest neighbor rationale to detect outliers, to avoid some activities dominating the distance computations, a common preprocessing step is to normalize the data. Therefore, no attribute dominates the distance calculation during the outlier detection process. Overall, we create a trace matrix  $[t; t'; t''; \dots]$ , where each row is a trace vector and there are  $m$  columns. We then apply the z-score normalization Equation (1) to every attribute (column in the trace matrix), so that its mean value becomes equal to 0 and its standard deviation is equal to 1 Aggarwal (2017).

$$Z = \frac{X - \text{mean}(X)}{\text{std}(X)}, \quad X \text{ is a column of the trace matrix} \quad (1)$$

Any proximity-based outlier detection technique for log traces involves the computation of the distance between two trace vectors. To this end, in this work, we investigate four distance functions, as follows.

**Definition 3** Trace Vectors Distance: given two *trace vectors*  $t_1, t_2$ , where  $t_1 = (t_1^1, \dots, t_m^1)$  and  $t_2 = (t_1^2, \dots, t_m^2)$ , we investigate four different ways of calculating the distance  $\text{dist}(t_1, t_2)$  between two trace vectors:

$$RMSE(t_1, t_2) = \sqrt{\sum_{i=1}^m \frac{(t_i^1 - t_i^2)^2}{m}} \quad (2)$$

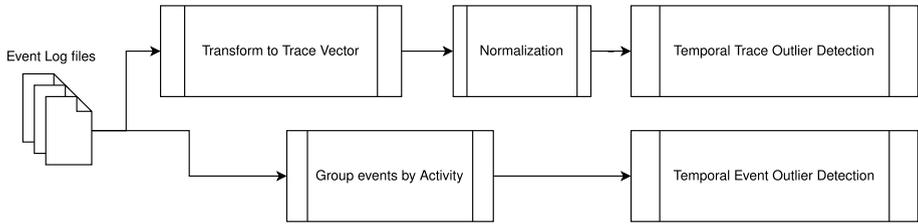
1.

$$Euclidean(t_1, t_2) = \sqrt{\sum_{i=1}^m (t_i^1 - t_i^2)^2} \quad (3)$$

2.

<sup>3</sup> If the logs contain the start and end finish time of each task explicitly, then our approach to detecting latency anomalies can be applied to detecting anomalous task durations in a straightforward manner. Also, in that case, the events need not ordered in a strict order. These aspects do not affect our techniques.

<sup>4</sup> This definition differs from the one in our previous work in Mavroudopoulos and Gounaris (2020), since, in this work we exclusively focus on the temporal attributes of a trace and not on complementary structural aspects.



**Fig. 1** High-level approach

$$3. \quad Chebyshev(t_1, t_2) = \max_{i=1}^m |t_i^1 - t_i^2| \quad (4)$$

$$4. \quad Mahalanobis(t_1, t_2) = \sqrt{(t_1 - t_2)S^{-1}(t_1 - t_2)} \quad (5)$$

$S$  is the covariance matrix.

All the distance functions above are broadly used for applications like proximity-based outlier detection and clustering. The first three ones are simple and easy to compute, whereas the last one can better capture the correlations between different activities. All functions can be used in the outlier detection methods described in Sect. 3 and we evaluate their performance in Sect. 4.1.1.

Outliers regarding the duration of individual activities of the same event type are referred to as temporal outliers at the event level, as opposed to the outliers at the trace level mentioned above. At the event level, each event type is examined separately i.e., log entries in  $L$  are grouped by the activity they refer to through the function  $\delta$ . Given that the dataset comprises single-dimensional points denoting the execution time of activity instances, we employ the Manhattan distance as the distance metric for two points, since RMSE, Euclidean and Chebyshev have the same formula as Manhattan when used in 1-dimension objects and Mahalanobis distance in 1-dimension is equal to Manhattan distance divided by standard deviation, which is a constant value for the events in the same activity and therefore does not impact on the outcome of proximity-based outlier detection. Hence, when looking for temporal outliers at the event level, it is not required to investigate multiple distance functions. Moreover, normalization in the execution latencies is not needed because distances are calculated between event instances of the same type exclusively. The aim of temporal event outlier detection is to identify the event log entries, for which the corresponding execution latency is highly dissimilar compared to the rest of the executions of the same activity type.

Our high-level approach is depicted in Figure 1 and is detailed in the next section.

### 3 Temporal outlier detection based on proximity methods

In this section, we introduce four different variants of outlier detection that can be applied at both the trace and event level. The first three have been introduced in our previous work (Mavroudpoulos and Gounaris 2020), however they were evaluated using only the Euclidean distance. The common feature of all these variants is that, for each trace or event, they perform range queries and examine its neighborhood. Another common feature is that they

heavily rely on user-defined parameters, the sensitivity to which needs to be evaluated. In all variants, the traces are represented as normalized vectors, as discussed in the previous section, forming a trace matrix. We also employ the same variants to detect isolated outlier events. We conclude this section with the discussion of existing techniques for temporal outliers in business process logs.

### 3.1 Proximity-based variants for trace and event outlier detection

When targeting trace outliers, each trace is treated as a multi-dimensional *point*, whereas, for event outliers, the points are single-dimensional ones. According to the first variant, each point is assigned an outlier factor, which is equal to the sum of distances from the  $k$ -nearest neighbors, where  $k$  is a user-defined tuning parameter. As such, the points that, when depicted in a euclidean space, are located in areas with low density, will be reported as outliers. Once we have calculated the outlier factor for every point, we report the top  $\zeta$  values as outliers. Formally, in this variant, the temporal outliers are defined as follows.

**Definition 4** Temporal outlier (Top- $\zeta$ ): given a set of points, a point  $p$  is an outlier if the sum of the distances  $dist(p, p')$  from the closest  $k$  other points  $p'$  is in the top  $\zeta$  values, where  $k$  and  $\zeta$  are parameters defined by the user.

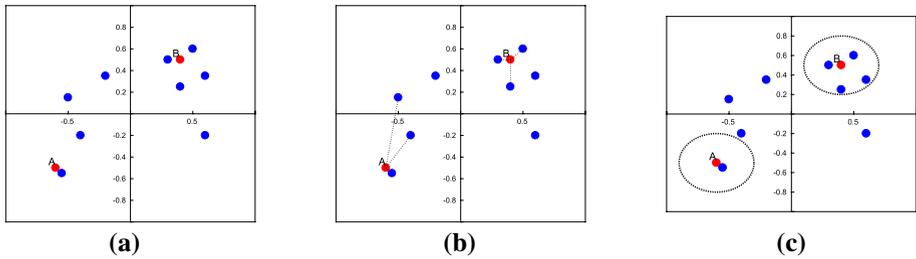
Alternatively, after we compute the outlying factor exactly as previously (i.e., as the sum of the distances from the  $k$ -closest neighbors), we can report the points that their factor deviates more than  $n$  times the standard deviation from the mean value. This method assigns a binary label to each point, depending on whether it is an outlier or inlier, in contrast to the previous variant, which assigns an outlying score and therefore, a threshold is required to distinguish the anomaly points. This variant is better tailored to settings where the number of outliers is unknown, since no  $\zeta$  threshold is defined. We formally define it in Definition 5 and refer to this method as *Probabilistic*.

**Definition 5** Temporal outlier (Probabilistic): given a set of points, a point  $p$  is an outlier if the sum of the distances  $dist(p, p')$  from the closest  $k$  other points  $p'$  is greater than  $mean_{of} + n * std_{of}$ , where  $mean_{of}$  and  $std_{of}$  are the mean value and standard deviation of all the outlier factors, respectively;  $k$  and  $n$  are parameters defined by the user.

In the third variant, we employ the most common form of proximity-based outlier detection, which is also known as binary distance-based, and is proposed in Knorr and Ng (1998), Knorr and Ng (1999). This method is more sensitive to the user defined parameters than the previous methods.

**Definition 6** Temporal outlier (Distance-based): given a set of points, a point  $p$  is an outlier if it has less than  $k$  neighbors in a radius  $R$ , where  $k$  and  $R$  are parameters defined by the user.

To better understand the above definitions, consider the example in Figure 2, where 10 traces of a process containing two activities are mapped to a 2d space. We focus on the two red points, A and B, as shown in the left-most figure. In the middle figure, assuming that we set  $k = 3$ , we depict the neighbors of these two points in line with the Top- $\zeta$  and Probabilistic definitions using the Euclidean distance. It is obvious that the sum of the



**Fig. 2** **a** An example of 10 traces with two activities. **b** The 3 closest neighbors of traces A and B. **c** Example application of the Distance-based variant over traces

distances to the neighbors is greater for point A than for point B, but reporting A as an outlier depends on the  $\zeta$  and  $n$  parameters. Figure 2(c) shows an application of the Distance-based variant, where, with  $k=3$ , A is reported as an outlier and B as an inlier. In general, different results are expected from the different variants.

LOF (Local Outlier Factor) is a (local) density-based outlier detection method, proposed in Breunig et al. (2000). It accounts for varying point densities within the same dataset and aims to address the limitations of the techniques in Knorr and Ng (1998), Knorr and Ng (1999), which can be deemed as global density ones. All variants till now consider a global view of the dataset, since they rely on parameters that are constant for all points on the dataset exclusively. By contrast, LOF compares its neighbourhood against the neighbourhood of its  $k$  closest neighbors. It works similar to the Top- $\zeta$  variant, as it assigns a LOF degree of abnormality, denoted as  $LOF_k$ , for each point (Breunig et al. 2000). When the value of LOF increases, this indicates that fewer points are in close distance from the reference point and thus this point should be reported as an outlier. In order to derive binary decisions as to whether a trace or an event is outlier or not rather than a ranking of traces, we report the top  $\zeta$  points.

**Definition 7** Temporal outlier (LOF): given a set of points, a point  $p$  is an outlier if its  $LOF_k$  is in the top  $\zeta$  values, where  $k$  and  $\zeta$  are parameters defined by the user.

A small note regarding the Top- $\zeta$  and LOF variants when applied on individual events is that it might be the case that we want to report  $\zeta$  outliers across all activities rather than a single one. In that case, we employ the following procedure. We apply 0-1 normalization to the outlying factors reported for each activity; transforming the factors into a common range allows us to report the  $\zeta$  most outlying events from all the activities rather than the top- $\zeta$  of every single activity.

### 3.2 Competitors

Our work is not the first one that proposes nearest-neighbor-based techniques to detect temporal outliers in the field of business processes. Also, there is another technique in the literature that deals with temporal outliers using probability distribution fitting. We briefly describe these two existing relevant techniques below.

ODAL is an earlier proximity-based method proposed in Hsu et al. (2017) to detect anomaly business process instances (traces) using a set of activity-level durations, namely execution, transmission, queue and procrastination. ODAL also uses fuzzy values to present contextual information, such as agents experience, that may have impact on the process. First, all the traces with the same activities are grouped together, and then, for each trace, the distance from the  $k$ -th neighbor in the group is calculated and reported as outlier, if this distance is greater than  $n$  times the standard deviation from the mean value of the other  $k$ -distances in this group. In order to calculate the distance between two traces, ODAL employs the Euclidean distance for every activity in the trace and sums the results. The main differences from the *Probabilistic* variant is that (i) it uses the distance from the  $k$ -th neighbor instead of the sum of distances from the  $k$ -neighbors, making it less robust and (ii) operates on non-normalized trace vectors. We directly compare ODAL against our proposals for proximity-based variants over traces and the experimental results show that the differences mentioned above have a significant impact.

Temporal event outlier detection is first addressed in Rogge-Solti and Kasneci (2014), where Rogge-Solti *et al* presented a method to find temporal anomalies, i.e. anomalies concerning the running time of an activity in a process, using probability distribution fitting. The key point is that, most commonly in real cases, the distribution of the task execution latencies does not follow a normal distribution; so the proposal in Rogge-Solti and Kasneci (2014) leverages a more robust distribution fitting method, which relies on the work in Yeung and Chow (2002). We directly compare the proximity-based variants against this distribution fitting technique as well.

Finally, since there is a trend to employ deep learning for anomaly detection (Pang et al. 2021), we compare the nearest-neighbor-based techniques against autoencoders, which are neural networks that try to recreate the input into their output and have been used in several state-of-the-art solutions for outlier detection. The distance between the input and the output vector can be used as the outlierness score. Two different implementations of autoencoders are used. The first one is a Deep-Learning Autoencoder. Such autoencoders contain more than one hidden layers and they are trained using normal input only Sakurada and Yairi (2014). We use a similar approach to the one in Zong et al. (2018) without the Gaussian mixture and we will describe its structure in detail in the evaluation section. The second one employs Denoising Autoencoders, which are neural networks that consist of one hidden layer. Their advantage is that they are trained in the whole dataset, which is aligned with the methods proposed in this work. In Nolle et al. (2016), the authors use this type of autoencoders to detect structural anomalies in business process logs; however, there is no work to date that employs them for temporal anomalies.

## 4 Evaluation

In our experiments, we have used both real-world and synthetic datasets to evaluate the performance of the proposed methods. We start by presenting the datasets, followed by the evaluation of the different proximity-based trace outlier detection methods, along with the ODAL Hsu et al. (2017). Then, we adapt the proximity-based methods and apply them to the event outlier detection case, while comparing them against the proposal in Rogge-Solti and Kasneci (2014). All tests were conducted in a machine with 16GB of RAM and a 3.2GHz CPU with 8 cores. The source code for all of the proposed proximity-based outlier

**Table 1** Real world dataset characteristics

Dataset	#Traces	#Activities	#Events	Min. events/trace	Max. events/trace	Mean events/trace
BPI_2011	1199	624	52,217	2	101	43.5
BPI_2012	13,087	24	262,200	3	175	20.3
BPI_2015	1143	398	150,291	1	1814	131.4
BPI_2017	31,509	26	1,202,267	10	180	38.1

detection methods is implemented in Scala-Spark to allow for scalability and is publicly available on GitHub, as already stated.

The real-world datasets are taken from the Business Process Intelligence (BPI) Challenges, and more specifically from the 2011, 2012, 2015 and 2017 ones.

- **BPI\_2011**<sup>5</sup> is an event log file from an academic hospital.
- **BPI\_2012**<sup>6</sup> is an event log of a loan application process.
- **BPI\_2015**<sup>7</sup> is a log file that contains traces from a Dutch municipality.
- **BPI\_2017**<sup>8</sup> is an event log, which also corresponds to a loan application of an Dutch financial institute.

More information for each of these datasets are presented in Table 1.

In order to generate a synthetic dataset with ground truth data, we use the PLG2 tool to create a dataset with 10,000 traces, which contain events from 30 different activities. Then, following a similar approach to de Lima Bezerra and Wainer (2013), Nolle et al. (2019), Böhmer and Rinderle-Ma (2016) we inject artificial anomalies that deviate from 3 to 3.5 times the standard deviation from the mean duration time of the activity, affecting a portion of the traces ranging from 0.5% to 10%. The anomalies types are either *Delay anomalies*, where the completion of an activity is delayed, or *Acceleration*, where the completion of an activity is accelerated.

#### 4.1 Evaluation of trace outlier detection

We divide this part of the evaluation in two parts for the synthetic and the real datasets, respectively. The synthetic datasets are accompanied by ground truth data, so that we can accurately report precision and recall values. On the other hand, the real datasets enable us to demonstrate the effectiveness of our proposals in practice.

<sup>5</sup> <http://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffc54>.

<sup>6</sup> <http://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f>.

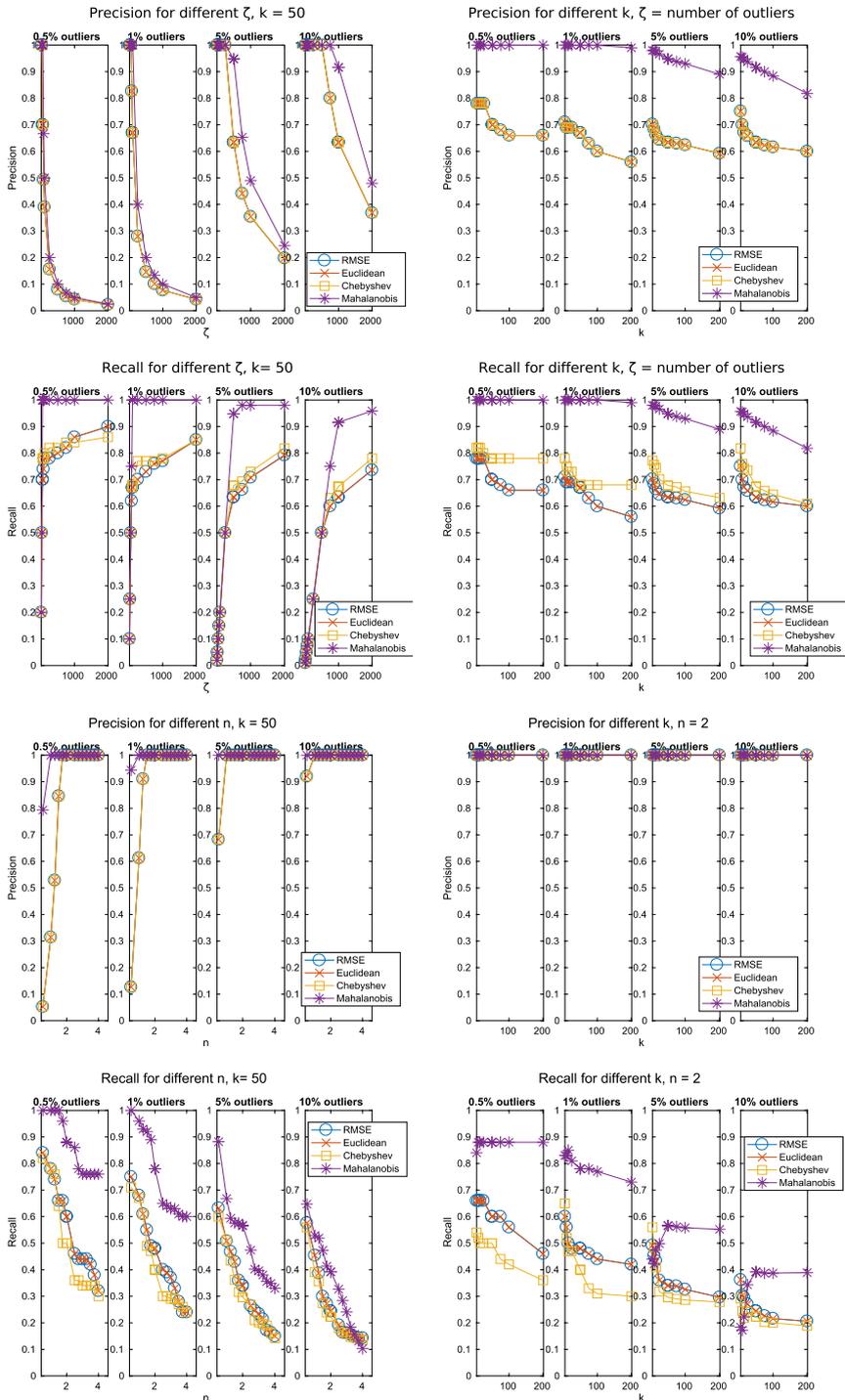
<sup>7</sup> <http://doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17>.

<sup>8</sup> <https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b>.

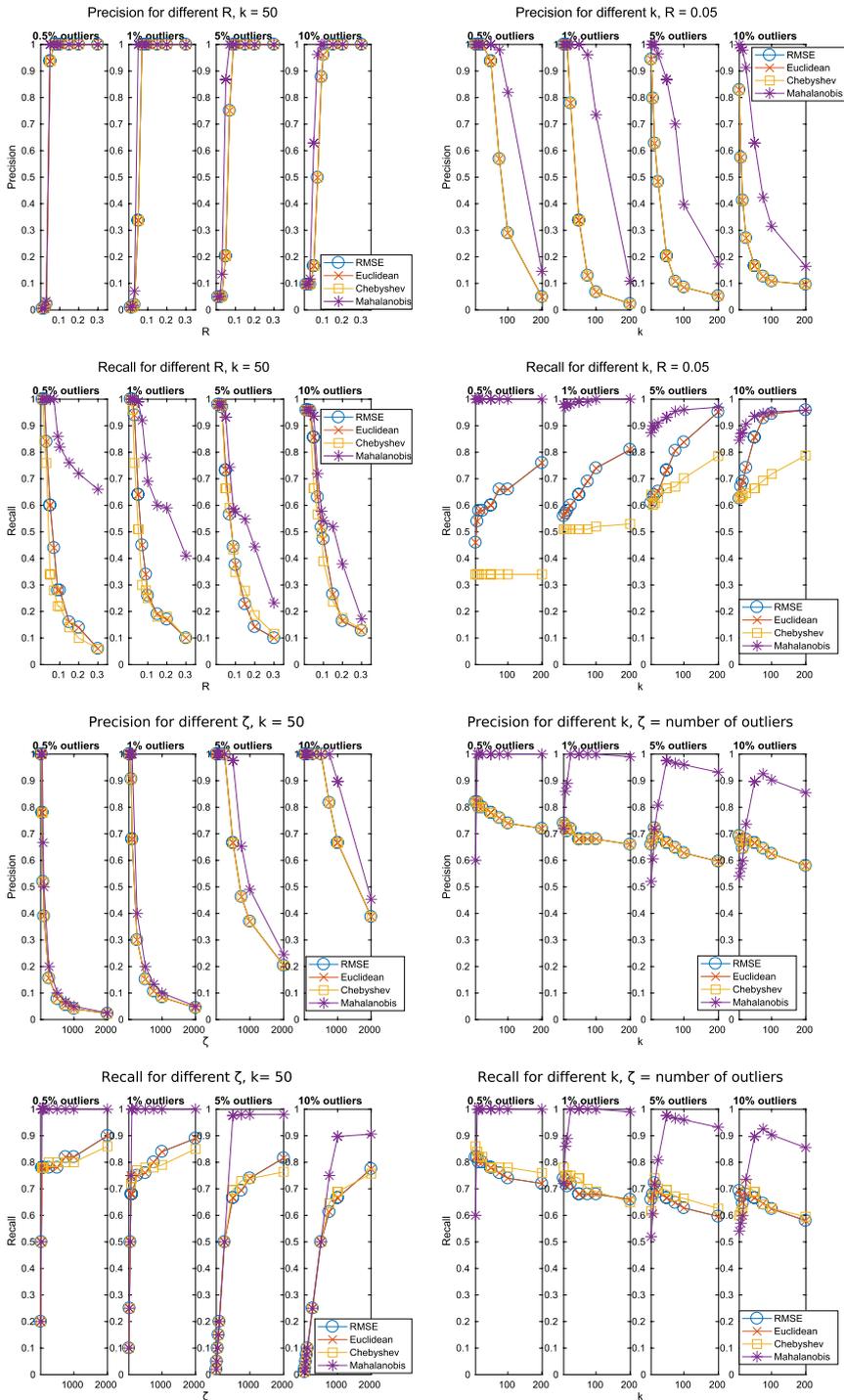
### 4.1.1 Synthetic datasets

We have conducted experiments for the 5 different proximity-based methods in order to find outlier traces. We use 4 synthetic datasets. Each one contains 10,000 traces with injected artificial temporal anomalies affecting a portion of the traces (from 0.5 to 10%) as stated above. Each experiment has been executed for a range of different values of parameters depending on the exact method employed. All methods employ the parameter  $k$ , which defines the number of closest neighbors that will be taken into account; we vary  $k$  in a range from 1 to 200. For Top- $\zeta$  and LOF, we first keep  $k$  equal to 50 while varying  $\zeta$  between 1 and 2000 and then we set  $\zeta$  equal to the number of outliers in the dataset, to test the parameter  $k$ . Regarding the Probabilistic variant and the ODAL method, we first test the parameter  $n$  taking values from 0.5 up to 4.5 with  $k = 50$  and then we keep  $n = 2$  while varying the  $k$ . Similarly, for the Distance-based variant, first we vary  $R$  with  $k = 50$  and then we vary  $k$  with  $R = 0.05$ . Except for the different input parameters, each experiment has been executed using the 4 different distance functions. The results are shown in Figs. 3, 4, 5, the main role of which is to provide strong evidence regarding the suitability of the Mahalanobis distance and shed light on the sensitivity of the variants. We summarize our main observations as follows:

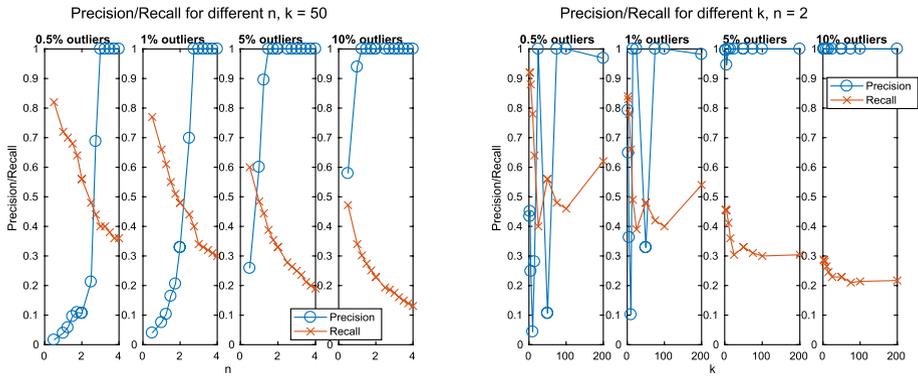
- From all the experiments, we can see that Mahalanobis distance yields either the best results or performs equivalently to the other techniques, and as such is regarded as the dominant choice for the distance metric. Specifically, in LOF (bottom part of Fig. 4), and Top- $\zeta$  (top part in Fig. 3), the Mahalanobis distance performs significantly better, e.g., exhibiting 20% higher precision and 20% higher recall at the same time. The second best distance metric is Chebyshev for all variants apart from the Distance-based one.
- We can observe that LOF and the Top- $\zeta$  have similar results and type of plots with regards to precision and recall. Both methods perform better when the percentage of outliers is small (i.e., in the first two datasets where outliers are up to 1%). The precision and recall decreases as the  $k$  value increases in both methods, which is attributed to the fact that large  $k$  values in these methods correspond to the examination of a particularly extended neighborhood that is typically quite different from the close region.
- Distance-based outlier detection can achieve precision and recall close to 1 across all datasets, for small number of  $R$  and  $k$ . As  $R$  increases, while keeping  $k$  constant, precision increases but recall decreases, since less outliers are reported. The same phenomenon happens while decreasing the  $k$  parameter. Both parameters  $k$  and  $R$  heavily impact on the performance; we discuss tuning later when examining real datasets.
- The Probabilistic variant is evaluated for different values of  $n$ , ranging from 0.5 to 4.5 and even though it can achieve some good combinations of precision and recall, in the generic case (i) the performance is better for the datasets that contain less outliers, and (ii) on average, it is an inferior variant.
- The ODAL method is best suited for large real-world datasets, where the number of neighbors for each trace is sufficiently large so that the statistic threshold is meaningful. In order to make the technique applicable to the synthetic datasets, we consider as neighbors of a trace  $t$ , all the traces that share the same activities with  $t$ , regardless of the order or the number of activity executions. This modification created neighborhoods with approximately 400 traces. We present the results in Figure 5, where



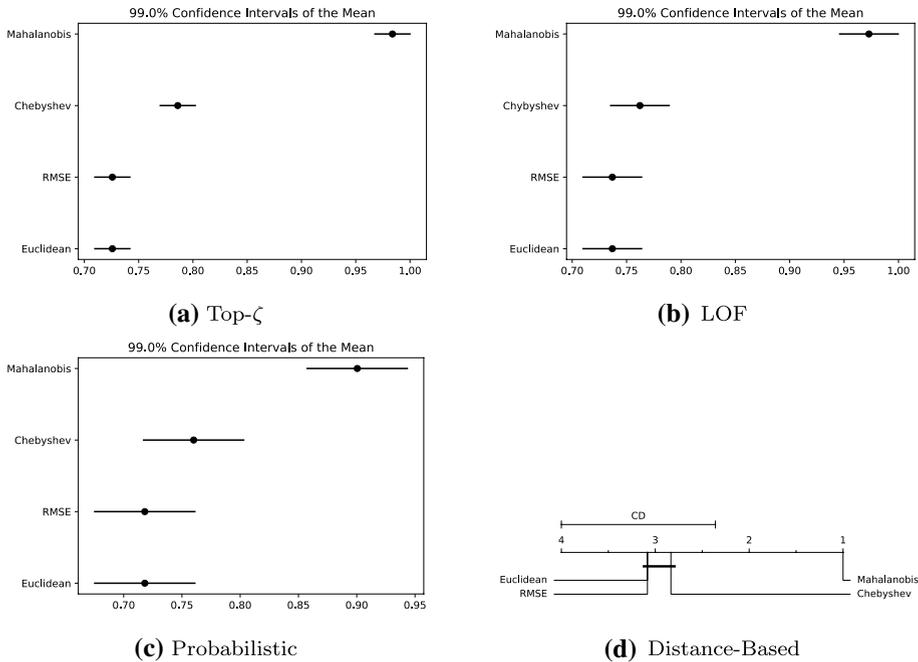
**Fig. 3** Precision and recall for the 4 datasets and the different distance metrics using the Top- $\zeta$  (first two rows) and Probabilistic variant (bottom two rows)



**Fig. 4** Precision and recall for the 4 datasets and the different distance metrics using the Distance-based variant (first two rows) and LOF (bottom two rows)

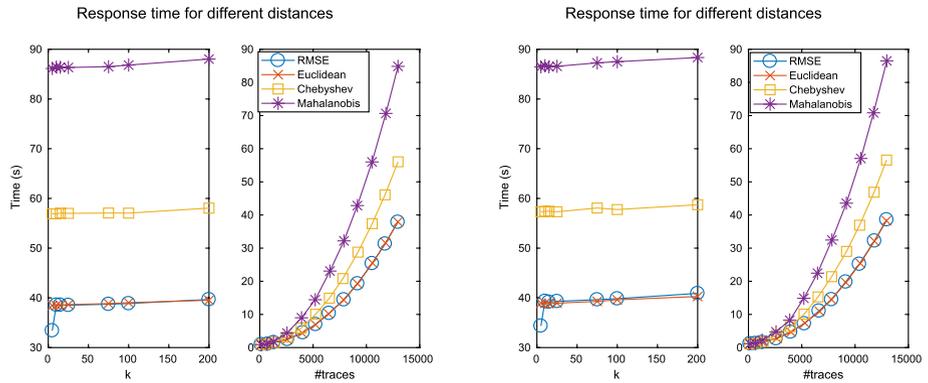


**Fig. 5** Precision and recall for the 4 datasets and the different distance metrics using the ODAL method (Hsu et al. 2017)



**Fig. 6** Critical difference between the different distance metrics for the 4 proposed trace outlier detection methods

we test a wide range of values for  $n$ , ranging from 0.5 to 4.5 and as expected the precision increases as the recall decreases. For different values of  $k$  while keeping the  $n$  parameter constant and equal to 2, we can see that precision is 1 in most of the cases, but reduces significantly for  $k = 10$  and  $k = 50$ . The  $k$  parameter has a large impact on the performance of this method and a fine-tuning is required.



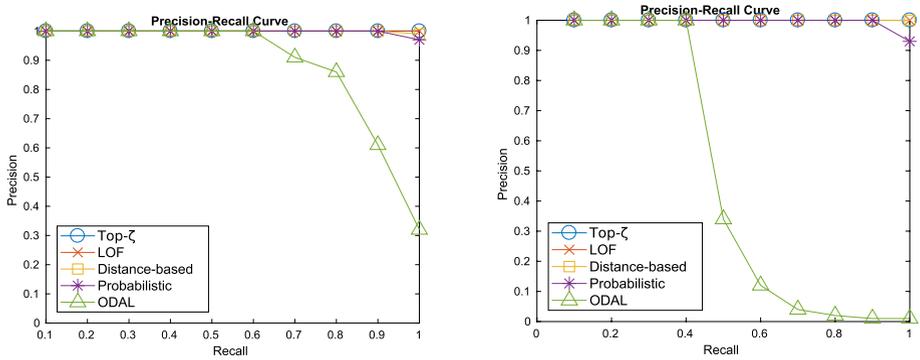
**Fig. 7** Response time for different distance types when calculating the sum of distance from the  $k$ -nearest neighbors as outlying factor (left) and LOF (right)

**Table 2** Best F1 scores produced by each technique for each dataset

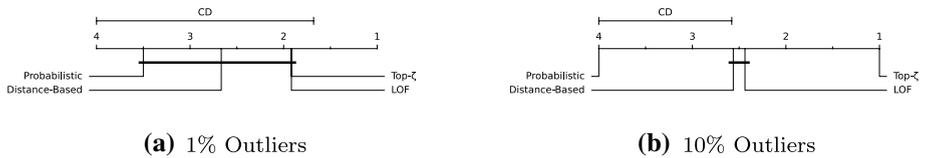
Technique	Outlier percentage			
	0.5%	1%	5%	10%
Top- $\zeta$	1	1	0.98	0.957
Propabilistic	1	0.97	0.93	0.78
Distance-Based	1	0.99	0.94	0.92
LOF	1	1	0.976	0.926
ODAL	0.75	0.81	0.62	0.52

In order to further investigate our observation that Mahalanobis distance has the best performance in terms of F1 score, we conducted critical difference calculations, as described in Demšar (2006), with the use of the *autorank* tool (Herbold 2020). The significance level was set to 0.01 across all tests and we present the results in Fig. 6. In the first 3 methods, the populations proved to be normal and homoscedastic and thus a repeated-measures ANOVA test is used with a post-hoc Tukey HSD (Honest Significant Difference) to generate the diagrams. In the Distance-based method, the populations are also normal but heteroscedastic, which requires a Friedman test to determine if there is significance difference between the metrics, followed by a Nemenyi test for the Critical Difference Diagram. The results from all 4 experiments clearly show that Mahalanobis exhibits a significant difference from the others. The second best distance metric is Chebyshev, that yields significant difference from the other two distance metrics only for the Top- $\zeta$  method. Finally, there is a trade-off between performance and execution time. In Fig. 7, we show the response time of calculating the sum of the distances to the  $k$  closest neighbor, while changing the parameter  $k$  and the number of traces in the logfile. Mahalanobis incurs the highest response time, followed by the Chebyshev. It also worth mentioning that both methods, presented in the figure, have similar response times, despite the fact that LOF is more complex to calculate. That is due to difficulties in the parallelization in the Scala-Spark framework.

Additionally, from Figs. 3, 4, 5, it can be deduced that, in general, Top- $\zeta$ , Distance-based and LOF are capable of achieving better combinations of precision and recall in general, but it is hard to extract a clear picture. Table 2 shows the best F1 score achieved by all



**Fig. 8** The AUC curves when there are 1% outliers for the best  $k$  for each technique (left) and for  $k$  fixed to 50 (right)



**Fig. 9** Critical difference between the different methods using Mahalanobis distance for different portion of outliers

variants examined. From the above experiments, we can draw the following conclusions: (i) the proposed variants significantly outperform ODAL and they can yield up to 1.84X higher F1 score; (ii) Top- $\zeta$  is the dominant variant in terms of the best F1 score achieved, followed by LOF and Distance-based; and (iii) the performance of the Probabilistic variant degrades significantly with increased number of outliers. These conclusions are further supported by the AUC-Precision/Recall curves in Fig. 8.

To assess the statistical significance of the conclusions above, a T-test is performed to assess the significance of the difference between ODAL and the Top- $\zeta$  variant. To this end, we repeated each experiment 10 times using a random sample of 50% of all synthetic data in each iteration. At a confidence level of 0.01, the t-value was 32.1 with the significance threshold being 3.25. Moreover, for the four proposed variants, we show the Critical Difference Diagrams in Fig. 9, where the Mahalanobis distance is employed. On the left side, where the number of outlier is small (1%), there is no significant difference between the variants as also evidenced in Table 2; this is because the small number of outliers, which make it easier to distinguish them from the normal ones in all variants, and consequently, there is no significant difference between them. On the other hand, when the number of outliers is increased to 10% of the total number of traces, we can observe that Top- $\zeta$  has the highest performance, followed by Distance-based and LOF, with no significant difference between them; the Probabilistic variant is inferior to all other tree techniques.

**Comparison against autoencoders.**

We compare Top- $\zeta$  and LOF against the two different implementations of autoencoders.

Since the input vector has length equal to 30, the neural network that implements the deep learning autoencoder runs with FC(30,16,tanh)-FC(16,8,tanh)-FC(8,4,none)-FC(4,8,tanh)-FC(8,16,tanh)-FC(16,30,none), where FC( $x, y, z$ ) means fully-connected

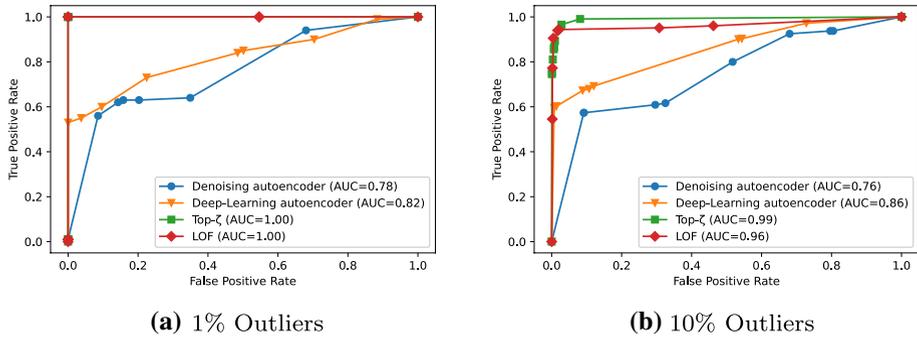
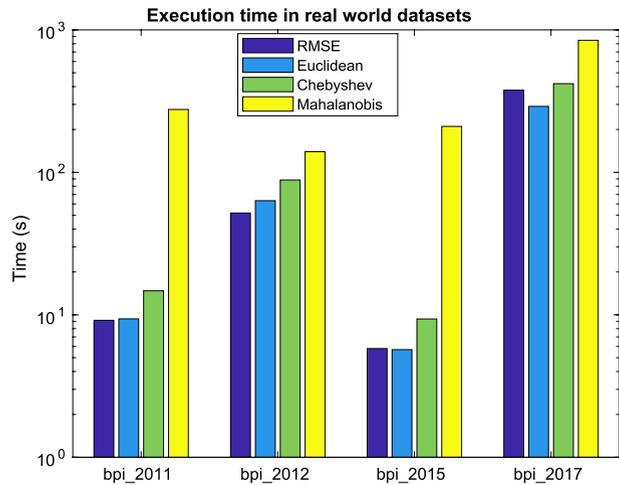


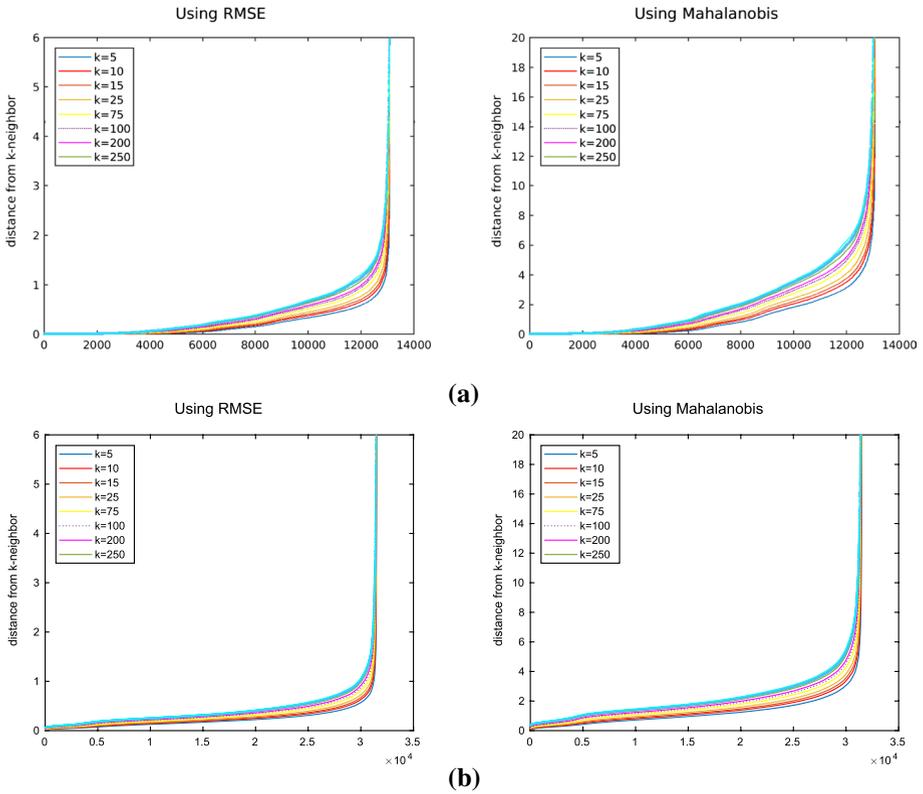
Fig. 10 AUC-ROC curves for different number of outliers

Fig. 11 Execution time for real world datasets, using Top- $\zeta$  and different distances



layer with  $x$  number of input neurons,  $y$  number of output neurons and activation function  $z$  (where none means that there is no activation function. For the evaluation, we split the dataset into 2 sets of equal size: we train the model with the normal traces in the first set and use the second set to evaluate its performance. On the other hand, in order to evaluate the Denoising Autoencoders, we used the whole dataset with the injected Gaussian noise to train the model and then we tested its performance in the original dataset. For the implementation we used the default parameters (e.g. learning rate = 0.01).

The AUC-ROC curves for different number of outliers are presented in Fig. 10. We can observe that Deep-learning autoencoders have better performance than the Denoising ones, but Top- $\zeta$  and LOF perform significantly better than both of them. Because of this, we do not further employ autoencoders in our experiments.



**Fig. 12** Distance from the k-nearest neighbor for different values of k for BPI\_2012 (a) and BPI\_2017 (b) and for 2 different distances

### 4.1.2 Real-world datasets

In this section, we first aim to test the efficiency of the different distances in real world datasets and then verify the effectiveness of our proximity-based outlier detection in the same datasets. Since, as shown in Fig. 7, LOF and Top- $\zeta$  have similar execution time, we only use Top- $\zeta$  to evaluate the different distances and we present the results in Fig. 11. As we can see the response is increasing as the number of traces in the event log increases. Mahalanobis is the distance with the incurs the highest response time, followed by Chebyshev, across all datasets. An interesting observation is that in BPI\_2011 and BPI\_2015, Mahalanobis distance calculation reaches 12 times higher response time than the second slowest distance function. These are two datasets that contain a large number of different activities (624 and 398 respectively), which causes the high overhead when computing the covariance matrix.

Next, to showcase the effectiveness of the approach, we use BPI\_2012 and BPI\_2017, which have the highest ratio of number of traces to different activities. For brevity, we show the traditional Distance-based variant only, because it directly returns the outliers (after setting  $R$  and  $k$ ) and not a ranked list of points, like LOF and Top- $\zeta$  and performs significantly better than the Probabilistic variant and the ODAL method.

As a first step, we need to take an informed decision regarding  $R$  and  $k$  and to this end, we adopt the technique that is used for DBSCAN tuning (Ester et al. 1996; Schubert et al. 2017). Fig. 12 shows the ordered distances from the  $k$ -nearest neighbor for every trace for different values of  $k$ . The most suitable value for parameter  $k$  is the one that corresponds to the plot that is initially as parallel to the horizontal axis as possible, and then, after a sharp change, becomes parallel to the vertical axis. After identifying the  $k$  to which such a plot corresponds, we determine the most suitable  $R$  value by detecting the point where the sharp change occurs. The intuition behind this process is to find the values for parameters  $k$  and  $R$  that best separate the dataset into clusters. In both datasets, RMSE is more suitable than Mahalanobis with regards to specifying the parameters. Most of the lowest values of  $k$  have similar behavior, which implies that some traces are so isolated that parameter  $k$  does not have much impact. We executed the trace outlier detection using the distance-based definition for  $k = 10$  and  $R = 1$  for the BPI\_2012 and  $k = 15$  and  $R = 0.8$  for the BPI\_2017 dataset, and we give 2 examples for each dataset, explaining why this trace is reported as an outlier.

For BPI\_2012 :

1. Trace with id 7905 is reported as an outlier mostly because the mean execution time of its activity “W\_Nabellen offerter” is 340869.5 seconds, while this activity has mean execution time 119704.7 seconds.
2. Trace with id 2099 is reported as an outlier mostly because the mean execution time of its activity “A\_APPROVED” is 19555 secs, while this activity has mean execution time 222.8 secs in the complete log.

For BPI\_2017 :

1. Trace with id 2462 is reported as an outlier mostly because the mean execution time of its activity “W Personal Loan collection” is 128702 secs, while this activity has mean execution time 21333.3 secs in the complete log.
2. Trace with id 23193 is reported as an outlier mostly because the mean execution time of its activity “A\_DENIED” is 0.7 secs. This activity has mean execution time 88042.4 secs in the complete log.

In most cases, reported outlying traces are those violating a correlation between activities. For example, if it is common in a log file that a delay in Activity A is followed by a delay in Activity B, the traces that follow this pattern will have enough neighbors not to be considered as outliers. On the other hand, in the same scenario, if a trace contains a delayed Activity A followed by an accelerated Activity B, this will be a pattern that will render the trace an outlier.

Finally, the configuration procedure described above does not pre-determine the number of outliers reported. A similar approach can be followed when using the Top- $\zeta$  and LOF, but then, the configuration setting of the  $\zeta$  parameter would define the number of reported outliers.

## 4.2 Evaluation of event outlier detection

The experiments above regarding the real data sets reveal the value of assessing the behavior of individual activities. Here, we evaluate the same proximity-based techniques for

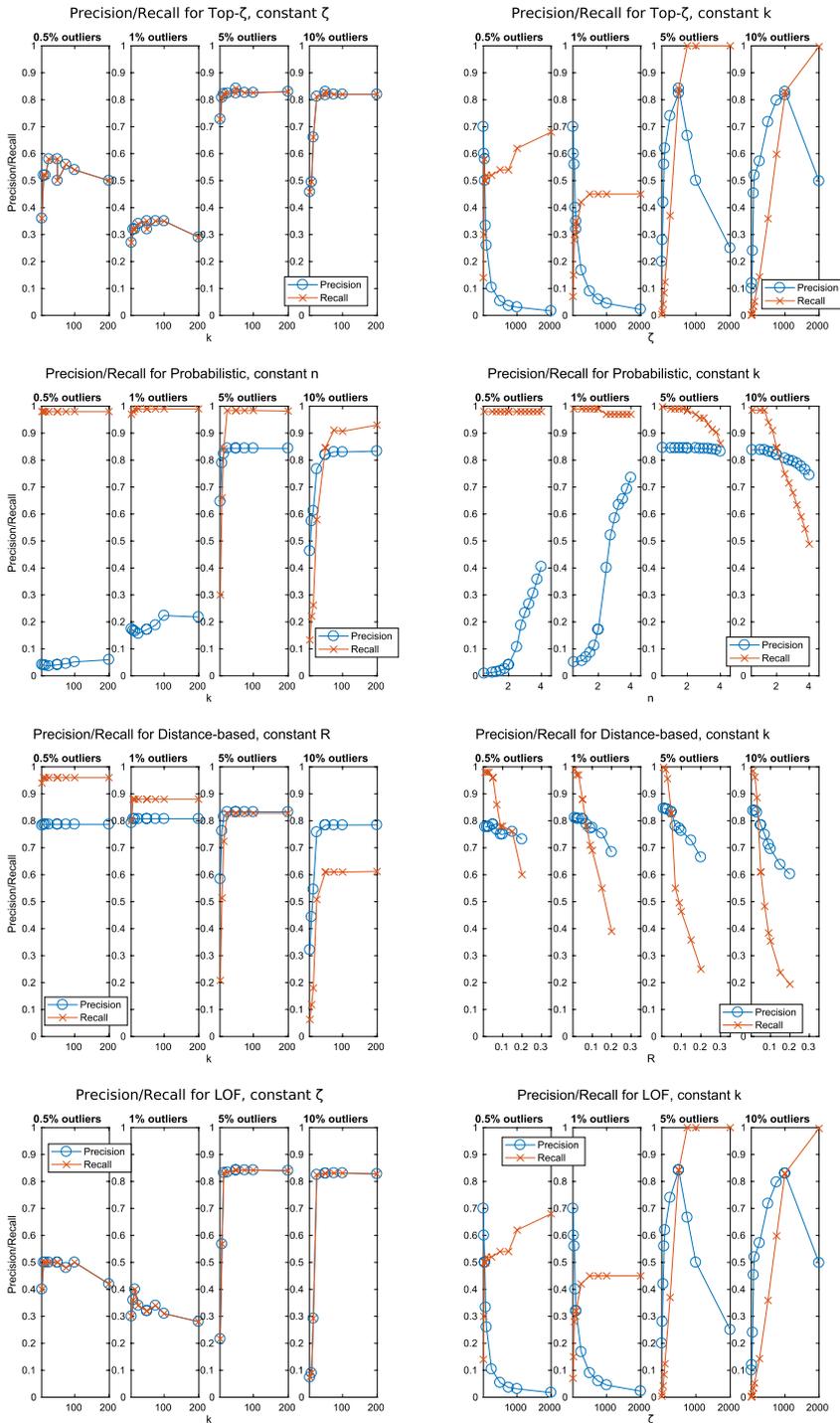
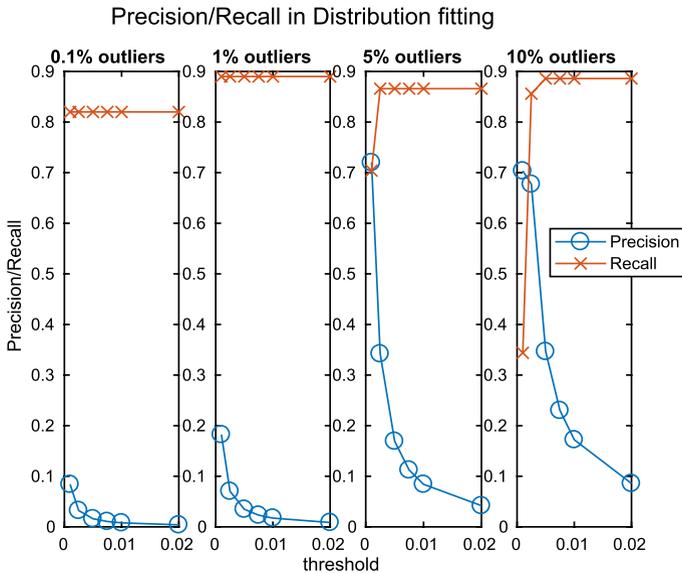


Fig. 13 Precision and recall results for several configurations for the variants Top- $\zeta$ , Probabilistic, Distance-based and LOF (from top to bottom)



**Fig. 14** Precision and recall values using distribution fitting from Rogge-Solti and Kasneci (2014) for varying thresholds

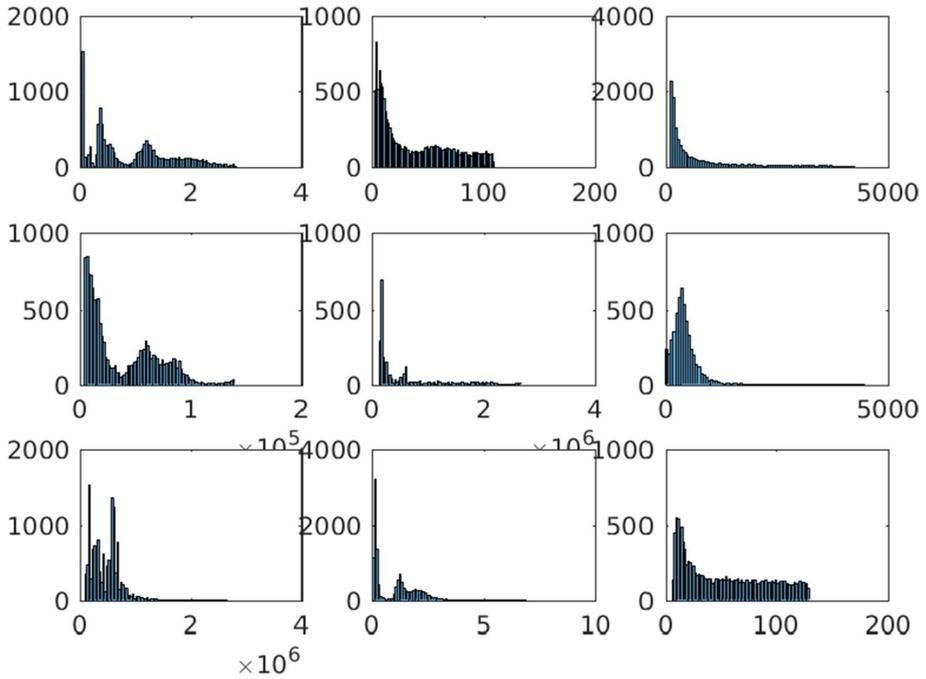
detecting temporal anomalies in events in business process logs systematically. We compare these techniques against the probability distribution-based technique in Rogge-Solti and Kasneci (2014) using the synthetic datasets.

The behavior of the proposed variants is presented in Figure 13 for several configurations. For Top- $\zeta$ , we first keep  $\zeta$  equal to the number of outliers in the dataset and then, in the right plot, we set  $k$  equal to 50. Regarding the Probabilistic variant, we first test the behavior with varying  $k$  and  $n = 2$ , and then we vary  $n$  while  $k$  is set to 50. Similarly, for the Distance-based case, first, we vary  $k$  with  $R = 0.05$ , and then we vary  $R$  with  $k = 50$ . Finally, for LOF we follow exactly the same approach as for Top- $\zeta$ . We aim to report outliers across all activities, after examining each activity one-by-one. Therefore, we also run the normalization step described at the end of Sect. 2. A normalization is also needed regarding the Distance-based variant, in order to use a common  $R$  parameter across different activities that have varying mean execution times.

We summarize some main observations below:

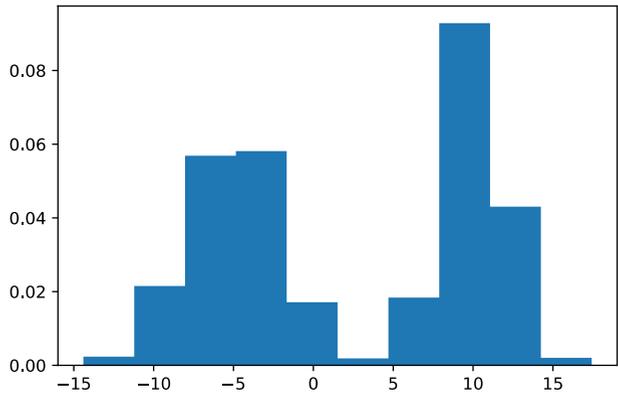
- LOF and Top- $\zeta$  perform similarly, as in the trace case. Their precision and recall is much higher when the number of outliers is increased, where they reach a value of approx. 0.8 (when keeping  $\zeta$  equal to the number of outliers).
- The Probabilistic variant also exhibits higher performance when the number of outliers is higher, whereas  $n$  has more impact than  $k$ .
- The distance-based outlier detection variant is more sensitive to  $R$  than  $k$  and in general, achieves the best average performance.
- Overall, there is not a clear winner, but in general, the Probabilistic and the Distance-based variants are superior to the other two. As previously, there is at least one configuration that can yield very high F1 scores for each dataset.

## Distributions for different Activities in BPI 2012



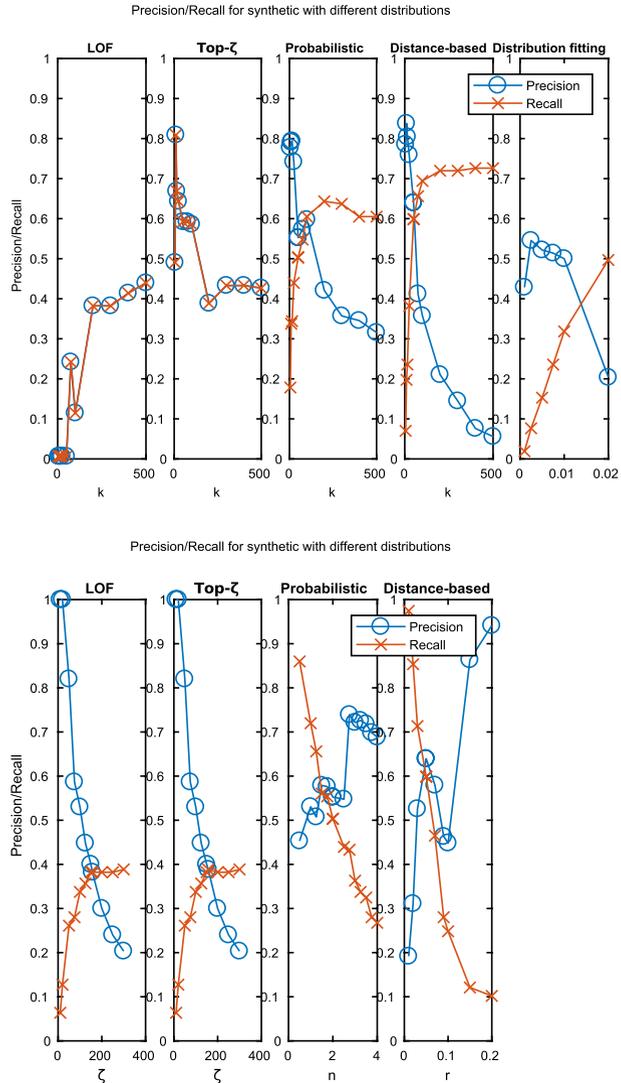
**Fig. 15** Distribution of time duration for 9 different activities in BPI 2012

**Fig. 16** Distribution of task execution latencies for the first activity in the synthetic dataset with different distributions



For completeness, in Fig. 14, we present the results of the technique in Rogge-Solti and Kasneci (2014), which is shown to have poor performance. More specifically, the precision never exceeds 0.2, even for small values of threshold when there are 1% outliers or less. For the other two datasets that have more outliers, recall is close to 1, but it decreases significantly while the threshold increases.

**Fig. 17** Experiments for the synthetic dataset with different distributions for different  $k$  or threshold (top) and fixed  $k$  (bottom)



### 4.2.1 Synthetic dataset with different distributions

In the previous datasets, the duration times for all of the activities follow a (different) normal distribution and this can be considered as a convenient scenario for all methods. Nevertheless, as we show in Figure 15, in real datasets, the distribution of the task latencies in the real-world follow are described by different distributions. In order to better evaluate the methods, we created an additional synthetic dataset, the activities of which do not follow the normal distribution. This additional dataset contains 4 activities with different distributions, namely (i) a combination of two normal distributions (see Figure 16), (ii) a combination of two alpha distributions, (iii) a combination of two exponential distributions and (iv) a power lognormal distribution. It includes 526 traces of 8K events overall, where

every trace contains between 5 to 25 events. The synthetic dataset is provided along with the source code.

We test the effectiveness of each method in this synthetic dataset. Each event is classified as outlier or normal based on whether the probability density function is below 0.01. The precision and recall results appear in Fig. 17.

In the top part of the figure, we set  $\zeta = 157$  for Top- $\zeta$  and LOF, which is equal to the number of outliers.  $n$  is set to 2 for the Probabilistic variant and  $R = 0.05$  for the Distance-based one. We can observe that LOF has rather poor performance with precision and recall not exceeding 0.5 for all  $k$  values. Both the Probabilistic and Distance-based variants achieve precision and recall close to 0.6 for  $k = 15$ . Top- $\zeta$  exhibits the best performance for  $\zeta$  being equal to the number of outliers and  $k = 15$ , where both metrics are above 0.8. On the other hand, the distribution fitting technique from Rogge-Solti and Kasneci (2014) best combinations are inferior to any of our proposals.

Increasing value of  $k$ , while maintaining the other parameter constant in general leads to an increase in the events identified as outliers; therefore in all techniques apart from Top- $\zeta$  the recall is improved. This is accompanied by a negative impact on precision in the variants except LOF. In this dataset, LOF and Top- $\zeta$  behave differently, as the performance of Top- $\zeta$  degrades with larger  $k$  values.

Considering the other parameters apart from  $k$  (bottom part of Figure 17), an increase in the value of  $\zeta$  is shown to have a positive impact in the recall at the expense of lower precision. The opposite impact has the parameter  $n$  for the Probabilistic variant, since increasing the value of  $n$ , a smaller portion of events exceed the threshold. Finally, an increase in the radius  $R$ , where we are looking for neighbors leads to a decrease in the number of reported outlier events and so yields lower recall values.

Overall, despite their sensitivity to their tuning parameters, we can see that proximity-based methods can be adjusted to perform well in datasets where events follow different distributions in a straightforward manner. In this scenario, the best F1 scores achieved for Top- $\zeta$ , Probabilistic, Distance-based, LOF and distribution fitting from Rogge-Solti and Kasneci (2014) are 0.8, 0.61, 0.62, 0.44 and 0.39, respectively. In other words, the proposed proximity-based variants can achieve up to 2.05X higher F1 than the technique in Rogge-Solti and Kasneci (2014).

### 4.3 Summary of lessons learned

In each of the previous sections, we have provided our conclusions and key observations. We complement such conclusions with the three main lessons learned:

- In temporal trace outlier detection, the best nearest-neighbor-based methods are Top- $\zeta$  and LOF, combined with Mahalanobis distance. However, the overhead incurred by this distance function is negatively affected by the number of different activities in the event log. Also, all techniques exhibit better performance in datasets with small amount of outliers.
- In temporal event outlier detection, we observe a different behavior. The best variants with respect to precision and recall are the Probabilistic and Distance-based and the performance of all the variants is increasing as the number of outliers increases. However, when looking for event outliers in activities, the latencies of which follow complex distributions, Top- $\zeta$  and LOF are a safer choice.

- The above statements hold in general, but the amount of outliers also plays a role. More specifically, our distance-based variant should be preferred when searching for event outliers or trace outliers with a few outlier instances. The Distance-based variant is more sensitive to its parameters than the other 3 variants though and thus requires a finer tuning (like the one shown in Sect. 4.1.2) to determine the best values for a given dataset.
- The proximity-based variants that we propose achieve significantly higher F1 score compared to the existing proximity-based approach of ODAL and the distribution fitting technique in Rogge-Solti and Kasneci (2014), 1.84X and 2.05X, respectively. Overall, performing proximity-based outlier detection without first performing normalization yields poor results. Similarly, taking into account only the distance from the  $k$ -closest neighbor and not of the sum of distances from the  $k$ -closest neighbors is sub-optimal. Finally, our proposals are shown to be superior to existing autoencoder-based solutions.

## 5 Discussion

Thus far, we have provided strong evidence regarding the effectiveness of our proximity-based proposals in detecting temporal outliers both at the trace and the activity level. For completeness, we briefly discuss the following four issues:

- As discussed in Rogge-Solti and Kasneci (2014), Mavroudpoulos and Gounaris (2020) it is hard to interpret whether temporal event anomalies are true anomalies or measurement errors. In the latter case, the timestamp of an event has been falsely logged, causing an event to appear as outlier. The key remark is that measurements errors typically affect two consecutive tasks in the trace in a negatively correlated manner. More specifically, if there is no abnormal behavior in the real execution, but due to delayed (resp. early) recording, a task instance has a long (resp. short) latency in the logs, it is expected that the subsequent event will have a recorded short (resp. long) latency. Then, this is (most probably) a measurement error and has to be distinguished from real outliers. We have directly tackled this issue in our previous work (Mavroudpoulos and Gounaris 2020), and the main results are orthogonal to the exact proximity-based outlier detection. More specifically, we advocate, after detecting an outlier in a trace, to check whether the immediate successor is an outlier as well.
- The outlier detection techniques may need to be big-data aware, in terms of both the volume and the velocity of data. The variants examined, and especially the Distance-based one, can directly benefit from recent advances that manage to combine massive parallelism and runtime detection in an efficient manner. Also, to tackle the issue of the sensitivity to the parameters, multiple combinations of parameters can be applied simultaneously. The details are out of the scope of this work and can be found at Toliopoulos et al. (2020).
- Apart from the sensitivity to the parameters, a well known weak point of proximity-based methods is their inefficiency in high-dimensional spaces. In this work, although the trace vectors are high-dimensional, there is no such inefficiency observed. However, in the future, our proposals can be extended with specific techniques tailored to high-dimensional spaces according to the state-of-the-art, as described in surveys, such as Boukerche et al. (2020). However, our techniques are applicable in settings where the

event attributes are not single-dimensional (i.e., activity execution time only). In that case, the trace outlier techniques need to employ a flattened vector with as many elements as the product of the number of activities and the number of dimensions of each event record.

- Another extension to the temporal trace outlier detection methods is the anomaly explanation and the deeper investigation of deep learning solutions. Providing explanations is crucial in the field of business process and helps the domain experts to diagnose the root cause of the anomaly. There are a number of different approaches proposed in the literature, e.g., Myrtakis et al. (2021), which rely on the detection of the appropriate subspaces to explain why an outliers is detected. Additionally, deep-learning solutions for anomaly detection with no need for large volumes of training data is a very active research area. We plan to explore both these directions in the future.

## 6 Related work

There are several proposals about outlier or deviation detection in business processes (Francesco and Luigi 2018). However, most of them focus on detecting outliers regarding the structure of the underlying process model. As business process log trace is typically in the form of a sequence of tasks, outliers can be found in these sequences, e.g., (Conforti et al. 2017; de Lima Bezerra and Wainer 2013). In addition, a typical application is to clear the log dataset removing infrequent behavior in order to facilitate process discovery; process discovery aims to derive the underlying process model out of event logs. Another example of dealing with variations in the process model structure is to allow configurable models, as thoroughly covered in Rosa et al. (2017).

Anomaly detection methods that take into account both the structure of the model and the data attributes, known as multi-perspective ones, have been developed, such as the proposals in Böhmer and Rinderle-Ma (2016), Nolle et al. (2019), Böhmer and Rinderle-Ma (2020). However, none of these approaches consider execution times as continuous variable, and thus are not directly comparable to our approach.

Outlier detection can be used in order to predict the failure of an ongoing process. In Kang et al. (2012), Borkowski et al. (2019), different approaches to predicting the next tasks of an active trace, and based on this prediction to determine if a trace will fail to execute properly, are presented. In addition, a number of different methods for predictive process monitoring have been proposed in the field of machine learning that either predict the outcome of an active trace (Teinemaa et al. 2019; Theis and Darabi 2019) or the type and attributes of the next event (Tax et al. 2017; Pasquadibisceglie et al. 2021; Philipp et al. 2020). Even though there are some similarities between our method and predictive process mining, we aim to just detect the temporal anomalies in a business process logfile and not use this information to predict the future state of a trace, although this would be an interesting extension.

Temporal outlier detection, in the context of BPM, have been addressed in Rogge-Solti and Kasneci (2014), and we have directly compared it to the proximity-based methods. Also, we compare against ODAL, a method proposed in Hsu et al. (2017), which proposes a  $k$ -nearest neighbor technique that considers as neighborhood the traces that contain identical tasks.

From the outlier detection point of view, there exist several textbooks and recent surveys, e.g., (Aggarwal 2017; Boukerche et al. 2020). The techniques based on statistics were the first to be proposed, e.g., through determining data values at the tails of a univariate distribution and the corresponding level of statistical significance. Nearest-neighbor-based outlier detection, which is leveraged in this work, is a representative of proximity-based anomaly detection.

Some of the advantages of this technique is the linear scalability in the size of the dataset (Knorr and Ng 1998; Dai et al. 2019), the ability to interpret the results and operate in a streaming and/or massively parallel environment, e.g., (Toliopoulos et al. 2020), and their wide applicability as reported in Subramaniam et al. (2006). In this work, we have explored a wide range of different proximity-based alternatives to outlier detection, and our work can be deemed as a comparison study of these alternatives in a specific application setting.

## 7 Conclusion

In this work, we deal with the problem of detecting temporal anomalies in event logs of business processes. We investigate four different proximity-based variants and we also investigate the impact of different distance metrics. In our initial work in Mavroudopoulos and Gounaris (2020), we have already provided strong insights into the superiority of distance-based outlier detection over techniques that perform probability distribution fitting. In this work, we complement these early results with new conclusions covering additional proximity-based variants that are explicitly stated in Sect. 4.3. These conclusions also provide guidelines as to which variant to be chosen given the type of anomaly targeted (trace vs. event) and the dataset characteristics. Finally, we have provided all our implementation in open source so that third parties can repeat our experiments.

**Acknowledgements** The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment Grant No.” (Project No.:1052).

**Authors contribution** IM (PhD candidate) and AG (supervisor) actively participated in all stages of research and writing. The experiments and the implementation were conducted by IM.

**Funding** The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the “First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant” (Project Number:1052).

**Data availability** We have used publicly available datasets (providing the appropriate references in the main text) and publicly available tools for generating synthetic data.

**Code availability** All code is available at <https://github.com/mavroudo/Proximity-basedOutlierDetectionBPM>

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Ethics approval** Not applicable

**Consent to participate** Not applicable

Consent for publication Not applicable

## References

- Aggarwal, C.C. (2017). *Outlier Analysis*. Springer.
- Böhmer, K., & Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: International conference on cooperative information systems (CoopIS) 2016 (2016). <http://eprints.cs.univie.ac.at/4785/>
- Böhmer, K., & Rinderle-Ma, S. (2020). Mining association rules for anomaly detection in dynamic process runtime behavior and explaining the root cause to users. *Information Systems*, 90, 101–143.
- Borkowski, M., Fdhila, W., Nardelli, M., Rinderle-Ma, S., & Schulte, S. (2019). Event-based failure prediction in distributed business processes. *Information System*, 81, 220–235.
- Boukerche, A., Zheng, L., & Alfandi, O. (2020). Outlier detection: Methods, models, and classification. *ACM Computing Surveys*, 53(3), 55:1-55:37.
- Breunig, M., Kriegel, H.P., Ng, R.T., & Sander, J. (2000). Lof: Identifying density-based local outliers. In: SIGMOD, pp. 93–104
- Conforti, R., Rosa, M. L., & ter Hofstede, A. H. M. (2017). Filtering out infrequent behavior from business process event logs. *IEEE Transaction Knowledge Data Engineering*, 29(2), 300–314.
- Dai, Q. Z., Xiong, Z. Y., Xie, J., Wang, X. X., Zhang, Y. F., & Shang, J. X. (2019). A novel clustering algorithm based on the natural reverse nearest neighbor structure. *Information Systems*, 84, 1–16.
- de Lima Bezerra, F., & Wainer, J. (2013). Algorithms for anomaly detection of traces in logs of process aware information systems. *Information System*, 38, 33–44.
- Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7, 1–30.
- Ester, M., Kriegel, H.P., Sander, J., & Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In: kdd, vol. 96, pp. 226–231.
- Francesco, F., & Luigi, P. (2018). *Business process deviance mining*. Cham: Springer.
- Goldstein, M., & Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLOS ONE*, 11(4), 1–31. <https://doi.org/10.1371/journal.pone.0152173>.
- Hawkins, D. (1980). *Identification of Outliers*. Springer Netherlands
- Herbold, S. (2020). Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48), 2173.
- Hsu, P. Y., Chuang, Y. C., Lo, Y. C., & He, S. C. (2017). Using contextualized activity-level duration to discover irregular process instances in business operations. *Information Science*. <https://doi.org/10.1016/j.ins.2016.10.027>.
- Jiang, S., Song, X., Wang, H., Han, J. J., & Li, Q. H. (2006). A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letters*, 27(7), 802–810.
- Kang, B., Kim, D., & Kang, S. H. (2012). Real-time business process monitoring method for prediction of abnormal termination using knni-based lof prediction. *Expert Systems with Applications*, 39(5), 6061–6068.
- Knorr, E.M., & Ng, R.T. (1999). Finding intensional knowledge of distance-based outliers.
- Knorr, E.M., & Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the 24th International conference on very large data bases, pp. 392–403 (1998)
- Kueng, P., & Kawalek, P. (1997). Goal-based business process models: Creation and evaluation. *Business Process Management Journal*, 3(1), 17–38.
- Marlon, D., Marcello, L.R., Jan, M., & Hajo A. R. (2013). *Fundamentals of Business Process Management*. Springer.
- Mavroudopoulos, I., & Gounaris, A. (2020). Detecting temporal anomalies in business processes using distance-based methods. In A. Appice, G. Tsoumakas, Y. Manolopoulos, & S. Matwin (Eds.), *Discovery Science* (pp. 615–629). Cham: Springer.
- Myrtakis, N., Christophides, V., & Simon, E.: A comparative evaluation of anomaly explanation algorithms. In: Y. Velegrakis, D. Zeinalipour-Yazti, P.K. Chrysanthis, F. Guerra (eds.) Proceedings of the 24th International conference on extending database technology, EDBT 2021, Nicosia, Cyprus, March 23–26, 2021, pp. 97–108. OpenProceedings.org (2021). <https://doi.org/10.5441/002/edbt.2021.10>.
- Nolle, T., Seeliger, A., & Mühlhäuser, M.: Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: International conference on discovery science, pp. 442–456. Springer (2016)

- Nolle, T., Luetzgen, S., Seeliger, A., & Mühlhäuser, M. (2019). Binet: Multi-perspective business process anomaly classification. *Information Systems*, *103*, 101458.
- Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys*. <https://doi.org/10.1145/3439950>.
- Pasquadibisceglie, V., Appice, A., Castellano, G., & Malerba, D. (2021). A multi-view deep learning approach for predictive business process monitoring. *IEEE Transactions on Services Computing*. <https://doi.org/10.1109/TSC.2021.3051771>.
- Philipp, P., Jacob, R., Robert, S., & Beyerer, J.: Predictive analysis of business processes using neural networks with attention mechanism. In: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC), pp. 225–230 (2020). <https://doi.org/10.1109/ICAIIIC48513.2020.9065057>
- Rogge-Solti, A., & Kasneci, G.: Temporal anomaly detection in business processes. In: BPM, pp. 234–249 (2014)
- Rosa, M. L., van der Aalst, W. M. P., Dumas, M., & Milani, F. (2017). Business process variability modeling: A survey. *ACM Computer Surveys*, *50*(1), 1–45.
- Sakurada, M., & Yairi, T.: Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis, pp. 4–11 (2014)
- Satyal, S., Weber, I., Young Paik, H., Ciccio, C. D., & Mendling, J. (2019). Business process improvement with the ab-bpm methodology. *Information Systems*, *84*, 283–298.
- Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Transaction Database System*. <https://doi.org/10.1145/3068335>.
- Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., & Gunopulos, D. (2006). Online outlier detection in sensor data using non-parametric models. In: VLDB, pp. 187–198.
- Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017). Predictive business process monitoring with lstm neural networks. pp. 477–492.
- Teinemaa, I., Dumas, M., Rosa, M. L., & Maggi, F. M. (2019). Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data*. <https://doi.org/10.1145/3301300>.
- Theis, J., & Darabi, H. (2019). Decay replay mining to predict next process events. *IEEE Access*, *7*, 119787–119803. <https://doi.org/10.1109/ACCESS.2019.2937085>.
- Toliopoulos, T., Gounaris, A., Tsihlias, K., Papadopoulos, A., & Sampaio, S. (2020). Continuous outlier mining of streaming data in flink. *Information System*, *93*, 101569.
- Tom E. & Fawcett, F.P. (2002). Fraud detection. Handbook of data mining and knowledge discovery pp. 726–731
- van der Aalst, W.M.P. (2016). Process Mining - Data Science in Action, Second Edition. Springer.
- Yeung, D. Y., & Chow, C. (2002). Parzen-window network intrusion detectors. *Object recognition supported by user interaction for service robots*, *4*, 385–388.
- Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., & Chen, H.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: International conference on learning representations (2018).

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.