Correlation-based Anomaly Detection in Time Series

Master Thesis

Adrian Hänni

University of Bern

Supervisor
Dr. Mourad KHAYATI
eXascale Infolab, Department of Informatics, University of Fribourg

Co-Supervisor
Prof. Dr. Philippe CUDRÉ-MAUROUX
eXascale Infolab, Department of Informatics, University of Fribourg

September 3, 2020

Abstract

Anomaly detection is one of the most fundamental tasks in time series analysis. The proposed techniques belong generally to one of the following categories: supervised, semi-supervised or unsupervised. Each of these categories assumes a certain trade-off to achieve a good accuracy. In this thesis, we propose to study six different outlier detection techniques, each of them belongs to one of the previous categories. The anomaly detection techniques we study are: Histogram, Cluster (Gaussian Mixture), One-Class Support Vector Machine, Isolation Forest and Robust PCA together with an unsupervised variant of Local Indicators of Spatial Association (LISA). We focus on outliers characterized by significant deviation from other data points in the context of multiple correlated time series instead of exploring point outliers or anomalous patterns within isolated time series. Additionally, we extend LISA using the Dynamic Time Warping to leverage the temporal correlation between time series. All models, but LISA, are applied in a semi-supervised fashion using only normal data for training in all experiments. The result of the empirical evaluation pinpoints to the advantages and disadvantages of each of these techniques.

Finally, we implemented a new online tool, called VADETIS, which allows to i) display time series, ii) evaluate outlier detection techniques and iii) recommend the best technique for a specific dataset. Users can also upload their own time series datasets as well as training data and inject synthetic outliers into time series data.

Acknowledgements

I would like to thank my thesis supervisor, Dr. Mourad Khayati, for his guidance, patience and constant support through this process. His ideas, feedback and our discussions have been absolutely invaluable. Furthermore I express my thanks to Prof. Philippe Cudré-Mauroux for co-supervising this thesis. I cannot forget to thank my family and friends for all the unconditional support during the time I spent working on this thesis. From the bottom of my heart I thank my beloved girlfriend Rese who passed away for all her support and encouragement.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Motivation

Patterns and points in datasets that do not conform well to a model of normal or expected behavior are referred as outliers or anomalies. Often, these two terms are used interchangeably. Detection can be performed on any type of data such as discrete or continuous, univariate or multivariate data. The output is either a score value that defines the confidence for being an outlier, or a binary label resolving whether a data instance or point is considered anomalous.

In many applications, data is generated by one or several recording processes. The occurrence of outliers is a sign of unusual behavior and can provide useful information about anomalous characteristics of the system. Thus, the proper identification of outliers leads to a deeper understanding of the application and the data. In order to define normal behavior within a dataset a comprehensive model of normality is derived. Outliers are recognized as observations which do not fit into such a model as they are characterized by a significant discrepancy. The definition of significance depends, among others, on individual application specific attributes such as type of data, time, domains, seasonality, frequency, trends, noise, etc. Thus, a set of values may be anomalous in a given context, but an identical pattern could be considered as normal behavior in a different context. Many ways are possible to define a model of normal behavior. However, the model is usually formed by an algorithm and the choice of the model is critical for the effectiveness of detection. In order to increase detection rate and confidence, the model may be trained using additional datasets of the same or related domain. The correlation between those datasets can be used to express and improve the model as datasets with strong correlation among themselves are much likely to experience the same trends for the evolution of their observations.

In general, anomaly detection is a complex subject and there is no universal solution that fits best. The choice of an appropriate model using proper parameters to be applied at the correct time as well as the definition of anomalous behavior in the investigated datasets are essential for accurate detection.

1.2 Problem Definition

Detection of anomalies in correlated time series has recently attracted a lot of attention with a wide range of applications, such as intrusion detection implied by identification of strange access patterns, system health monitoring, stock exchange, fault detection in operating environments and many more. The importance of anomaly detection arises from the fact that outliers can lead to misjudgments and wrong interpretations. As outlier event occurrences may not always lead to a severe impact on a system, a successful detection of anomalies can help to detect misbehavior. The quality of the data should be ensured before it is considered for decision making. Detection methods that require model training need classified training data which may not be available or underlies assumptions that do not hold. In this case, detection methods without training that perform similar may be beneficial. Instead of training, detection may be derived from correlations within the data.

In this work, a solution for outlier detection in time series using correlation-based *Local Indicators of Spatial Association* (LISA) is implemented, evaluated and compared to other state of the art techniques. This thesis investigates the detection of anomalies by generalizing the solution proposed in *Scalable Anomaly Detection for Smart City Infrastructure Networks*[1] using different types of correlations among time series. The main contributions of this thesis are:

- We implement different anomaly detection algorithms and we evaluate them on various real-world and synthetic time series data. We empirically study these technique and investigate their performance.
- We introduce a novel and efficient modification of LISA. Instead of the spatial correlation which is applied on vanilla LISA, the correlation between points of different time series is derived with Pearson correlation using a moving window of fixed size. In correlated time series, the evolution of observations is often slightly shifted across different time series. Thus, we propose a variant of Pearson correlation with *Dynamic Time Warping* (DTW) taking this temporal aspect into account. This variant changes the pairing of point values grouping those with lowest distance to each other before Pearson correlation is applied.
- We implement a new online tool called VADETIS (Validator for Anomaly Detection in Time Series). The tool allows users to upload their own time series datasets as well as training data in order to perform and evaluate outlier detection. Users can either share their datasets with others or just use them themselves. Time series can be corrupted by injecting synthetic outliers of different types before detection is performed. The tool is able to recommend the most suitable technique for different performance metrics on a specific dataset.

1.3 Outline

The structure of this thesis is separated into seven chapters. Chapter 2 provides an overview about fundamental concepts integrated in the proposed extension of LISA. Chapter 3 describes anomalous behavior in the context of time series and presents their characteristics and patterns. LISA and other notable detection techniques are covered as well. In Chapter 4, the novel approach for correlation-based LISA is described with example computations. Further, an extension for Pearson correlation that incorporates Dynamic Time Warping is introduced. The performed experiments on real-world and synthetic data are shown in Chapter 5. The performance of all mentioned algorithms is further analyzed, investigated and discussed. A web application VADETIS is presented in Chapter 6, outlining the architecture, layout, implementation and usage of this application. Chapter 7 concludes this thesis and points out to possible future extensions that could be subject to further improvements.

2
Background

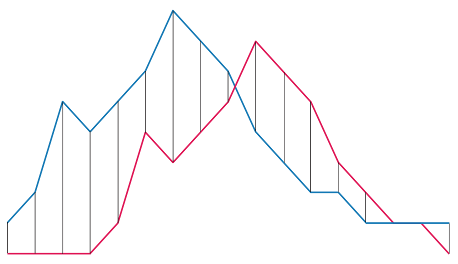
This chapter covers the main concepts used throughout the thesis. We focus on the concepts used for the extension of the LISA technique.

2.1 Dynamic Time Warping

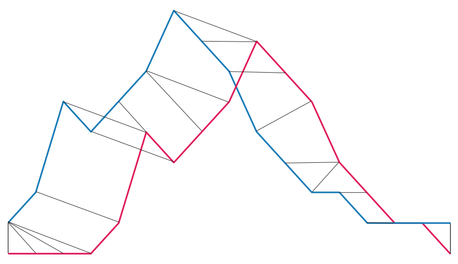
Univariate time series record the changing value of one attribute continuously or over a time period ordinarily at fixed intervals. A time series $X = [x_1, \ldots, x_n]$ is a temporally ordered sequence of n consecutive points in time. The similarity between two time series X and Y in a n-dimensional space is measured by aligning two sequences and computing a distance between them. The distance value quantifies the extent of the similarity between the two time series and can be calculated using distance functions such as Euclidean distance. Time series might be shifted or be compressed in time which which is not taking into account by Euclidean distance. Consequently, two time series that share a very similar pattern might have a high distance according to the Euclidean formula, although they look visually similar [2].

A more flexible method that can arrange the best suitable mapping from elements in X to those in Y is used in order to compute the distance. A more optimal alignment between two time series can be obtained using Dynamic Time Warping (DTW). This method allows to find the best alignment between two time series by compressing or expanding in time. The best mapping is determined by the mapping resulting with the minimum distance that is achievable using a given distance metric. The more similar two elements x and y the lower their distance $d(x, y)$ to each other. Figure 2.1 contrasts DTW with the Euclidean distance mapping. Both time series show the same pattern but shifted slightly on the vertical and more obvious on the time axis. Black lines indicate which values are matched for a distance function. Euclidean distance computes a larger distance compared to DTW because DTW matches the time series in a way that the patterns are aligned by warping the time axis.

To find the best mapping for two time series $X = [x_1, x_2, ..., x_n]$ and $Y = [y_1, y_2, ..., y_m]$ of different or equal length n respectively m, a distance matrix $D \in \mathbb{R}^{n \times m}$ is computed [3]. Figure 2.2a illustrates the distance matrix for series X and Y with dark elements for low distances and bright elements for high distances. The optimal path through this matrix has minimal overall distance and defines the values mapping for DTW as each matrix element $(i, j) \in D$ represents an alignment between point pair (x_i, y_j). The warping path is a sequence $W = (w_1, w_2, ..., w_L)$ with $w_l = (i_l, j_l) \in [1 : n] \times [1 : m]$ for $l \in [1 : L]$.



(a) Euclidean distance



(b) DTW

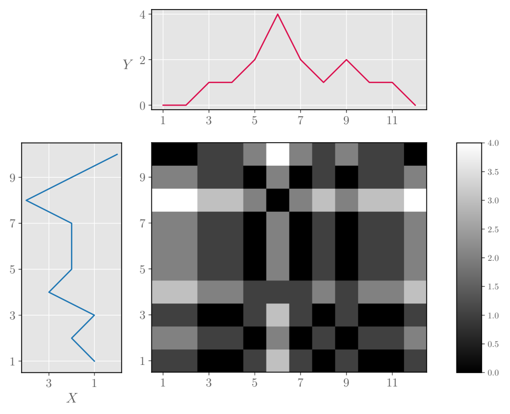
Figure 2.1: Alignment of two time series for Euclidean distance and DTW. Aligned points are connected with black lines.

L denotes the pair alignment number with $\max\{n,m\} \leq L \leq n+m-1$. Valid warping paths must satisfy the following conditions:

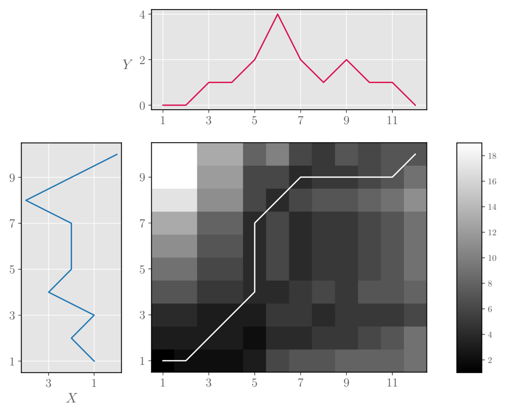
(i) **Boundary condition**: $w_1 = (1,1)$ and $w_L = (m,n)$.

(ii) **Monotonicity condition**: $i_1 \leq i_2 \leq ... \leq i_L$ and $j_1 \leq j_2 \leq ... \leq j_L$.

(iii) **Step size condition**: $w_l - w_{l+1} \in \{(1,1),(1,0),(0,1)\}$ for $l \in [1:L-1]$



(a) Distance matrix



(b) Accumulated distance matrix

Figure 2.2: **(a)** Distance matrix of two time series X and Y using the Euclidean distance as local distance measure d and **(b)** accumulated distance matrix with optimal warping path W.

The boundary condition (i) specifies that the first elements of time series X and Y as well as their last elements are aligned with each other. Therefore, the warping path covers the entire sequences of X and Y. The monotonicity condition (ii) constrains the path as it will not turn back on itself because a subsequent index i_{l+1} or j_{l+1} either remains unchanged or increases by one. Further, the step size condition (iii) enforces that the alignment path does not skip a time index and no duplicates of alignments exists. Consequently, no element in X as well as in Y is omitted by a warping path. The step size condition (iii) implies monotonicity condition (ii) because $w_l = (i_l, j_l)$ and $w_{l+1} = (i_{l+1}, j_{l+1}) \in \{(i_l + 1, j_l + 1), (i_l + 1, j_l), (i_l, j_l + 1)\}$, then $i_l \leq i_{l+1}$ and $j_l \leq j_{l+1}$.

The warping path distance is calculated from the first cell $(1, 1)$ to the last (n, m) with equation 2.1 as sum of all distances of each index alignment of the path. The DTW distance is the optimal path between X and Y having minimal total distance among all possible paths as equation 2.2 shows.

$$d(W) = \sum_{i=1}^{L} d(w_i) \tag{2.1}$$

$$d(w_l) = d(x_{i_l}, y_{j_l})$$

$$DTW(X, Y) = \min_{\forall W}\{d(W)\} \tag{2.2}$$

Calculating all possible warping paths to determine the optimal path is expensive (i.e., exponential time complexity with n). The most commonly used technique to calculate DTW distance is the concept of an accumulated distance matrix (ADM) which can be computed with complexity $O(nm)$. Each element in ADM, as illustrated in Figure 2.2b, contains the respective value in the distance matrix plus the lowest preceding accumulated distance. Possible preceding elements for accumulated distance values are defined similar to the step size condition as shown in equation 2.3. This matrix is used to develop a warping path which follows through the cells with the lowest accumulated distances, thereby minimizing the total distance difference between the two sequences. The value of the top right cell (n, m) is the DTW distance of X and Y.

$$\begin{aligned} DTW(X, Y) &= f(n, m) \\ f(i, j) &= d(x_i, y_j) + \min \begin{cases} f(i, j-1) \\ f(i-1, j) \\ f(i-1, j-1) \end{cases} \\ f(0, 0) &= 0 \\ f(i, 0) &= \infty \qquad \text{for} \quad i \geq 1 \\ f(0, j) &= \infty \qquad \text{for} \quad j \geq 1 \end{aligned} \tag{2.3}$$

2.2 Correlation

Correlation describes the degree of similarity and relationships between two variables or time series. Positive correlation is a relationship between two time series in which both evolve in the same direction while negative correlation is a relationship in which both evolve in opposite directions to each other. The Pearson correlation coefficient, also known as r, measures the strength of the linear correlation between two time series $X = [x_1, \ldots, x_n]$ and $Y = [y_1, \ldots, y_n]$ of equal length n. It is defined as:

$$r(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.4}$$

The value of r is undefined if all x_i (and/or y_i) are equal. If not, the value is in range from -1 to 1 and is interpreted as follows:

- $r \approx 0$: If two variables have a correlation value of approximately zero, no correlation is recognizable. This means the two variables are uncorrelated and that there is no linear correlation between the variables.
- $r > 0$: If r is higher than zero, this is called a positive correlation. Higher values of X correlate with higher values of Y as well as lower values of X correlate with lower values of Y. $(x_i - \bar{x})(y_i - \bar{y})$ is positive only if x_i and y_i lie on the same side of their respective mean values. Therefor the correlation is positive if x_i and y_i tend to be simultaneously greater than, or simultaneously less than, their respective means.
- $r < 0$: If r is negative, then higher values of X correlate with lower values from Y and vice-versa. A negative correlation value refers to anti-correlation and happens if x_i and y_i lie on opposite sides of their respective mean values. Further, the more significant is either tendency, the larger the absolute value of r.

In general, values of $r \in [0.7, 1]$ stand for highly correlated variables.

3 Anomaly Detection in Time Series

This chapter defines anomalous behavior in univariate time series. It gives an insight into the various challenges, anomaly types and categories that can occur and presents dissimilar outlier patterns. Further, we introduce the detection methods which have been used for evaluation. All the techniques, except LISA which is an unsupervised approach, can be applied in a semi-supervised fashion.

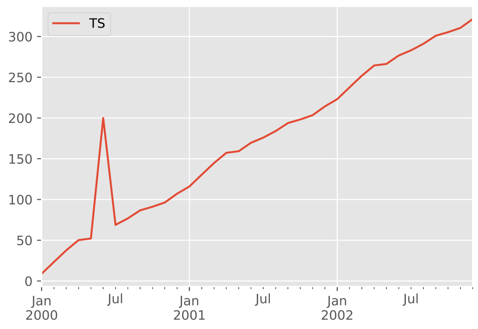
3.1 Challenges

The problem of outlier detection in time series is often formulated as finding outlier data instances or points relative to standard or conventional signal which are significantly distinct from the majority of the data. Statistically speaking, significance means that the statistical properties of the data instance or point is not in alignment with the rest of the data. Thus, the detection of anomalies is often not unequivocal. Several factors associated with anomaly detection for time series lead to major challenges [4]:

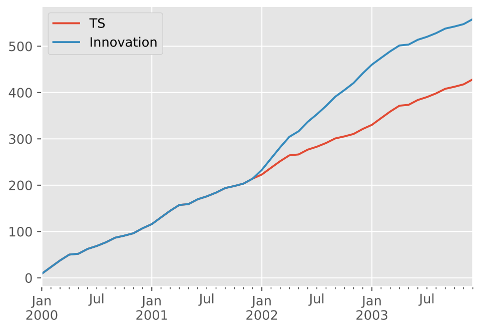
- Different ways exist to define an anomaly that occurs in a time series. Single observations within a time series may be anomalous as well as subsequences of consecutive observations. Further, an entire time series could be anomalous with respect to others which are considered as normal.
- The data may contain noise which might be similar to anomalous behavior. This arises from the fact that the boundary between normal and anomalous behavior is often not precise.
- It can be difficult to define a model of normal behavior of the data that covers the whole normal behavior.
- The definition of normal or anomalous may frequently change as the data keeps evolving. Thus, normal behavior of the past might not fit into normal behavior in the future.
- The classification of outliers varies for different application domains. A deviation level that signals abnormality in one domain might be considered within normal behavior in another.

3.2 Types of Anomalies

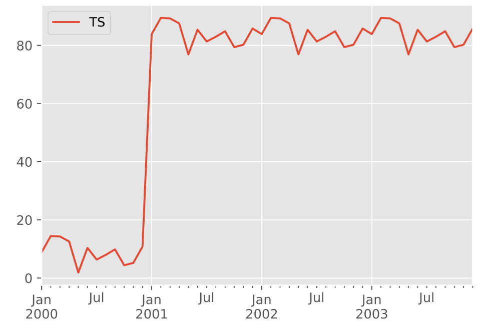
The classification of an observation as an anomaly depends on its definition as well as a given context. In real-world time series datasets outliers arise as consequence of a temporal or continuous anomalous behavior of a system. The causes can be human error, sensor failure, strange traffic patterns as sign of intrusion, attacks, fraudulent behaviour or simply natural deviations in observations. While there are lots of anomaly types, the focus of interest is on the most common ones such as unexpected spikes, drops, trend changes and level shifts. Anomalies in time series data can be classified into four different types [5–7]. Figure 3.1 graphically represents the four types of anomalies and an example how values are affected by outliers is shown in Table 3.1.



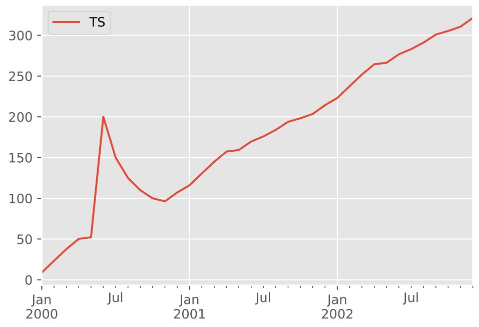
(a) Additive outlier



(b) Innovative outlier



(c) Level shift



(d) Temporary change

Figure 3.1: Types of Time Series Anomalies

- **Additive Outlier**: The value of a single observation is affected and anomalous as it is out of range from the normal data. After the anomaly occurred, the value of the next observation is within normal behavior. For example, an additive outlier may be the result of a single recording failure of a sensor due to a voltage spike.
- **Innovative Outlier**: A trend is induced by a random process that causes an unusual innovation in time series data affecting not only the level of the observations at the time when the outlier occurs but also subsequent observations, i.e, due to an unusual increasing growth of network traffic.

- **Level Shift**: The time series experiences a temporal level change of the data. The values of later observations stay either permanently or temporal on a different level at a similar extent. If activation of a related subsystem influences a main system with a constant amount, this type of anomaly can be observed.
- **Temporary Change**: An outlier that initially produces an extreme high or low value, then the size of the deviation reduces gradually until the effect dies out. Finally, the time series returns to the initial level. This behavior can also be considered as a sequence of additive outliers. For example, a voltage spike occurs which leads to an additive outlier with a fade-out effect after the initial spike.

Timestamp	Original Value	Additive Outlier	Innovative Outlier	Level Shift	Temporary Change
1	6	6	6	6	6
2	2	2	**4.48**	2	2
3	7	7	**9.48**	7	7
4	8	**10.48**	**10.48**	**9**	**10.48**
5	9	9	**11.48**	**11**	**9.32**
6	5	5	**7.48**	**8**	**8.16**
7	7	7	7	10	7

Table 3.1: Example of Outlier Values

When anomaly detection is performed, the output of the detection algorithm can be either a score or a label [4, 8]:

- **Outlier scores**: Scoring techniques compute a score value for each data instance that signs to which extent it can be considered as an anomaly. All data values can be ranked by their score in order of their tendency of being an outlier. The decision which values are threatened as outliers is based on either a domain-specific threshold defining all values above or below as anomalous or by taking the top n values out from the ranked list.
- **Binary labels**: This technique uses algorithms that classify each data point with a binary label marking each point as an outlier or not. Some algorithms return directly a binary label, others compute a score value first and mark based on a threshold value. Binary labeling provides less information if the score value is not presented as well.

3.3 Anomaly Detection

3.3.1 Categories

Based on the availability of labels for normal data and outliers, anomaly detection techniques can operate in one of three different modes [4].

Supervised Anomaly Detection Supervised anomaly detection techniques require the knowledge of both normal and anomaly class. Training datasets in which each data instance is marked as either normal or anomaly are used to build a predictive model that characterizes anomalous as well as normal patterns. Afterwards, a classifier applying the predictive model is performed on the detection datasets. This process is illustrated in Figure 3.2 where green points represent normal data and red points the anomalies.

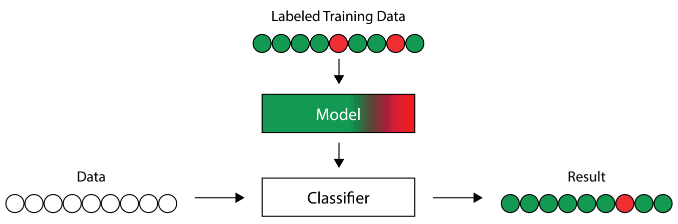


Figure 3.2: Supervised Anomaly Detection

Semi-Supervised Anomaly Detection Unlike supervised techniques, semi-supervised anomaly detection, as shown in Figure 3.3, requires only the knowledge of normal class whereby training dataset solely contain normal data without any anomalies. The idea behind this approach is to learn a model of the normal class and to detect outliers as they deviate from this model.

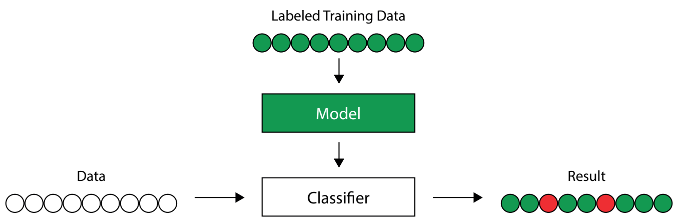


Figure 3.3: Semi-Supervised Anomaly Detection

Unsupervised Anomaly Detection The unsupervised anomaly detection approach as shown in Figure 3.4 does not involve training for the creation of a model. Instead, detection can be directly applied on any unlabeled dataset. Because in many application domains it can be difficult or even impossible to acquire representative datasets with correctly labeled data instances, this approach is widely employed. An unsupervised anomaly detection algorithm computes scores based on intrinsic properties of the dataset. Typically, geometrical properties of the data such as distances, densities or clusters between data instances are used to estimate a pattern of normal or anomalous behavior. However, this approach assumes that normal data occurs far more frequently in the dataset than outliers. If this assumption does not hold, the result of recognition is characterized by many false classifications.

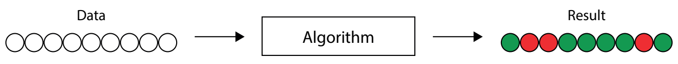


Figure 3.4: Unsupervised Anomaly Detection

3.3.2 Techniques

Local Indicators for Spatial Association (LISA) The Local Indicators of Spatial Association (LISA) [9] is a statistical technique that measures the degree of spatial correlation at specific locations. It can be directly applied to the data without additional training. LISA satisfies two requirements:

- The LISA for each observation indicates the extent to which there is significant spatial clustering of similar values around that observation; and

- The sum of the LISAs for all observations is proportional to the global indicator of spatial association.

The LISA statistics serve two purposes. They can be interpreted as indicators for local clusters of high or low values. They can also be used to assess the influence of individual locations on the magnitude of the global statistic which enables to identify outliers. This type of statistics examines the local level of spatial correlation and classifies areas where values are extreme and detects localities exhibiting similar values that do not follow the global trend. Additionally, the LISA value is an indicator of the extent to which the value of an observation is similar or different from its neighboring observations.

Consider a set of m univariate time series $\mathbf{X} = \{X_1, \dots, X_m\}$ where the p-th time series is a set of n temporal values v_{pi} ordered with respect to their timestamps t_{pi} and defined as $X_p = \{(t_{p1}, v_{p1}), \dots, (t_{pn}, v_{pn})\}$. Assume $z_{pi} = \frac{v_{pi} - \bar{v}_i}{\sigma_i}$, the LISA of a value v_{pi} is defined as:

$$L(v_{pi}) = z_{pi} \cdot \sum_{q=1, q \neq p}^{k \leq m} \omega_{pq_i} \cdot z_{qi} \tag{3.1}$$

where k is the number of all v_{*i} values at position i, $\bar{v}_i$ is the mean value and σ_i represents the standard deviation. The similarity weight between two time series X_p and X_q at index i is expressed as ω_{pq_i}.

Algorithm 1: LISA

```
Input  : ${X_1, ..., X_m}$: a set of $m$ univariate time series where $X_p = [v_{p1}, \dots, v_{pn}]$ is a series
         of $n$ temporal ordered values $v_{pi}$,
         $p \in \{1, \dots, m\}$: the index of the time series to perform anomaly detection,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_1, l_2, \dots, l_n]$: a series of predictive labels $l_i \in [0, 1]$ for $X_p$
1 for $i = 1$ to $n$ do
2     if $L(v_{pi}) < \delta$ then
3         $l_i := 1$;
4     else
5         $l_i := 0$;
6     end
7     $L := l_i$;
8 end
9 return $L$;
```

Algorithm 1 shows the pseudocode of detection with LISA classifying $L(v_{pi})$ values below the threshold δ as outliers. The threshold is usually close to zero but can be altered to a more suitable value which depends on the data. If the threshold value is too low, many outliers are not detected. On the other hand, if its too high many normal values may be classified as outliers. Further, the LISA values are highly impacted by the definition of weights w_{pq_i} among time series. Thus, the choice of weights is a pivotal step for detection and there exist different ways to assign weights depending on the research question as well as the nature of the spatial relation. For each point in a time series, its LISA score indicates the similarity to the values of correlated time series. It is worth noticing that the higher $|L(v_{pi})|$ is, the more dissimilar is v_{pi} compared to its k correlated time series. For strong negative LISA values it cannot be determined if an outlier is an anomalous low or high value compared to the values of its correlated time series. The same problem arises for positive LISA values as it cannot determine if a cluster of high or low values is present. Table 3.2 describes the possible ranges of LISA values.

Scenario	LISA value	Description
High-High	$L(v_{pi}) > 0$	Locations with high values with similar neighbours, also known as hot spots or clusters of high values.
Low-Low	$L(v_{pi}) > 0$	Locations with low values with similar neighbours, also known as cold spots or clusters of low values.
Low-High	$L(v_{pi}) < 0$	Locations with low values with high-value neighbours. A potential outlier occurred.
High-Low	$L(v_{pi}) < 0$	Locations with high values with low-value neighbours. A potential outlier occurred.
Not Significant	$L(v_{pi}) \approx 0$	Locations with no significant local correlation.

Table 3.2: Significance of LISA values

Histogram The concept of histogram-based outlier detection [10] involves the creation of a histogram for each feature in the dataset. This statistical technique uses those histograms to model the pattern of normal data and assumes independence of features, which benefits the processing speed for this algorithm at the cost of less precision. First, normal training data is used to model one histogram for each feature in the dataset. The height of a bin in a histogram corresponds to the number of observations that fall within the bins value range. Histograms need to be normalized to ensure equal weighting of features for the outlier score as well as to achieve a good way to compare histograms of different sample sizes for a fixed value range. In this context, normalization is done to keep the relative contribution of histogram bins regardless of their absolute contribution. Probability density function (PDF) has been commonly used to normalize histograms. This function specifies how the probability density is distributed over the range of values that a random variable can take. Therefore, the height of a bin is the value of the PDF at the bin, such that the total area of a histogram is always normalized to one. At evaluation, the detection algorithm determines for each feature of a data instance in which bin of its histogram the feature's values fall in. The combined height of these bins produces a final score for each data instance. In case of a low final score, the algorithm assumes that there are few or no normal data values present and scores them as outliers. Algorithm 2 illustrates how detection with this technique is performed as pseudo code.

The number of bins used to model the histogram as well as the size of the value range which they cover is essential for the detection performance. If few bins are used each covering a large value range, then many anomalous values will fall in such frequent bins yielding an increased false negative rate. In the case where many bins are used where each bin will cover a small value range, then many normal data instances will fall in empty or rare bins yielding an increased false positive rate. Hence, the optimal size of the bins to sustain low false positive and false negative rates is fundamental for the performance of histogram based detection. The square root of the number of data instances has been often proposed to optimally set the number of bins. Further, the histogram-based technique does not capture any relations between variables.

The values of a data instance may be frequent on their own, but in combination they might not be.

Algorithm 2: Histogram

```
Input  : $X = [x_1, x_2, \dots, x_n]$: a training dataset of length $n$ containing normal data instances of
         $k$-dimensional values $x_i = (x_{i_1}, \dots, x_{i_k})$,
         $Y = [y_1, y_2, \dots, y_m]$: a validation dataset of length $m$ containing $k$-dimensional values
         $y_i = (y_{i_1}, \dots, y_{i_k})$,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_1, l_2, \dots, l_m]$: a series of predictive labels $l_i \in [0, 1]$ for $Y$
1  $num\_bin := round(\sqrt{n})$;
2  $h := train\_model(num\_bin, X)$;
3  for $j = 1$ to $m$ do
4      $s_j :=$ combined bin height of all features of $y_j$ by their corresponding histogram in $h$;
5      if $s_j < \delta$ then
6          $l_j := 1$;
7      else
8          $l_j := 0$;
9      end
10     $L := l_j$;
11 end
12 return $L$;
```

Gaussian Mixture Model (Cluster) Gaussian mixture model (GMM) is a statistical anomaly detection technique [11]. GMM assumes that all the data instances are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. A mixture model is a probabilistic model that expresses the presence of subpopulations within an overall population. Mixture models in general do not postulate the knowledge about the specific subpopulation to which an individual observation belongs to, allowing the model to learn the subpopulations automatically. This constitutes a form of unsupervised learning since subpopulation assignment is not known when similar data instances are grouped together. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians [12]. It is expected that normal data instances lie close to their cluster centroid while outliers are characterized by a significant distance. In order to perform clustering an expectation–maximization (EM) algorithm fits data instances into the p Gaussian mixture components. In case the feature distributions are well approximated by a mixture of Gaussian distributions, the process of data generation can be greatly captured. When detection is performed, the probability that a data instance belongs to each of the fitted components is computed. Although this model assumes that data originates from a mixture of Gaussian distributions which is theoretically sufficient to represent any distribution, assuming we have enough centroids, adapting too many clusters will result in over fitting. In consequence, even anomalies have a high probability under the model distribution. If anomalies form clusters by themselves, this technique will not be able to detect them. It is better not to use too many components of the Gaussian mixture and thus not to capture the distributions perfectly. The pseudo code of anomaly detection using GMM is described in Algorithm 3.

The fact that the number of Gaussian distributions in the mixture model is unknown is a drawback, since the number of Gaussian distributions to be fitted is a parameter that has to be tuned. The sensitivity to the starting location of the centroid is an additional disadvantage of this method. Multiple repetitions might be necessary to achieve acceptable results. However, if full covariance matrix for the Gaussian are used then each component has its own general covariance matrix and independently adopts any position and shape, thus correlations between features are also considered and limitations of histogram-based techniques are overcome.

Algorithm 3: Gaussian Mixture Model

```
Input  : $X = [x_1, x_2, \dots, x_n]$: a training dataset of length $n$ containing normal data instances of
         $k$-dimensional values $x_i = (x_{i_1}, \dots, x_{i_k})$,
         $Y = [y_1, y_2, \dots, y_m]$: a validation dataset of length $m$ containing $k$-dimensional values
         $y_i = (y_{i_1}, \dots, y_{i_k})$,
         $p$: number of mixture components,
         $o$: number of initializations to perform,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_1, l_2, \dots, l_m]$: a series of predictive labels $l_i \in [0, 1]$ for $Y$
1  $r := 0$;
2  while $r \neq o$ do
3      $m := train\_model(p, X)$;
4      if $m$ has the largest likelihood or lower bound so far then
5          $gmm := m$
6      end
7      $r := r + 1$;
8  end
9  for $j = 1$ to $m$ do
10     $s_j :=$ the weighted log probabilities for sample $y_j$ in $gmm$;
11     if $s_j < \delta$ then
12         $l_j := 1$;
13     else
14         $l_j := 0$;
15     end
16     $L := l_j$;
17 end
18 return $L$;
```

One-Class Support Vector Machine Another effective technique to detect outliers is One-Class Support Vector Machine[13], a variant of SVM. One-Class SVM operates as a classifier by constructing a hyperplane in a high dimensional space using training data to separate into two groups of classes. In such a classifier, the training data is presumed to belong to only one class, and the goal during training is to derive a binary function that signs data instance if they belong to the class of normal data or not. The algorithm separates all data instance in feature space F from the origin, then maximizes the distance from origin to this hyperplane. The result is a function that marks data instances as normal if they are inside the separating boundary whereas the observations outside the boundary are predicted as outlier as illustrated in Algorithm 4. Thus, normal data is given with class label $y_i = 1$ whereas outliers are predicted by $y_i = -1$ for each x_i. The hyperplane is represented in equation 3.2 and the objective function of the One-Class SVM classifier is shown in formula 3.3.

$$w^T \cdot x + b = 0 \tag{3.2}$$

with $w \in F$ as normal vector, x as input vector and $b \in R$ as its displacement.

$$\min_{w \in F, \xi_i \in R, p \in R} \frac{1}{2}\|w\|^2 + \frac{1}{vl}\sum_{i}^{l} \xi_i - p \tag{3.3}$$

subject to:

$$\begin{aligned} y_i(w \cdot x_i) &\geq p - \xi_i && \text{for all } i = 1, 2, ..., n \\ \xi_i &\geq 0 && \text{for all } i = 1, 2, ..., n \end{aligned}$$

The number of data values is given by l, the origin with p and slack variables with ξ_i. The solution to the detection problem is provided through parameter v, also known as the margin of the One-Class SVM. The value of v is always between $(0, 1)$. It sets an upper bound on the fraction of outliers and it is also a lower bound of fraction on the number of training examples used as support vectors[14]. In the semi-supervised application only normal training data is provided to fit the model of the One-Class SVM. Therefore, parameter v can be used to optimize the false detection rate. If v is increased, it is much more likely to experience an increased false detection rate.

Different types of kernel can be used in this algorithm in order to achieve the decision boundary. Most popular choices are linear, polynomial, and sigmoid as well as Gaussian Radial Base Function (RBF) which was mainly used in this thesis. The Gaussian RBF kernel is defined as:

$$K(x, x') = exp(-\frac{\|x - x'\|^2}{2 \cdot \sigma^2}) \tag{3.4}$$

where $\sigma \in R$ is a kernel parameter and $\|x - x'\|$ is the dissimilarity measure. Finally, with the kernel function for the dot-product calculations the decision function is given with equation 3.5.

$$f(x) = sgn(w \cdot x_i - p) \tag{3.5}$$

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i K(x, x_i) - p) \tag{3.6}$$

Algorithm 4: One-Class SVM

```
Input  : $X = [x_1, x_2, \dots, x_n]$: a training dataset of length $n$ containing normal data instances of
         $k$-dimensional values $x_i = (x_{i_1}, \dots, x_{i_k})$,
         $Y = [y_1, y_2, \dots, y_m]$: a validation dataset of length $m$ containing $k$-dimensional values
         $y_i = (y_{i_1}, \dots, y_{i_k})$,
         $K$: a kernel function,
         $\gamma$: the kernel coefficient,
         $v$: an upper bound on the fraction of training errors and a lower bound of the fraction of
         support vectors,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_1, l_2, \dots, l_m]$: a series of predictive labels $l_i \in [0, 1]$ for $Y$
1  $svm := train\_model(K, \gamma, v, X)$;
2  for $j = 1$ to $m$ do
3      $s_j :=$ signed distance of $y_j$ to the separating hyperplane of $svm$;
4      if $s_j < \delta$ then
5          $l_j := 1$;
6      else
7          $l_j := 0$;
8      end
9      $L := l_j$;
10 end
11 return $L$;
```

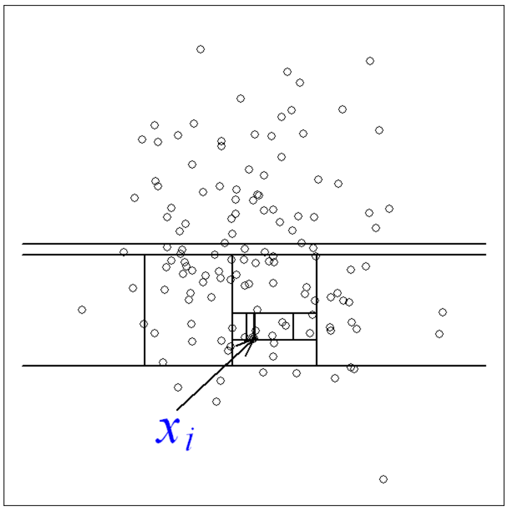
Isolation Forest Isolation forest as introduced in [15] is a tree-based model that provides an efficient way to perform outlier detection in high-dimensional datasets. The main characteristics of this method are *isolation tree*, *isolation forest* and *path length*. Given a sample of data $X = \{x_1, ..., x_n\}$ of n instances from a d-variate distribution, the isolation tree is built by recursively dividing X by random selection of a feature and a split value between the maximum and minimum values of the selected feature. The recursion stops if either:

(i) The tree reaches a height limit.

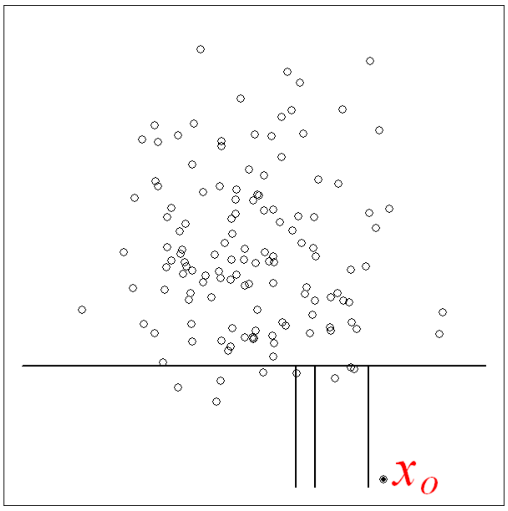
(ii) Only one value is left in the node.

(iii) All data in the node have the same values.

The term *isolation forest* is used for a group of isolation trees that have been built from the same data. Further, the path length $h(x)$ of a point x from the root node to the terminating node is equivalent to the number of splittings required to isolate x. The random partitioning results in shorter paths for outliers because the fewer observations of outliers result in a smaller number of partitions and observations with distinguishable deviation values are more likely to be separated in early partitioning. An example of the partition of a normal value and an outlier is presented in Figure 3.5.



(a) Isolating x_i: A normal point requires twelve random partitions to be isolated



(b) Isolating x_0: Four partitions are required to isolate an outlier

Figure 3.5: Isolating normal and anomalous values[15]

It is expected that the average value of the path length $h(x)$ converges if the number of trees is large enough. Thus, the average of $h(x)$ is used as an outlier score, and the smaller the average, the more likely value x is an outlier as illustrated in Algorithm 5.

Algorithm 5: Isolation Forest

```
Input  : $X = [x_1, x_2, \dots, x_n]$: a training dataset of length $n$ containing normal data instances of
         $k$-dimensional values $x_i = (x_{i_1}, \dots, x_{i_k})$,
         $Y = [y_1, y_2, \dots, y_m]$: a validation dataset of length $m$ containing $k$-dimensional values
         $y_i = (y_{i_1}, \dots, y_{i_k})$,
         $p$: number of base estimators in the ensemble,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_1, l_2, \dots, l_m]$: a series of predictive labels $l_i \in [0, 1]$ for $Y$
1  $isolation\_forest := train\_model(p, X)$;
2  for $j = 1$ to $m$ do
3      $s_j := h(y_j)$ , the mean depth of the leaf that isolates $y_j$ over all trees of $isolation\_forest$;
4      if $s_j < \delta$ then
5          $l_j := 1$;
6      else
7          $l_j := 0$;
8      end
9      $L := l_j$;
10 end
11 return $L$;
```

Robust PCA Principal Component Analysis (PCA) is a dimensionality reduction technique that computes a compact representation of a multi-dimensional dataset by reducing the number of features to a lower dimensional subspace $\Pi : \dim \Pi = n - m$. The vectors associated with the largest eigenvalues that define the hyperplane Π are termed *principal components*. The cluster X, a set of n-dimensional points x_i with $n > 1$, is defined as

$$X = \{x_i \in R^n, i = 1, \dots, N\}$$

with the subspace Π defined by m linear equations written of the form

$$Cx = b,\ C \in R^{m \times n},\ b \in R^m$$

with the rows $c_1, \dots, c_m$ of the matrix C normalized and pairwise orthogonal, then

$$CC^\top = I$$

In this case, the distance between $x_i \in X$ and Π is equal to $||Cx_i - b||$. The basis of the standard PCA approach is to solve the following optimization problem:

$$\min_{b, CC^\top = I} \sum_{i=1}^{N} ||Cx_i - b||^2 \tag{3.7}$$

PCA minimizes the L_2 norms and therefore it is highly sensitive to extreme value outliers as their occurrence can pollute PCA's normal subspace. Because of the squaring of deviations from the outliers, they dominate the total norm and steer the components. Thus, robustness of PCA to reduce this effect is highly desirable. In [16], the authors propose a Robust PCA application on Huber's approach using M-estimator loss. A robust estimate of the centre of the one-dimensional cluster $X_1 = \{x_i \in R, i = 1, \dots, N\}$ can be found by sample median which is obtained by the least absolute value method.

$$\text{Me}\, x = \operatorname*{argmin}_x \sum_{i=1}^{N} |x_i - x|$$

For points in order of value $x_1 \leq x_2 \leq \cdots \leq x_N$ with $N = 2l - 1$, we get $\text{Me}\, x = x_l$ which does not depend on extreme values x_1 and x_N. If the independent variables have identical Gaussian distributions and contain a small number of outliers the estimate has the following form:

$$x^* = \operatorname*{argmin}_x \sum_{i=1}^{N} h(|x_i - x|)$$

The *Huber function* given by $h(t)$ is defined as follows:

$$h(t) = \begin{cases} \frac{t^2}{2} & \text{for } |t| \leq \Delta \\ \Delta|t| - \frac{\Delta^2}{2} & \text{for } |t| > \Delta \end{cases} \tag{3.8}$$

Huber's approach leads to a modified version of the optimization problem from (3.7)

$$\min_{b, CC^\top = I} \sum_{i=1}^{N} h(\|Cx_i - b\|) \tag{3.9}$$

The choice of the value of the threshold Δ depends on the contamination of the data. The higher the contamination, the smaller the threshold is to be chosen. The robustness decreases by increasing the Δ value. In case it is extremely high, the outcome may be much like as in standard PCA.

In [17], PCA has been used to detect outliers which can be extended to RPCA. The procedure of outlier detection with RPCA is shown in Algorithm 6. The detection of outliers relies on the reconstruction error caused by information loss from the reduction. When transforming back from the reduced dimensions, the original values can only be approximatively recovered. Since outliers are rare and presumably different than normal data instances, anomalous data instances have higher reconstruction error as they are harder to model. In other words, pattern that occur the least often are the most anomalous. The reconstruction error will depend largely on the number of principal components respectively to the number of dimensions in the subspace. The more principal components are kept, the better RPCA will be at mapping the underlying structure of the original data. However, there is a trade-off. If too many principal components are kept, RPCA may reconstruct the original data very well and the reconstruction error will be minimal. On the other hand, if too few principal components are kept, this method may not be able to reconstruct the data to an acceptable extent.

Algorithm 6: Robust PCA

```
Input  : $X = [x_1, x_2, \dots, x_n]$: a training dataset of length $n$ containing normal data instances of
         $k$-dimensional values $x_i = (x_{i_1}, \dots, x_{i_k})$,
         $Y = [y_1, y_2, \dots, y_m]$: a validation dataset of length $m$ containing $k$-dimensional values
         $y_i = (y_{i_1}, \dots, y_{i_k})$,
         $\Delta$: threshold for Huber function,
         $p$: number of principal components,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_1, l_2, \dots, l_m]$: a series of predictive labels $l_i \in [0, 1]$ for $Y$
1  $h(t) := HuberFunction(\Delta)$;
2  $rpca := train\_model(p, h(t), X)$;
3  $Y' :=$ reduce $Y$ to dimension $p$ with $rpca$;
4  $Y'' :=$ expand $Y'$ back to dimension $k$ with $rpca$;
5  for $j = 1$ to $m$ do
6      $s_j :=$ sum of the squared differences between original $y_j$ and reconstructed $y''_j$;
7      if $s_j > \delta$ then
8          $l_j := 1$;
9      else
10         $l_j := 0$;
11     end
12     $L := l_j$;
13 end
14 return $L$;
```

4
Correlation-based LISA Anomaly Detection

This chapter introduces an extension of LISA for unsupervised anomaly detection. The extension makes use of the different types of correlations to represent different relationships across time series.

4.1 Intuition

In the case where multiple univariate time series from the same domain are used, the extent of their relationship can be expressed using correlation. It is assumed that anomalous behaviour can be derived from the discrepancy in their correlation. The time series that share a common evolution of their observations are more highly correlated than others and should therefore be more relevant for the detection. A time series observation can be anomalous with respect to other values or the mean of that time series, while having a normal behaviour with respect to other correlated time series. In this case, the deviation observed in this time series may not be anomalous as it is witnessed globally. Time series which share similar trends as the used one should be given more importance when outlier detection is performed. In real-world time series datasets, we usually do not deal with only high or low correlated time series. A wide range of different correlation values occur as correlation keeps evolving. Time series may experience similar evolution only occasionally or within limited time periods. Thus, correlation should not be derived globally but instead over a limited time period of the past using a moving frame. The weights applied on LISA computation express the strength of the relation between time series at the time frame locations. Therefore, absolute value of the correlation is used. In contrast to vanilla LISA, the use of correlation coefficients provides a benefit, since neither the locations nor the topology have to be known, and therefore yields a more general applicable approach.

4.2 Pearson-based LISA

Algorithm 7 describes the pseudo code of correlation-based LISA with Pearson. Unlike vanilla LISA, weights between time series are derived from correlation within a moving window of size w. Only the current values at each time index i and the first $w-1$ preceding values from i are taken into account to compute the correlation. Thus, the first $w-1$ LISA values of a time series cannot be calculated because it is required to have at least $w-1$ preceding values.

Algorithm 7: Pearson-based LISA

```
Input  : $\mathbf{X} = \{X_1, \dots, X_m\}$: a set of m univariate time series where $X_p = [v_{p1}, \dots, v_{pn}]$ is a
         series of n temporal ordered values $v_{pi}$,
         $w$: the length of the moving window,
         $p \in \{1, \dots, m\}$: the index of the time series to perform anomaly detection,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_w, l_{w+1}, \dots, l_n]$: a series of predictive labels $l_i \in [0, 1]$ for $X_p$
1  for $i \in \{w, w+1, \dots, n\}$ do
2      $sum := 0$;
3      $z_{pi} := \frac{v_{pi} - \bar{v}_i}{\sigma_i}$;
4      $X'_p := [v_{p(i-w+1)}, \dots, v_{pi}]$;
5      for $q = 1$ to $m \wedge q \neq p$ do
6          $X'_q := [v_{q(i-w+1)}, \dots, v_{qi}]$;
7          $\omega_{pq_i} := |corr(X'_p, X'_q)|$;
8          $z_{qi} := \frac{v_{qi} - \bar{v}_i}{\sigma_i}$;
9          $sum := sum + \omega_{pq_i} \cdot z_{qi}$;
10     end
11     $L(v_{pi}) := z_{pi} \cdot sum$;
12     if $L(v_{pi}) < \delta$ then
13         $l_i := 1$;
14     else
15         $l_i := 0$;
16     end
17     $L := l_i$;
18 end
19 return $L$;
```

Example 1. *Assume we have a set $\mathbf{X} = \{X_1, X_2, X_3\}$ of 3 time series of length $n = 7$ given by $X_1 = [4,2,2,5,9,8,3], X_2 = [10,5,4,2,6,8,9], X_3 = [6,4,10,3,1,2,8]$. At position $i = 5$, window size $w = 4$ and time series index $p = 2$, the value for v_{25} is equal to 6. We compute the LISA value $L(v_{25})$ from positions $[i-w+1, \dots, i]$:*

$$X'_1 = [v_{12}, v_{13}, v_{14}, v_{15}] = [2,2,5,9]$$
$$X'_2 = [v_{22}, v_{23}, v_{24}, v_{25}] = [5,4,2,6]$$
$$X'_3 = [v_{32}, v_{33}, v_{34}, v_{35}] = [4,10,3,1]$$

We compute z_{*i} values by mean value and standard deviation at index $i = 5$:

$$\begin{aligned} z_{15} &= \frac{v_{15} - \bar{v}_5}{\sigma_5} = \frac{9 - 5.33}{3.3} = 1.11 \\ z_{25} &= \frac{v_{25} - \bar{v}_5}{\sigma_5} = \frac{6 - 5.33}{3.3} = 0.2 \\ z_{35} &= \frac{v_{35} - \bar{v}_5}{\sigma_5} = \frac{1 - 5.33}{3.3} = -1.31 \end{aligned}$$

The weights to correlated time series X_1 and X_3 are calculated with Pearson over the window frame of w values. Small values for weights w_{21_5} and w_{23_5} mean that no or weak correlation can be observed.

$$\begin{aligned} \omega_{21_5} &= \left| \frac{cov(X_2', X_1')}{\sigma_{X_2'}\sigma_{X_1'}} \right| = \left| \frac{1.38}{1.48 \cdot 2.87} \right| = 0.32 \\ \omega_{23_5} &= \left| \frac{cov(X_2', X_3')}{\sigma_{X_2'}\sigma_{X_3'}} \right| = \left| \frac{-1.13}{1.48 \cdot 3.35} \right| = 0.23 \end{aligned}$$

Finally,

$$\begin{aligned} L(v_{25}) &= z_{25} \cdot (z_{15} \cdot \omega_{21_5} + z_{35} \cdot \omega_{23_5}) \\ &= 0.2 \cdot (1.11 \cdot 0.32 + -1.31 \cdot 0.23) & = 0.01 \end{aligned}$$

The resulting LISA value for v_{25} does not express an outlier. It is close to 0 and therefore no significant dissimilarity can be derived for this point to its correlated time series.

4.3 DTW-based LISA

In correlated time series the evolution of observations is often temporally shifted among different time series. In order to take this temporal aspect into consideration, Dynamic Time Warping can be applied before Pearson correlation is computed. This variant changes the pairing of point values grouping those pairs with lowest distance to each other as defined by the optimal warping path. The pseudo-code of DTW-based LISA is described in Algorithm 8.

Example 2. *We use the same running example as in Example 1 and we compute weights by applying Dynamic Time Warping before applying Pearson.*

$$DM(X_2', X_1') = \begin{bmatrix} 4 & 4 & 1 & 3 \\ 0 & 0 & 3 & 7 \\ 2 & 2 & 1 & 5 \\ 3 & 3 & 0 & 4 \end{bmatrix}_{w \times w} \quad DM(X_2', X_3') = \begin{bmatrix} 2 & 4 & 3 & 5 \\ 2 & 8 & 1 & 1 \\ 0 & 6 & 1 & 3 \\ 1 & 5 & 2 & 4 \end{bmatrix}_{w \times w}$$

With distance matrices for pairings (X_2', X_1') and (X_2', X_3') with values of X_1' respectively X_3' on the x-axis and X_2' on y-axis we can compute the accumulated distance matrices:

$$Acc\_DM(X_2', X_1') = \begin{bmatrix} 9 & 9 & 6 & 9 \\ 5 & 5 & 8 & 13 \\ 5 & 5 & 6 & 11 \\ 3 & 6 & 6 & 10 \end{bmatrix}_{w \times w} \quad Acc\_DM(X_2', X_3') = \begin{bmatrix} 5 & 7 & 10 & 13 \\ 3 & 9 & 8 & 8 \\ 1 & 7 & 7 & 10 \\ 1 & 6 & 8 & 12 \end{bmatrix}_{w \times w}$$

Algorithm 8: DTW-based LISA

```
Input  : X = $\{X_1, \ldots, X_m\}$: a set of m univariate time series where $X_p = [v_{p1}, \ldots, v_{pn}]$ is a
         series of $n$ temporal ordered values $v_{pi}$,
         $w$: the length of the moving window,
         $p \in \{1, \ldots, m\}$: the index of the time series to perform anomaly detection,
         $\delta$: a threshold that defines the decision boundary for outlier scores
Output : $L = [l_w, l_{w+1}, \ldots, l_n]$: a series of predictive labels $l_i \in [0, 1]$ for $X_p$
1  for $i \in \{w, w+1, \ldots, n\}$ do
2      $sum := 0$;
3      $z_{pi} := \frac{v_{pi} - \bar{v}_i}{\sigma_i}$;
4      $X'_p := [v_{p(i-w+1)}, \ldots, v_{pi}]$;
5      for $q = 1$ to $m \wedge q \neq p$ do
6          $X'_q := [v_{q(i-w+1)}, \ldots, v_{qi}]$;
7          $W_{X'_p X'_q} := DTW(X'_p, X'_q)$;
8          for $w = (j, k) \in \{W_{X'_p X'_q}\}$ do
9              $X''_p := v_{p(i-w+k)}$;
10             $X''_q := v_{q(i-w+j)}$;
11         end
12         $\omega'_{pq_i} := |corr(X''_p, X''_q)|$;
13         $z_{qi} := \frac{v_{qi} - \bar{v}_i}{\sigma_i}$;
14         $sum := sum + \omega'_{pq_i} \cdot z_{qi}$;
15     end
16     $L(v_{pi}) := z_{pi} \cdot sum$;
17     if $L(v_{pi}) < \delta$ then
18         $l_i := 1$;
19     else
20         $l_i := 0$;
21     end
22     $L := l_i$;
23 end
24 return $L$;
```

We conclude that for (X'_2, X'_1) *an optimal warping path is given with indexes* (x, y) *by* $W_{X'_2 X'_1} = \{(1,1), (1,2), (2,3), (3,4), (4,4)\}$. *For* (X'_2, X'_3) *the optimal path is no different compared to Pearson pairings as* $W_{X'_2 X'_3} = \{w_1, \dots, w_w\}$ *with* $w_i = (i,i)\, \forall\, i \in \{1, \dots, w\}$. *With warping path* $W_{X'_2 X'_1}$ *we compute new pairings for* X'_1 *and* X'_2*:*

$$\begin{aligned} X''_1 &= [v_{12}, v_{12}, v_{13}, v_{14}, v_{15}] = [2, 2, 2, 5, 9] \\ X''_2 &= [v_{22}, v_{23}, v_{24}, v_{25}, v_{25}] = [5, 4, 2, 6, 6] \end{aligned}$$

Whereas $\omega'_{23_5} = \omega_{23_5}$, *it is worth to note that each* X''_p *has to be computed for each warping path separately. Thus, the value pairings of* X''_1 *and* X''_2 *yield a different weight* ω'_{21_5}*:*

$$\omega'_{21_5} = \left| \frac{cov(X''_2, X''_1)}{\sigma_{X''_2} \sigma_{X''_1}} \right| = \left| \frac{2.8}{1.5 \cdot 2.76} \right| = 0.68$$

The weight between time series X_1 *and* X_2 *at index* $i = 5$ *for the given window frame is higher as DTW was able to adjust value pairings leading to an increased correlation. Finally, the resulting LISA value for* v_{25} *has slightly increased as well.*

$$\begin{aligned} L(v_{25}) &= z_{25} \cdot (z_{15} \cdot \omega'_{21_5} + z_{35} \cdot \omega'_{23_5}) \\ &= 0.2 \cdot (1.11 \cdot 0.68 + -1.31 \cdot 0.23) \\ &= 0.09 \end{aligned}$$

5
Empirical Evaluation

In this chapter, we evaluate anomaly detection techniques using real-world and synthetic datasets. In order to measure the effectiveness of the techniques, datasets with classified outliers are needed. The performance of the detection is evaluated using different metrics. In the following experiments, we vary different dimensions of the data to gain insights into their effects on performance. The following detection techniques, as previously introduced, are evaluated: LISA (Vanilla, Pearson-based and DTW-based LISA), Histogram, Gaussian Mixture Model (Cluster), One-Class Support Vector Machine, Isolation Forest, and Robust PCA.

5.1 Datasets

The first dataset we use is provided by Yahoo: *S5 - A Labeled Anomaly Detection Dataset* [18]. This dataset is divided into four parts termed A1, A2, A3 and A4. We selected A2 because it provides some highly correlated time series with point outliers. The A3 and A4 parts contain seasonality and changepoint information, which we do not intend to investigate, and A1 consists mostly of low correlated time series. The A2 dataset offers the following desiderata:

- No missing values.
- Time series belong to the same domain.
- Time series have same granularity.
- The data is supervised.

In addition to the Yahoo dataset, we selected two real-world datasets of temperature and humidity data from various Swiss weather stations. In these two datasets, we inject synthetically created outliers and use only a small portion of time series to make sure that the sequences are highly correlated. The number of values per time series is not extensive in A2 dataset. Thus, we selected comparable amounts of data in the real-world datasets. An overview about the used datasets for evaluation is shown in Table 5.1.

			Evaluation		Training	
ID	Source	Type	Number of TS	Values per TS	Number of TS	Values per TS
Temp1	Idaweb	Real-World	14	1000	14	1000
Hum1	Idaweb	Real-World	9	1500	9	1500
A2	Yahoo	Synthetic	10	710	10	710

Table 5.1: Description of Evaluation and Training Datasets

5.2 Performance Metrics

Accuracy, Precision, Recall, F_β-Score The anomaly detection algorithm can be seen as a binary classifier. A value is considered either as normal or anomalous. Consequently, there are four possible outcomes with respect to the actual type of a value and its prediction as illustrated in Table 5.2. The performance measures are calculated based on the number of *true positives*, *true negatives*, *false positives* and *false negatives*.

		Prediction of Model	
		Normal	Anomaly
Truth	Normal	TN	FP
	Anomaly	FN	TP

Table 5.2: Confusion Matrix

- **True Positive (TP)**: An outlier is correctly classified as an outlier.
- **True Negative (TN)**: A normal value is correctly classified as a normal value.
- **False Positives (FP)**: A normal value is falsely classified as outlier.
- **False Negatives (FN)**: An outlier is falsely classified as a normal value.

Accuracy, Precision, Recall and F_β measure are important metrics used to characterize the performance using TP, TN, FP and FN[19, 20]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Accuracy is the proportion of correctly classified results among the total number of cases examined. Accuracy alone is not a sufficient metric for evaluation as the occurrence of an anomaly can be very rare. For example, if a naive classifier marks each value as normal, an accuracy of nearly one hundred percent can be achieved in case there are very few anomalies present.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

Precision refers to exactness of the approach that gives the probability of predicting a true positive from all positive predictions.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

Recall is the proportion of correctly classified outliers based on the overall number of outliers in the dataset. Therefore, recall refers to sensitivity and is a measure for the completeness of the approach.

$$F_\beta = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall} \tag{5.4}$$

The general F_β formula for positive real β is a metric, where β is chosen such that recall is considered β times as important as precision. F_1 is the harmonic mean of precision and recall and weights them equally. When precision and recall are close the value of this metric is approximately their combined average. A reliable anomaly detection algorithm will maximize both precision and recall simultaneously. Thus, moderately good performance on both is favored over outstandingly good performance on one and terrible performance on the other.

Normalized Mutual Information Normalized Mutual Information (NMI) is a normalization of a common measure in information theory called Mutual Information (MI). MI considers the amount of information that can be extracted from a distribution of one variable with respect to the distribution of a second variable. Thus, it can be used to evaluate the quality of the detection between ground-truth labels and the prediction labels obtained by the detection method. NMI scales the results between 0 and 1 whereas 0 indicates that there is no mutual information and 1 for perfect correlation between two variables. NMI derives from entropy in information theory. For a discrete random variable X, its entropy is defined as:

$$H(X) = -\sum_{x \in X} p_X(x) \cdot \log p_X(x)$$

where p_X is the marginal probability mass function of X. Similarly, the joint entropy is defined as:

$$H(X,Y) = -\sum_{x \in X}\sum_{y \in Y} p_{(X,Y)}(x,y) \cdot \log p_{(X,Y)}(x,y)$$

where $p_{(X,Y)}$ is the joint probability mass function of X and Y. Further, the definition of conditional entropy is given with:

$$\begin{aligned} H(Y|X) &= -\sum_{x \in X} p_X(x) \cdot H(y|x) \\ &= -\sum_{x \in X}\sum_{y \in Y} p_{(X,Y)}(x,y) \cdot \log p_{Y|X}(y|x) \\ &= -\sum_{x \in X}\sum_{y \in Y} p_{(X,Y)}(x,y) \cdot \log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)}\right) \end{aligned}$$

where $p_{Y|X}$ is the probability mass function of Y conditioned on X. Based on these definitions, the mutual information $I(X,Y)$ which measures the mutual dependence of variables X and Y is defined as:

$$\begin{aligned} I(X,Y) &= H(X) - H(X|Y) \\ &= \sum_{x \in X} \sum_{y \in Y} p_{(X,Y)}(x,y) \log \left(\frac{p_{(X,Y)}(x,y)}{p_X(x) \cdot p_Y(y)} \right) \end{aligned}$$

In order to scale the range of mutual information between 0 and 1 to make interpretation more intuitive, normalized mutual information is proposed as follows:

$$NMI(X,Y) = \frac{I(X,Y)}{\sqrt{H(X) + H(Y)}} \tag{5.5}$$

Root Mean Square Error Root Mean Square Error (RMSE) is considered a common general-purpose error metric for models in predicting quantitative data. RMSE is the square root of the mean of the square of all errors between observations and defined as:

$$RMSE(X,Y) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2} \tag{5.6}$$

where $x_i \in X$ are ground-truth values, $y_i \in Y$ are predicted values and n the number of observations available for analysis.

5.3 Experiment Setup

5.3.1 Scenarios

The presence of several possibly contaminated time series within a dataset allows to infer two different types of scenarios. In an evaluation dataset either a single time series is contaminated or multiple. Further, different dimensions of datasets can be scaled to change their complexity and manner such as variation in time series length, number of time series and amount of anomalous data within the dataset.

5.3.2 Model Training

Histogram, Cluster, SVM, Isolation Forest and RPCA use a selected proportion of normal data instances in the training dataset for semi-supervised learning. The rest of the normal data as well as outlier instances will be split by the given proportion to a validation set. No data instances are shared between training and validation sets. For example, given a dataset of 1000 data instances containing 60 outlier data instances and a proportion of 0.5, then the training set will contain 470 normal data instances whereas the validation set contains 235 normal and 30 outlier instances.

The validation set is used to determine the threshold for the decision boundary that maximizes a performance metric. Outlier scores are normalized to a value range from 0 to 1. Then, performance metrics are computed for 200 linear distributed threshold candidates within this value range. Because the data is supervised the performance can be computed for each threshold candidate. Depending on the metric (NMI, RMSE, Accuracy, F1-Score, Precision or Recall) which is selected to be maximized, the most appropriate threshold is chosen.

Afterwards, the trained model is applied on the evaluation dataset. Data instances with a score below the threshold will be marked as anomalies for Histogram, Cluster, SVM and Isolation Forest whereas RPCA marks outliers if they are above the threshold. Contrary to these techniques, LISA does not require training and is directly applied to the evaluation dataset. It marks points as outliers if their score is below the threshold.

5.3.3 Parameterization

Several detection techniques offer parameters that can be adjusted to improve performance. In order to make the results comparable between different scenarios, we use the same settings for the detection algorithms in each evaluation setup as shown in Table. 5.3.

Technique	Parameter	Value
LISA (Pearson)	Window Size	10
LISA (DTW)	Window Size	10
	Distance Function	euclidean
RPCA	Delta	1
	Number of Components	2
	Training Size	0.5
Cluster	Number of Components	3
	Number of Inits	3
	Training Size	0.5
SVM	Kernel	Gaussian RBF
	γ	Scale
	v	0.95
	Training Size	0.5
Isolation Forest	Number of Estimators	40
	Training Size	0.5

Table 5.3: Configuration of Technique Parameters

5.3.4 Environment

For computation, the built-in lightweight webserver of Django running on an Ubuntu operating system is used. Algorithms are implemented in Python whereas DTW additionally used Cython generated classes in order to speed up the path finding computation. Table 5.4 shows the system specifications of the machine running the experiments.

Operating System	Environment	Processor	RAM
Linux Ubuntu 18.04	Django Webserver	hexa-core at 3.33 GHz	12 GB at 1333 MHz

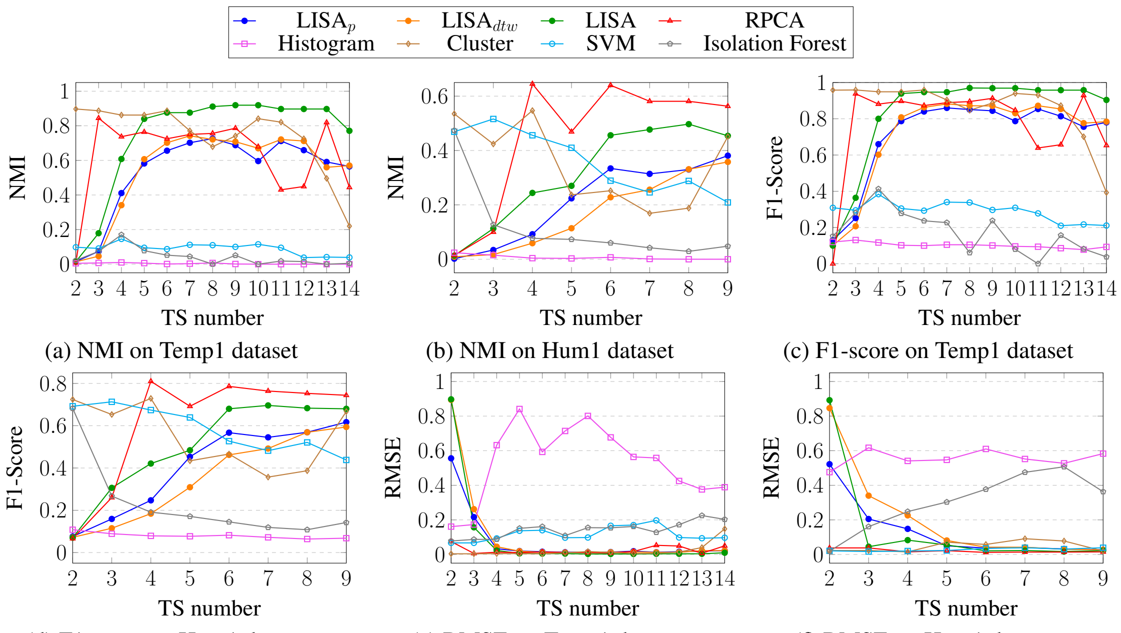
Table 5.4: System Specifications

5.4 Accuracy

5.4.1 Single Contaminated Time Series

We first consider the case where only one time series is contaminated. We use the datasets Temp1 and Hum1 with a single contaminated time series of 50 synthetically generated outliers. The same training data is used for each experiment with the same amount of outliers in only one time series and the decision boundary is set on the best F1-score.

Varying sequence number. In this experiment, we incrementally add time series to the training and evaluation dataset. At each iteration we add the next highest correlated time series available from the dataset in reference to the contaminated time series. Thus, each evaluation subset contains 50 outliers. An accurate detection is obtained by a high NMI and F1 score, and a low RMSE. The accuracy scores for datasets Temp1 and Hum1 are shown in Figure 5.1. The results show that the more accurate techniques are LISA variants, RPCA and, to a lesser extent, cluster-based Gaussian mixture model. SVM performs unsteady whereas Histogram and Isolation Forest have stochastic results.



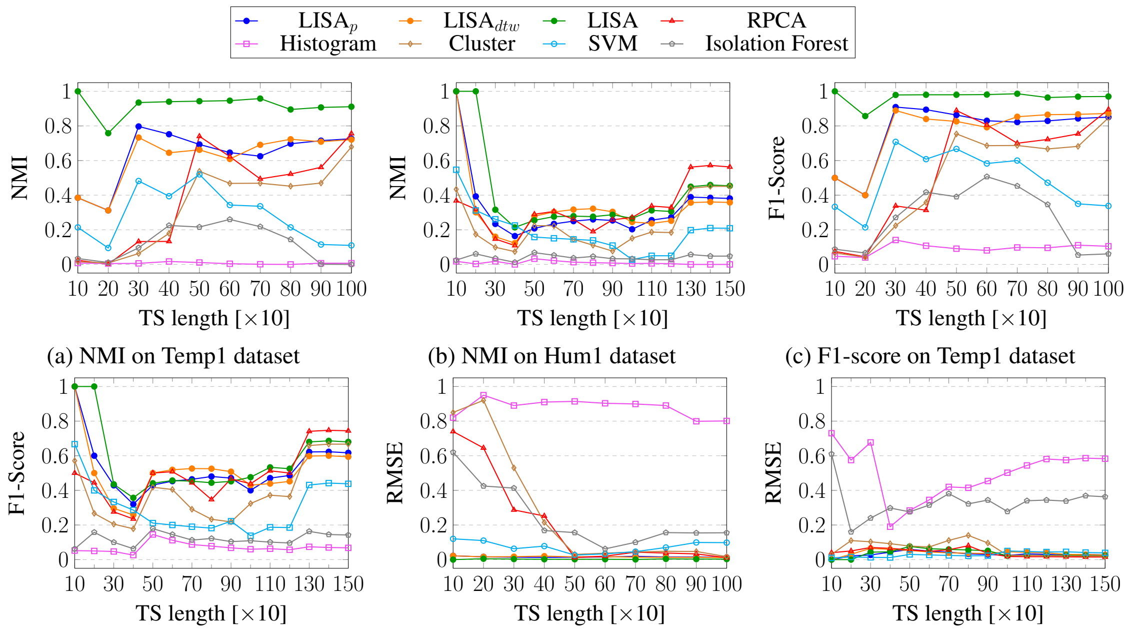
(a) NMI on Temp1 dataset (b) NMI on Hum1 dataset (c) F1-score on Temp1 dataset

(d) F1-score on Hum1 dataset (e) RMSE on Temp1 dataset (f) RMSE on Hum1 dataset

Figure 5.1: Accuracy with varying sequence number

The results on the Temp1 dataset in Figure 5.1a show that the accuracy of LISA increases with more time series until reaching 8 or 9 time series where the accuracy starts to deteriorate. On the other hand, a similar improvement for LISA can be observed in the Hum1 dataset as illustrated in Figure 5.1b, while it does not reach the same level of valid detection compared to the Temp1 dataset. The results show no significant difference between Pearson-based and DTW-based LISA. If only few time series are used, LISA classifies poorly. Other notable performance can be achieved with RPCA and cluster-based Gaussian mixture model technique. We note that RPCA cannot detect outliers if only 2 time series are used, as we use 2 principal components in the setup. Thus, no dimension reduction is performed and therefore no reconstruction error occurs. Cluster-based technique is able to classify outliers extremely well on the

Temp1 dataset, as seen in Figure 5.1c, up to 12 used time series before it starts to decrease in a large extent. In fact, as more complexity to the dataset is added the original assumption that 3 mixture components may be sufficient to capture distribution does no longer hold. We observe bad detection performance for Histogram and Isolation Forest in both datasets. Because the histogram technique does not capture relationships between variables, it is expected that this drawback is responsible for the observed results. Even outliers have values in range of normal behaviour whereas they are characterized in relation to the values of correlated time series. Therefore, Histogram is bad in detecting the occurrence of a set of values together as anomalous or not. The detection of Isolation Forest is driven by many false positive classifications that lead to poor F1-scores as seen in Figures 5.2c and 5.2d. Isolation Forest separates data instances with distinguishable attribute values more likely in early partitioning which results in shorter paths for those resulting in a classification as outlier. Therefore, some regions of normal low values in the data are falsely classified as outliers as they are not aligned with the rest of the data. SVM performs only acceptable on the Hum1 dataset if few time series are used. The results in the Temp1 dataset for SVM are driven by a region of normal low values which SVM misclassifies because they are outside the learned separating hyperplane for normal behavior. In this case, the detection could be improved by using another training dataset that captures this behavior of the data.



(a) NMI on Temp1 dataset (b) NMI on Hum1 dataset (c) F1-score on Temp1 dataset

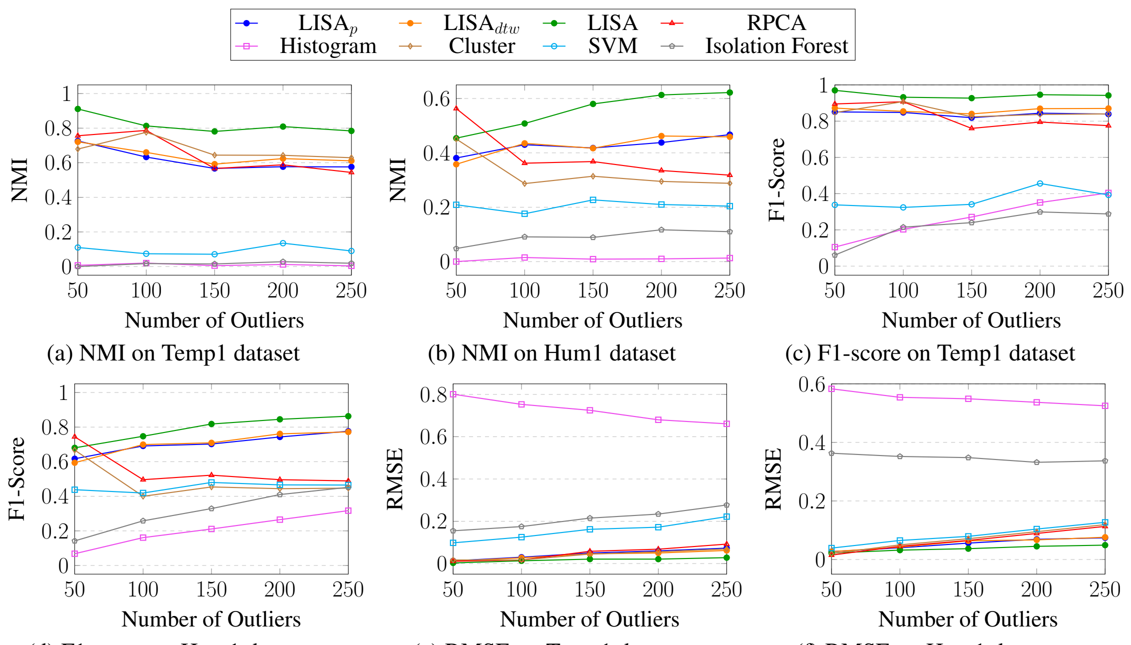
(d) F1-score on Hum1 dataset (e) RMSE on Temp1 dataset (f) RMSE on Hum1 dataset

Figure 5.2: Accuracy with varying sequence length

Varying sequence length. In this experiment, we used a subset of 8 time series for the Temp1 dataset, as we have seen in the previous experiment that this amount of data performed quite well for LISA, RPCA and Cluster. For Hum1 we use the full dataset of 9 time series. The amount of samples to evaluate in the detection dataset influences the performance. We observe different F1-scores as the number of outliers varies. The contamination rate varies between 0.02 and 0.08 for Temp1 and between 0.02 and 0.04 for Hum1. Thus, different decision boundaries are set for LISA with changing performance metrics. Further, we are witnessing altered outlier scores for same data instances at each iteration since score limits change.

This effect arise as min-max normalization of outlier scores is applied. Figure 5.2 shows the performance metrics for each evaluated time series length.

The best results are obtained with LISA and RPCA. Similar to the previous example, Histogram and Isolation Forest do not provide useful results on both datasets. For LISA, RPCA and Cluster techniques we observe a general accuracy improvement with increasing time series length whereas the scores for LISA are only influenced by the outliers which are present in evaluated subset. This is also the cause for outstanding F1-score on the shortest length of 100 timestamps in Hum1 dataset, as shown in Figure 5.3d. We note the unpreferable high RMSE scores for RPCA and Cluster in Figure 5.2e and their low NMI scores in Figure 5.2b for short lengths. They do not perform well as the amount of data is not sufficient to capture the whole behavior that was learnt from the training dataset. As mentioned before, the min-max normalization causes a different scale for outlier scores as different score limits are witnessed. In fact, the amount and behavior of training data must be aligned with the detection dataset, which is an optimization problem. On the other hand, SVM performs better but still not very persuasive for medium lengths in the Temp1 dataset than in the previous experiment as illustrated in Figure 5.2c. Indeed, the former observed areas of normal low values are not present in these subsets. The poor performance for short lengths is the result of inbalance between training and detection parts of the datasets and its impact on the outlier score normalization. By the same reason, the techniques perform worse on the Hum1 dataset.



(a) NMI on Temp1 dataset (b) NMI on Hum1 dataset (c) F1-score on Temp1 dataset

(d) F1-score on Hum1 dataset (e) RMSE on Temp1 dataset (f) RMSE on Hum1 dataset

Figure 5.3: Accuracy with varying contamination rate

Varying contamination rate. In this experiment, we use again the subset of 8 time series in the Temp1 dataset and the full dataset of Hum1. We enrich the contaminated time series by injection of additional outliers. Figure 5.3 illustrates the accuracy for different amounts of outliers. With respect to the F1-scores, we observe stable or increasing performance overall. The increasing effect results from the fact, that if the number of outliers which are present in the dataset increases and if the techniques correctly detect more outliers in similar extent in which they are added, the result is an higher F1-score. On the other hand, with increasing numbers of outliers we expect a decrease in false positive rate as such data instances may become outliers as more anomalous values are injected. It is misleading to assume that the poor performing SVM, Histogram and Isolation Forest techniques perform better with increasing numbers of outliers. We have seen in the previous experiments that their detection performance on these datasets is not good as many false positive classifications occur. As more outliers are added their F1-score rises because more outliers are classified by chance. However, their NMI score is pretty stable at a notably low level.

5.4.2 Multiple Contaminated Time Series

Next, we consider the case of multiple contaminated series. We use the Temp1 and Hum1 datasets with 4 contaminated time series each whereas the A2 dataset contains anomalies in all time series but most of them located in 4 time series. Each dataset contains 50 outliers. We discard LISA from this experiment as it is not able to handle more than one contaminated series. We also discard the NMI plots as the trend is similar to the F1-score. We analyze again the trend of performance metrics for varying number of time series, length of time series as well as number of outliers. The same training data is used for each experiment containing a total of 60 outliers in mostly 4 time series. Similar to the previous experiment, we set the decision boundary based on the best F1-score.

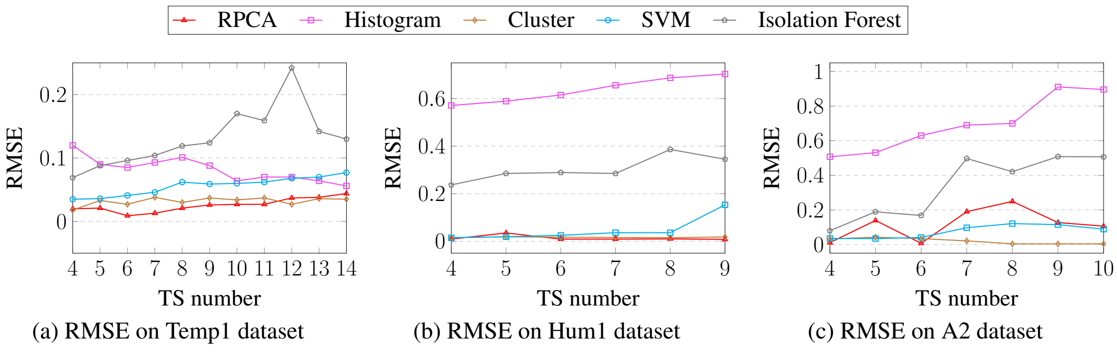


Figure 5.4: RMSE scores with varying sequence number

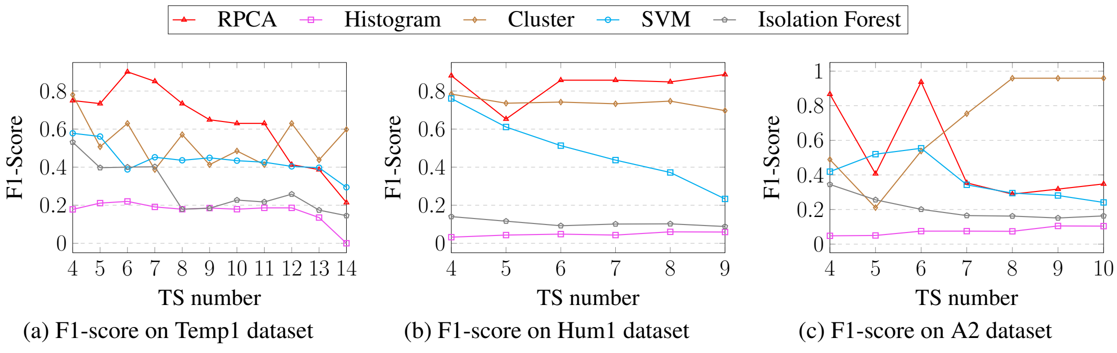
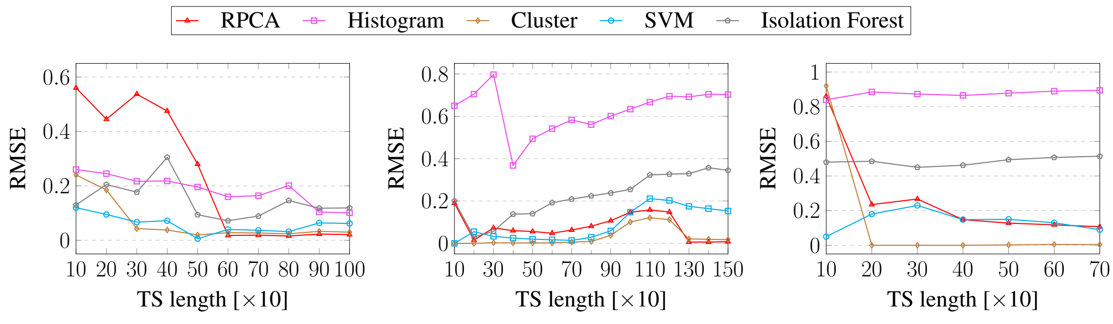


Figure 5.5: F1 scores with varying sequence number

Varying sequence number. In this experiment, we vary the number of time series which are present in training and evaluation dataset. We add the next highest correlated time series available from the dataset at each step. The RMSE scores are presented in Figure 5.4 and F1-scores in Figure 5.5. With increasing number of time series the performance of all techniques tend to decrease in most cases except for RPCA and Cluster. Again, Histogram and Isolation Forest perform worst. In Hum1 dataset, we witness that RPCA and Cluster perform best and stable even for increasing number of time series as shown in Figure 5.5b. In reference to Figure 5.7c, RPCA only achieves good F1-scores with 4 and 6 time series for A2 dataset. The unusual intermediate drop on 5 time series is a data driven effect. In this case, the added time series influences the reconstruction error by adding noise to the data which is smoothed out by adding an additional time series. Thus, we observe that RPCA can be sensitive to the amount of feature dimensions. Cluster increases rightful detection with incremented numbers of time series and performs extremely well from 8 time series and up as illustrated in Figure 5.7c. Although it is not known how the synthetically generated A2 dataset were constructed by Yahoo, this effect may be due to the fact that the data generation may be derived from or influenced by some sort of Gaussian distribution that can be captured with the Cluster technique.

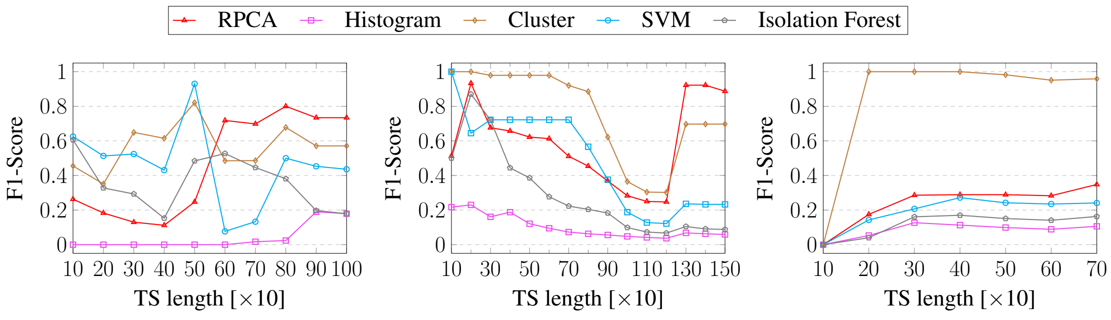


(a) RMSE on Temp1 dataset (b) RMSE on Hum1 dataset (c) RMSE on A2 dataset

Figure 5.6: RMSE scores with varying sequence length

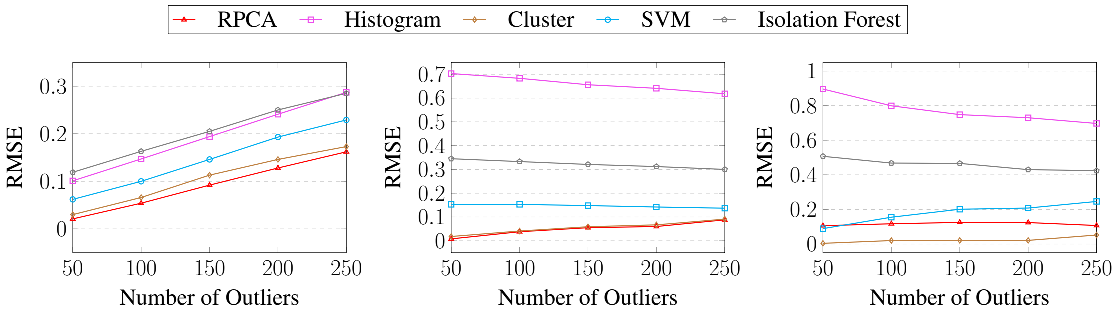
Varying sequence length. In this experiment, we used a subset of 8 time series for Temp1 and for others the complete dataset. Figure 5.6 illustrates the RMSE scores while Figure 5.7 shows the F1-scores. The best detection is achieved with RPCA and Cluster. We recapitulate that the number of samples to be evaluated in the detection dataset influences performance as the number of outliers varies and therefore different scores of performance metrics are observed. The number of samples to be classified further change the computed outlier scores since normalization of outlier scores is applied to a range from 0 to 1. Therefore we are witnessing different final outlier scores for different value limits. The results of this experiment are driven by these facts.

In A2, we see that only Cluster is able to detect outliers in an acceptable extent and we note the remarkable low RMSE scores in Figure 5.6c. As in the previous experiment, we can assume that the data is derived from a Gaussian distribution and can therefore be well captured by this technique. In Temp1, we achieve best results with RPCA for large lengths as illustrated in Figure 5.7a. The Cluster technique has mixed performance overall and achieves best results on 500 timestamps similar to SVM. In Hum1, the drop in performance as witnessed in Figure 5.7b for RPCA, Cluster and SVM is unexpected. However, with different amounts of samples we achieve different outlier score limits and therefore the effect of normalization plays a role.



(a) F1-score on Temp1 dataset (b) F1-score on Hum1 dataset (c) F1-score on A2 dataset

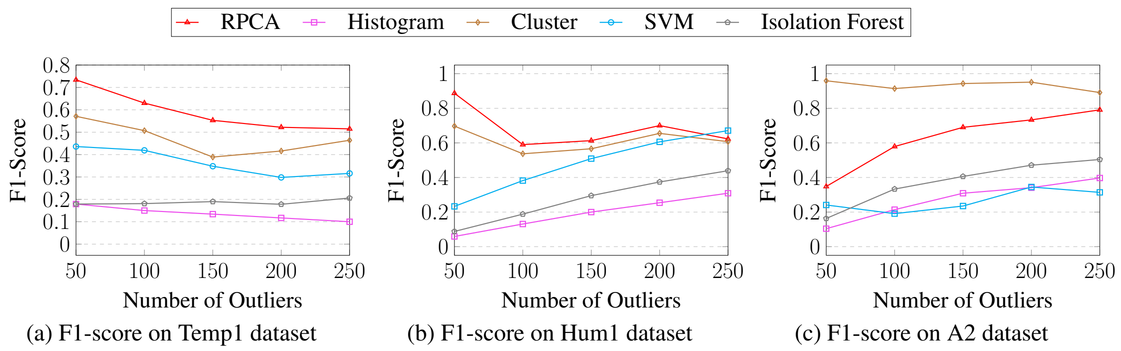
Figure 5.7: F1 scores with varying sequence length



(a) RMSE on Temp1 dataset (b) RMSE on Hum1 dataset (c) RMSE on A2 dataset

Figure 5.8: RMSE scores with varying contamination rate

Varying contamination rate. In this experiment, we use again the subset of 8 time series in the Temp1 dataset and for others the complete dataset. We enrich the 4 time series that contain outliers by injection of additional outliers and distribute them randomly. In the A2 dataset, we do the same to the 4 most contaminated time series. Figure 5.8 illustrates the RMSE scores and the F1-scores are shown in Figure 5.9. In reference to Figure 5.9a, we observe a decrease in F1-scores for incremented amounts of outliers for the Temp1 dataset. This arises as with more outliers which are present in the dataset the more false negative detections occur. This is also indicated by the increasing RMSE values as shown in Figure 5.8a. In contrast, the F1-scores for the Hum1 dataset rise for Histogram, Isolation Forest and SVM as more outliers are injected as illustrated in Figure 5.9b. In this case, the increase in correctly classified outliers is higher than the increase in falsely classified detections. For same reasons as noted in the variant of this experiment with one contaminated time series, we can state that for Histogram and Isolation Forest the increase in performance is a result of the increased presence of outliers which are much likely classified as outliers by chance. We note that Cluster still performs stable on a high level in the A2 dataset as seen by the low RMSE scores in Figure 5.8c and high F1-scores in Figure 5.9c. In A2, we also witness that RPCA increases performance with incremented amounts of outliers whereas its RMSE score stays low on a stable level. Thus, detection does indeed perform better for this technique as more outliers are present. As more outliers are available, RPCA achieves more reconstructed data instances with a remarkable reconstruction error. Thus, as more extreme values occur the min-max normalization over the reconstruction errors leads to a more suitable decision boundary.

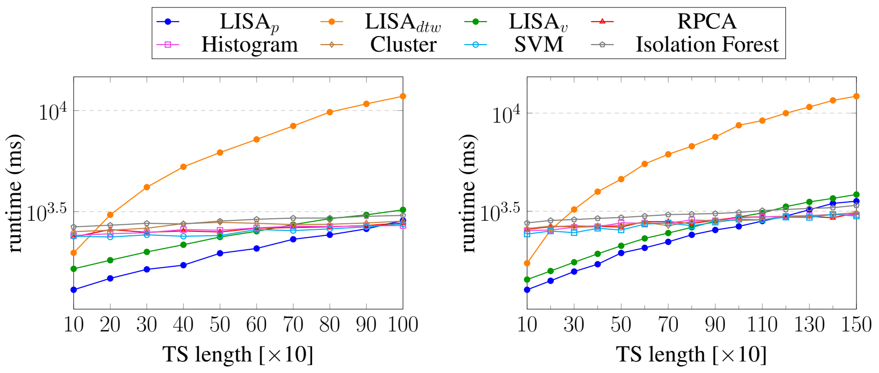


(a) F1-score on Temp1 dataset (b) F1-score on Hum1 dataset (c) F1-score on A2 dataset

Figure 5.9: F1 scores with varying contamination rate

5.5 Efficiency

In this experiment, we measure the runtime of each technique by varying the time series length and number. The runtime measurement includes the time required for model training, determination of the decision boundary by computing the performance scores for each metric and the classification of each point or data instance in the evaluation dataset. For each dimension the mean value of 7 runs is taken using the same training data. It should be noted that the number of anomalies present in the dataset has no influence on runtimes.

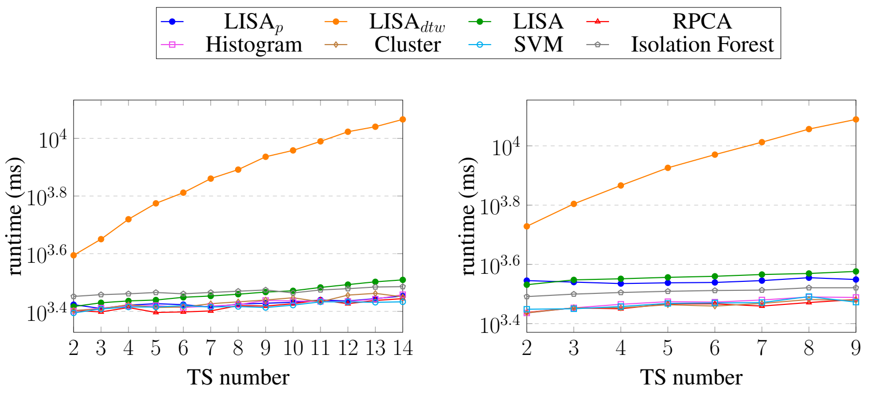


(a) Temp1 dataset ($n = 1000$). (b) Hum1 dataset ($n = 1500$).

Figure 5.10: Efficiency with increasing series length

Figure 5.10 depicts the results of varying lengths of time series (notice the y-axis log scale). The results show a linear increase of runtime with incremented lengths of time series. For short lengths, LISA performs faster than training-based techniques. This is expected as training itself requires additional time to build up the detection model whereas LISA directly performs detection. For large lengths, the training-based techniques outperform LISA. As soon as the model is built detection is fast as we used performance optimized libraries. LISA on the other hand calculates each point separately and therefore observe a linear trend with incremented lengths. It was not expected that vanilla LISA is slower than Pearson-based LISA because vanilla LISA uses same weights between each pair of time series contrarily to Pearson-based LISA which requires to compute weights separately for each pair of moving window frames. However, the effect arise from the time needed for database lookups for locations and as Pearson computation uses performance optimized libraries. We note that DTW-based LISA has the highest runtime overall. In fact, the pathfinding algorithm for DTW drives the increase in runtime because it requires accumulated cost matrices that must be calculated for each pair of moving window frames individually.

In the experiment of Figure 5.11, we measure the runtime with varying number of time series. In reference to the previous experiment with varying time series lengths, we observe similar pattern of runtime with increasing complexity whereas the number of time series inflate runtimes in a lower extent than increased lengths of time series. This effect arise as fitting a data instance into a model has higher impact on runtime from a computing perspective than the number of features present in the data instances. Indeed, for a input of n data instances with d features, the space complexity is $O(n \cdot d)$ and we witness a linear time complexity in reference to the amount of data.



(a) Temp1 dataset ($n = 1000$). (b) Hum1 dataset ($n = 1500$).

Figure 5.11: Efficiency with increasing number of time series

6
VADETIS Tool

As part of this thesis a web application termed VADETIS (Validator for Anomaly Detection in Time Series) has been implemented. The main features of this application are:

- Display real-world and synthetic time series datasets with marked outliers:
  - Search and download datasets.
  - Work with raw data as well as normalized data (Z-Score).
  - Upload time series datasets as its corresponding training data.
- Perform and evaluate different outlier detection techniques:
  - Inject different types of synthetic outliers into real-world datasets.
  - Maximize a performance metric for the detection.
  - Visualize confusion matrix and performance plots.
- Perform a recommendation for the best technique on a specific dataset using different performance metrics.

VADETIS is a client-server application that is used with a web browser. It is implemented with the web framework Django, an open source web application framework running in Python programming language and maintained by the Django Software Foundation. All data associated with the application such as accounts, settings and datasets are stored in a MySQL database. Datasets are stored as a whole in a so-called pickled object field in order to increase the performance when large datasets are requested. A pickled object field is useful for storing anything in the database when there is not a proper field for the value. This field accepts any Python object and it will automatically be converted in the background. Further, VADETIS provides a REST API for exchange of messages and data between client and server over HTTP using AJAX requests.

VADETIS consists of three main components: i) display, ii) detection, and iii) recommendation. The header bar of the page provides the navigation into the different components, to settings, into an internal account section as well as a site wide search for datasets.

6.1 Display Datasets

This component provides the display of a time series dataset as shown in Figure 6.1. The observations of the time series are presented as graph in the chart. At the bottom, meta data about the dataset and its time series is presented such as the owner of the dataset, number of normal and anomalous observations within dataset and each of its time series, level of contamination, granularity and number of available training datasets. If spatial information about the recording locations is available, a map with markers for each time series is added. The associated training datasets are listed with a link to their display page as well.

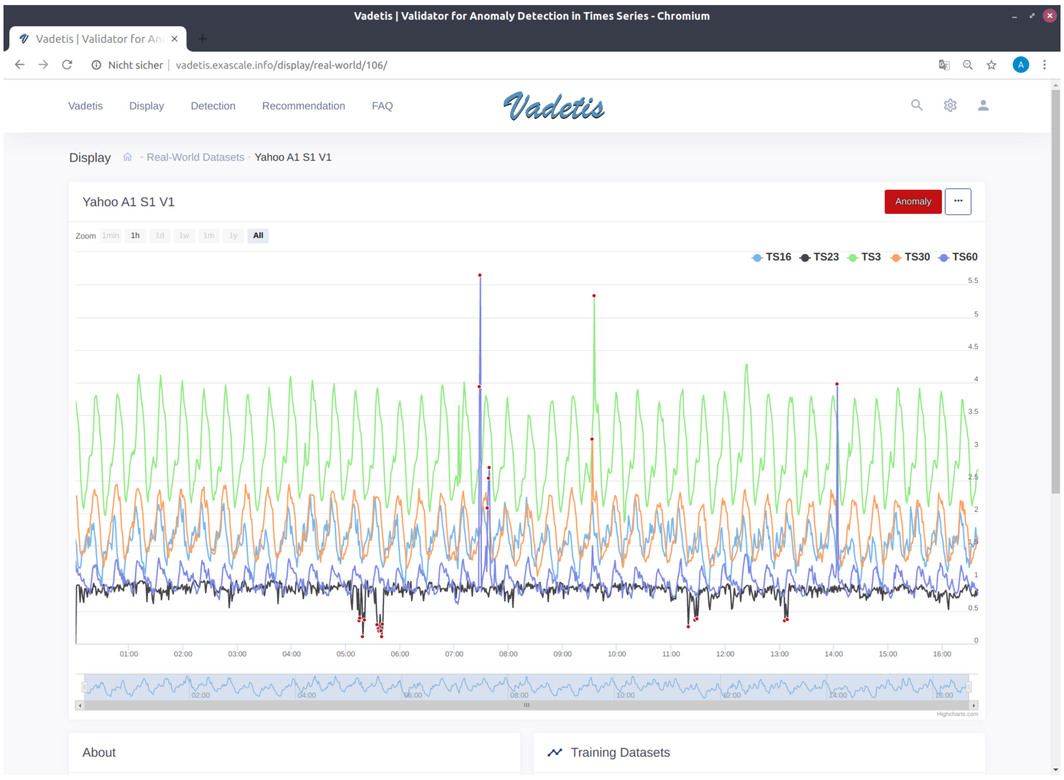


Figure 6.1: Display Time Series

When clicking on one of the time series label in the chart, the visibility of the graph of the corresponding time series is toggled. The zoom level can be selected either by clicking on one of the buttons in the upper left corner of the chart or by choosing a time range directly within the chart. At the bottom of the chart a navigator for range selection is available. On the top right corner the color legend of anomalies is shown. Right next to it, a dropdown menu to download the whole dataset as JSON or CSV formatted file or to change the representation to z-score normalized and raw data is accessible. By default, datasets are loaded with raw data.

6.2 Anomaly Detection

The anomaly detection component as presented in Figure 6.2 is the main feature of VADETIS. On the right side, a form to select the detection technique is accessible. The possibilities depend on whether training data and spatial data are available or not. After a technique is chosen, additional form fields for configuration of the techniques' parameters with preselected default settings are inserted. The detection can be performed for a selected time range or the entire dataset. The performance metric can be chosen for all techniques as part of the detection configuration. This will set the decision boundary for outlier detection on the most appropriate value. Time series may be altered by the injection of synthetically generated outliers before detection is performed.

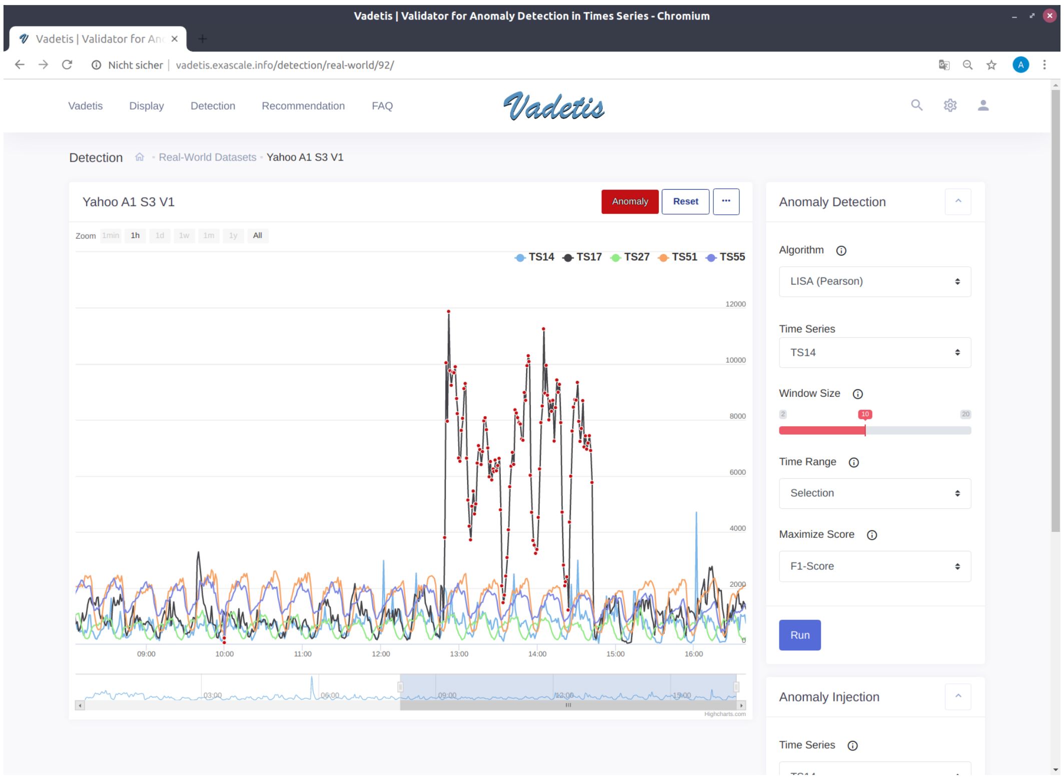
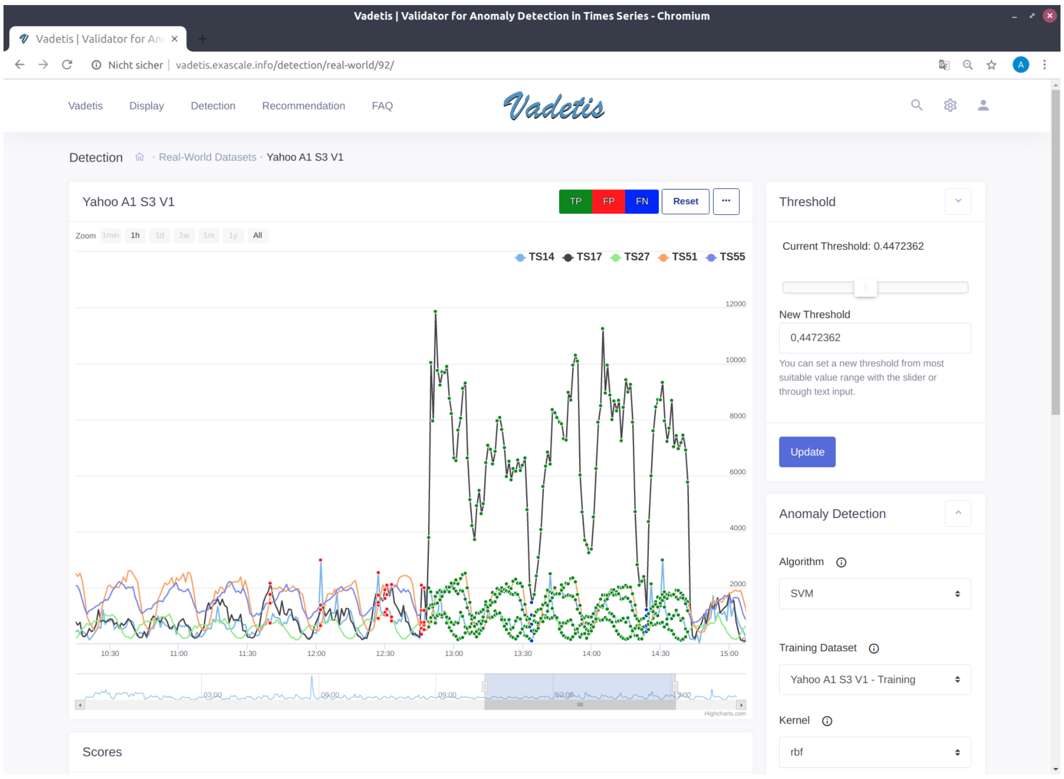


Figure 6.2: Anomaly Detection Page

VADETIS considers the different types of anomalies introduced in Chapter 3: *Point Outliers* (additive outlier), *Amplitude Shift* (level shift) and *Growth Change* (innovative outlier). In addition to these anomalies, we add *Missing Values* which sets values to zero imitating a recording failure and *Distortion* which adds a multiplied difference between two subsequent observations to the latter.

When detection is initiated the dataset from the chart is formatted into a JSON representation and sent to the server with the selected detection configuration. On the server side, the dataset is translated into object representation in order to perform detection. Afterwards a new dataset with marked points or data instances is sent back to the client. Figure 6.3 shows the page that presents the results after detection has been performed.



Threshold

Current Threshold: 0.4472362

New Threshold

0,4472362

You can set a new threshold from most suitable value range with the slider or through text input.

Update

Anomaly Detection

Algorithm: SVM

Training Dataset: Yahoo A1 S3 V1 - Training

Kernel: rbf

Figure 6.3: Anomaly Detection Results Page

All techniques, with the exception of LISA, mark data instances because they are not able to identify which individual points of data instance are anomalous. These techniques classify the collective presence of time series values at each timestamp as anomalous or normal. LISA, on the other hand, performs the detection on a single time series only and marks individual points. The color legend for points is shown in the upper right corner of the chart next to the button to reset the page. On the bottom of the results page, scores for the current performance metrics are presented in gauge clusters above a confusion matrix and threshold-score plots. The threshold-score plots as shown in Figure 6.4 present the evolution of performance metrics on the threshold value range.

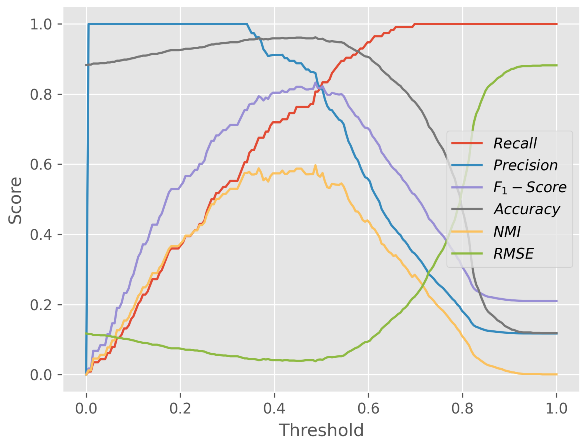
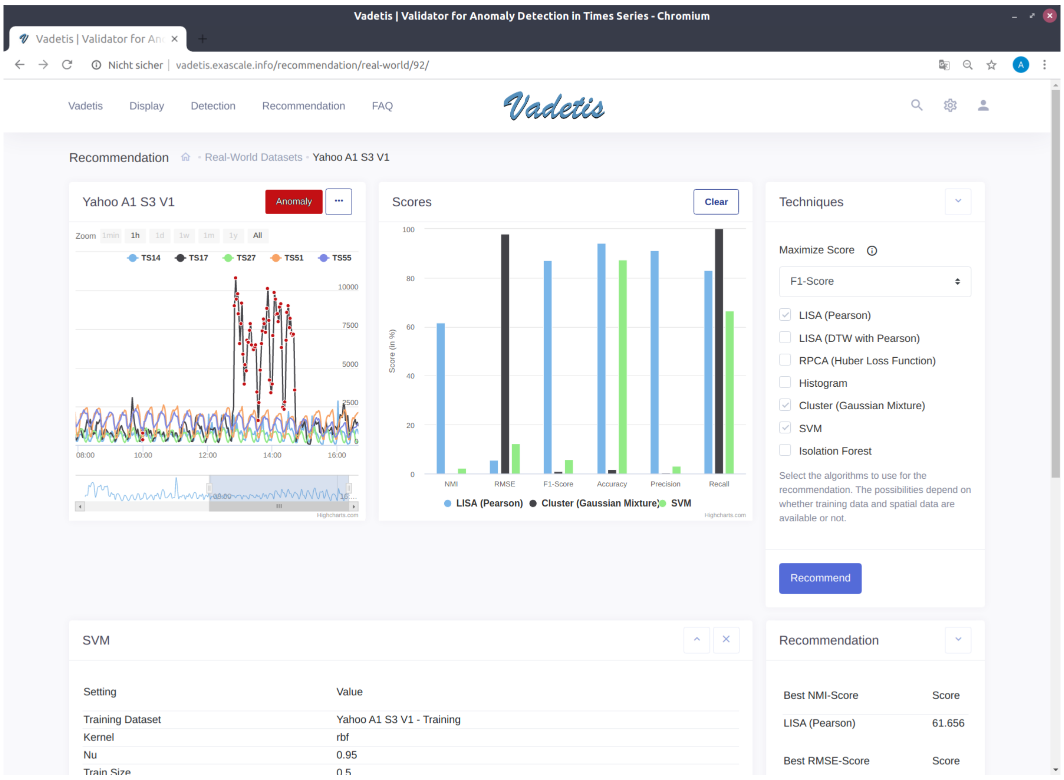


Figure 6.4: Threshold / Score Plot

Techniques that require training determine the optimal threshold value for the decision boundary automatically from the training data whereas LISA uses the evaluation dataset. Because the ground-truth of supervised data is known, the performance for each metric for each threshold candidate can be computed. VADETIS validates 200 linear distributed threshold candidates from range 0 to 1 and selects the most appropriate that maximizes the performance of the chosen metric. However, the user can further change the threshold with a slider or form field. The selection of an adjusted threshold triggers an update of chart and plots for this new decision boundary. Further detection can be performed directly from the result page without reset whereas it is only possible to inject outliers before the initial detection has been performed.

6.3 Recommendation

This component, as presented in Figure 6.5, compares the performance metrics across different techniques. On the upper left side the time series chart is shown for orientation whereas on the right side a form to request recommendations is accessible. The bar chart in the middle is divided into six groups corresponding to NMI, RMSE, Accuracy, F1-Score, Precision and Recall. It is possible to select multiple techniques together and which performance metric should be maximized. After results are received and presented it is possible to request further recommendations. This way it is possible to compare a technique with a certain maximized performance metric against other techniques with different maximized metric. In order to compute a recommendation, VADETIS uses the default configuration for each technique and automatically selects a subset of the dataset. A subset is chosen as it is computationally expensive to perform anomaly detection, especially if several techniques several techniques are performed simultaneously. When results are received a bar for each metric and requested technique is inserted or updated. The bottom of the page shows some supplementary information about each requested technique such as the parameters of the used configuration as well as confusion matrix and the threshold-score plot. A summary of the recommendations is shown in the bottom right corner, listing for each performance metric which technique works best.


Vadetis | Validator for Anomaly Detection in Times Series - Chromium
Nicht sicher | vadetis.exascale.info/recommendation/real-world/92/
Vadetis
Display
Detection
Recommendation
FAQ
Recommendation · Real-World Datasets · Yahoo A1 S3 V1
Yahoo A1 S3 V1
Anomaly
Zoom 1min 1h 1d 1w 1m 1y All
TS14 TS17 TS27 TS51 TS55
Scores
Clear
Score (in %)
NMI RMSE F1-Score Accuracy Precision Recall
LISA (Pearson) Cluster (Gaussian Mixture) SVM
Techniques
Recommend
SVM
Recommendation

Maximize Score: F1-Score

- [✓] LISA (Pearson)
- [ ] LISA (DTW with Pearson)
- [ ] RPCA (Huber Loss Function)
- [ ] Histogram
- [✓] Cluster (Gaussian Mixture)
- [✓] SVM
- [ ] Isolation Forest

Select the algorithms to use for the recommendation. The possibilities depend on whether training data and spatial data are available or not.

Setting	Value
Training Dataset	Yahoo A1 S3 V1 - Training
Kernel	rbf
Nu	0.95
Train Size	0.5

Best NMI-Score	Score
LISA (Pearson)	61.656

Best RMSE-Score	Score

Figure 6.5: Recommendation

6.4 Demo Scenarios

In this demo three scenarios are presented to illustrate the most important features of VADETIS:

Scenario 1: Anomaly detection in multiple time series. In this scenario, users can select one of the available anomaly detection techniques and define the parameters for the algorithm or proceed with the default settings. The detection will either mark anomalous data instances with training based techniques or points in single time series with LISA. The detection is initiated by clicking the "Run" button and applied either on the selected time range or complete dataset. After the results have been computed, the updated chart with marked true positives, false positives and false negatives is presented. The performance metrics, confusion matrix and threshold-score plots are loaded from the server to give insights into the conduct of the detection. If the decision boundary is not optimally set, users can adjust it (see S2 below). From this state, users are able to directly apply a different detection technique.

Scenario 2: Anomaly injection and threshold adjustment. In this scenario, users can alter time series in the dataset by injection of synthetically created outliers before detection. By clicking the Inject button the selected type of outlier is added into the chosen time series. This step can be repeated multiple times for different types of outliers and time series. Then, similar to S1, the detection is initiated by clicking the Run button. After the results have been loaded, the user adjusts the threshold for the decision boundary either with the slider or form field to a different value and clicks the "Update" button. The chart, the performance metrics, confusion matrix and plots get updated for the new decision boundary changing the markers of true positives, false positives and false negatives. By specifying a different threshold, users shall be able to observe which data instance or points are correctly classified for changing decision boundaries. From this state, users are able to repeat the adjustment of the threshold.

Scenario 3: Metric-based recommendation of the anomaly detection technique. In this scenario, users select several detection techniques which they want to compare in order to obtain a recommendation for the most appropriate method. Together with the techniques, a performance metric has to be chosen that should be maximized. By clicking the ”Recommend” button, detection with each selected technique is performed using default settings. If computation finished, the results for NMI, RMSE, F1-Score, Accuracy, Precision and Recall are compared in a bar chart for each of the selected techniques. Further, a summary which lists the best technique for each metric is presented and the confusion matrix and threshold-score plots are loaded for each of the used techniques to provide some transparency. From this state, users can request additional recommendations or adapt existing recommendations to maximize a different performance metric.

7
Conclusions and Future Work

In this thesis, we have studied different anomaly detection techniques for time series data. The main contributions of this work can be summarized as follows. First, we performed an extensive evaluation of various anomaly detection techniques using real-world and synthetic time series data. Our results show that there does not exist a single technique that outperforms in all datasets and scenarios. A proper detection can be achieved by specialization.

The second contribution of this thesis is a novel and efficient modification of LISA incorporating correlation across time series with moving window frames replacing global spatial weights of the vanilla implementation. This extension allows LISA to be performed on datasets that have no spatial relationships. Further, we extended the correlation computation with DTW. The results show no significant difference between Pearson-based and DTW-based LISA.

Finally, we have introduced a new online tool called VADETIS. It allows to display time series datasets and perform anomaly detection using various techniques and parameterization. The tool is able to present the evolution of performance metrics for changing decision boundaries. In addition, we implemented injection of synthetically computed outliers into the datasets. Because no technique performs best on every dataset, VADETIS is able to provide a recommendation which detection technique has to be used in order to achieve best results for a given performance metric.

As future work, it would be of interest to improve the efficiency of DTW, i.e., by implementing an incremental DTW approach. Although outlier detection has been studied, we did not investigate the classification of outliers into their different types and nature. In this case, the main problem to be solved is to determine the normal value that should have been observed at the locations of the outliers and to derive the type of the outliers from the deviation from this expectation. A potential solution could be to consider outliers as missing values and to estimate the normal values using a recovery technique. The automated analysis of training data can be another useful addition to the work of this thesis, as we have experienced the impact of training data on performance with regard to a possible evolution of normal behavior within the data. Last but not least, an automated way to find optimal settings for training-based techniques would be also beneficial.

Bibliography

[1] D. E. Difallah, P. Cudre-Mauroux, and S. A. McKenna, "Scalable anomaly detection for smart city infrastructure networks," *IEEE Internet Computing*, vol. 17, no. 6, pp. 39–47, 2013.

[2] I. Assent, M. Wichterich, R. Krieger, H. Kremer, and T. Seidl, "Anticipatory dtw for efficient similarity search in time series databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 826–837, 2009.

[3] G. Li, Y. Wang, M. Li, and Z. Wu, "Similarity match in time series streams under dynamic time warping distance," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 4. IEEE, 2008, pp. 399–402.

[4] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[5] R. S. Tsay, "Outliers, level shifts, and variance changes in time series," *Journal of forecasting*, vol. 7, no. 1, pp. 1–20, 1988.

[6] A. J. Fox, "Outliers in time series," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 350–363, 1972.

[7] A. E. Pankratz, D. Peña, R. S. Tsay, *et al.*, "Outliers in multivariate time series," Universidad Carlos III de Madrid. Departamento de Estadística, Tech. Rep., 1998.

[8] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer, 2017.

[9] L. Anselin, "Local indicators of spatial association—lisa," *Geographical analysis*, vol. 27, no. 2, pp. 93–115, 1995.

[10] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," *KI-2012: Poster and Demo Track*, pp. 59–63, 2012.

[11] V. Barnett and T. Lewis, "Outliers in statistical data," *osd*, 1984.

[12] S. learn developers. (2017) Gaussian mixture models. [Online]. Available: http://scikit-learn.org/stable/modules/mixture.html

[13] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, 2000, pp. 582–588.

[14] B. Lamrini, A. Gjini, S. Daudin, P. Pratmarty, F. Armando, and L. Travé-Massuyès, "Anomaly detection using similarity-based one-class svm for network traffic characterization." in *DX@ Safeprocess*, 2018.

[15] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[16] B. T. Polyak and M. V. Khlebnikov, "Principle component analysis: Robust versions," *Automation and Remote Control*, vol. 78, no. 3, pp. 490–506, 2017.

[17] A. A. Patel, *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data*. O'Reilly Media, 2019.

[18] Yahoo! (2010) Yahoo! webscope dataset ydata-labeled-time-series-anomalies-v1_0. [Online]. Available: http://labs.yahoo.com/Academic_Relations

[19] W. Chimphlee, A. H. Abdullah, M. N. M. Sap, S. Srinoy, and S. Chimphlee, "Anomaly-based intrusion detection using fuzzy rough clustering," in *Hybrid Information Technology, 2006. ICHIT'06. International Conference on*, vol. 1. IEEE, 2006, pp. 329–334.

[20] K. Limthong, "Performance of interval-based features in anomaly detection by using machine learning approach," *International Journal of Machine Learning and Computing*, vol. 4, no. 3, p. 292, 2014.