TECHNISCHE UNIVERSITÄT DRESDEN

Dep. of Computer Science Institute for System Architecture, Database Technology Group

Diploma Thesis

STREAM-BASED ANOMALY DETECTION ON TIME SERIES DATA

Josephine Rehak
Matr.-Nr.: 3962543

Supervised by:
Prof. Dr.-Ing. Wolfgang Lehner

and:
Dr.-Ing. Maik Thiele
Dr.-Ing. Claudio Hartmann

Submitted on 26 November 2019

ACKNOWLEDGEMENTS

Hereby, I would like to thank Dr. Claudio Hartmann in particular for teaching and guiding me. This was probably the most educational semester of my studies. Thank you for the nice time. Further on, I thank Prof. Dr. Lehner for the opportunity to write this diploma thesis about this exciting topic at his chair. Also, my thanks to Lucas Woltmann for being open for questions at unusual times. Without all of you the work in this form would not have been possible.

CONFIRMATION

I confirm that I independently prepared the thesis and that I used only the references and auxiliary means indicated in the thesis.

Dresden, 26 November 2019

ABSTRACT

Nowadays, servers are ubiquitous and indispensable. Unpredictable load peaks can affect whole server systems, cause server failures and thus can lead to expensive service losses. An early detection can reduce and avert damage, hence the focus is on finding a good fitting detection method. However, approach, many detection methods from various method families come into question. We present these families and methods in detail and examine multiple stream window techniques for minutely estimations. Further on, we construct a testing benchmark to simulate a stream-based scenario to evaluate these methods based on a given server monitoring data set. As a result, we receive a method which best detects abnormal server loads, thus server monitoring is improved and downtimes can be avoided.

CONTENTS

GLOSSARY

anomaly Anomalies are data points, which show unexpected behavior in context of parts or a complete data set. 13

anomaly detection method A method which employs techniques from various different method families, to divide data points into normals and anomalies. 13

anomaly detection model A model which takes data as input, employs a detection method and delivers scores or labels as output. 14

data stream A data stream is a continuous sequence of data, which is always incomplete. 13

datadriven The method depends on the characteristics of the data. 19

decomposition The splitting of a time series in season, trend, and remainder. 35

gold standard The closest approach to the basic truth, which divides the data points into normals and anomalies. 13

multivariate The data includes multiple variables. 18

normal A data point which fits in the overall pattern of the data. 15

parameterdriven The method depends solely on given parameters. 19

semi-supervised The method uses labeled as well as unlabeled data for training. 19

supervised The method requires labels for training to make predictions on the data. 19

time series A continuous sequence of points in time. 13

univariate A collection of data, which includes only one variable. 18

unsupervised The method requires no labels for training to make predictions on the data. 19

1 INTRODUCTION

The main task of this thesis is to find the best of the many possible anomaly detection methods for a given time series data set. For this, we will prepare an overview of anomaly detection methods published in scientific literature and then apply those to data given by SQL Projekt Ag. We created a benchmark, which will measure the prediction performance using a gold standard.
The data in use was created through database server monitoring and includes cpu, cache, and machine state data. We search for a sufficient anomaly detection method, which can estimate anomalous server states. This way, a warning system can be created that could avoid denial of service, reduce future downtimes, and thus reduce customer complaints and costs. In this chapter we will briefly show the background of anomaly detection, including a brief introduction to its peculiarities and main challenges, to then give details about the structure of this thesis.

1.1 BACKGROUND

Anomaly detection is a topic of ongoing high interest. It is applied in domains containing data streams, high-dimensional data, uncertain data, network data, graph data, distributed data and, as in our case, on time series data [GGAH14]. Areas of application include fraud detection [ESK02, EAP$^+$02], server system monitoring[NKSH09], intrusion detection [WFP99], software fault detection [RWHW09], production surveillance [GMESK99, She31], structure damage detection [SW$^+$98], and medical disease detection [LJK$^+$00].
It has been applied on medical data by [LW91] for finding abnormal absorption rates of drugs. While [WYK$^+$19] used anomaly detection on visual data to detect cracks in concrete. [KHM$^+$18] used it on ECG data to detect anomalies in the human heartbeat and could even diagnose cardiovascular diseases, one of the leading causes of death in the world. In a nutshell, the domains of application and the applied methods are numerous. Anomaly detection is known to allow early treatment of all kinds of anomalies, thus costs and damages can be avoided.
Anomaly detection does not consist of just one universal method. The large field of application domains leads to a vast number of anomaly detection techniques, which have varying strength for every application. For example, anomalies can be detected by the use of Nearest Neighbor methods, histogram methods or even neural networks. Where high dimensional data is the weakness

of nearest neighbor-based techniques, it is the strength histogram techniques as HBOS [GD12]. Commonly different methods are tried out, to obtain the detection technique with the best detection performance [GGAH14].
One of the challenges of anomaly detection is the sparseness of anomalies. As they commonly form a minority in the data, special measurement methods are required to evaluate the performance to detect them.
The other challenge has arisen over the years. As data is nowadays continuously collected, the underlying data is not or will never be complete. The anomaly detection process often has to run in an online fashion and close to real-time. This made common methods using labeled data unaffordable.
In this thesis, we will give an overview on various methods published in literature to tackle these hindrances. Two measurement techniques will be discussed, as well as multiple sliding-window techniques for stream-based anomaly detection.

1.2 STRUCTURE

The thesis will begin with an introduction to the basics of anomaly detection in Chapter 2. Here we address the different types of anomalies, explain the construction of static and stream-based anomaly detection models, discuss the associated challenges, and explain stream-based window techniques as well as measurement methods, then we provide an overview of common types of detection techniques. In Chapter 3, we briefly recall the basics of treating time series and then move on to describe anomaly detection methods for time series. Chapter 4 will be used for experimentation and comparison of the introduced anomaly detection methods using the presented measurements and for the discussion of the results. Here the detection method that has worked best on the given data is determined. At last, Chapter 5 will summarize how the investigated detection methods fitted the given detection task. Briefly non-treated methods will be addressed.

2 FUNDAMENTALS OF ANOMALY DETECTION

The first Section 2.1 of this chapter introduces the definition of an anomaly and will highlight some subtypes of this term. In Section 2.2 we address the concept of an anomaly detection model, including its underlying challenges, necessary decisions for its construction, as well as common measurement methods for its output. Then in Section 2.3, we go deeper into stream-based anomaly detection, its specific detection model and its challenges. As a result, we can then in Section 2.4 introduce several viable families of detection methods for both stream-based and general anomaly detection.

2.1 ANOMALIES

The definition of an anomaly is hard to grasp, as it varies with each application domain. In general anomalies are defined as outliers, discordant observations, particularities, unwontness, contaminants or exceptions in context of a specific set of data [CBK09].
In the domain of anomaly detection it is assumed for one, that anomalies always deviate from their counterparts, called normals or inlier. It is also assumed that these points occur rarely in the data, while normals represent the largest share [GD12]. As an example, in Figure 2.1 the data points in o_1, o_2, and o_3 are anomalies as they deviate from the normal groups of data N_1 and N_2.
Causes for such anomalies can be human error (e.g. mistyping), instrumental errors, changes or shifts in data point distribution and population, changes or errors in the system (e.g. defective sensory contacts, measurement errors) [HA04, AP14].
Anomalies should not be confused with noise, as noise does not carry as much information as an anomaly. A clear distinction between noise, normal and anomalous data is most often not possible, as the transitions are fluid [AY01].
This section introduces some general types of anomalies, which have spread in literature and are widely applicable. Which anomaly type is searched depends on the detection task.

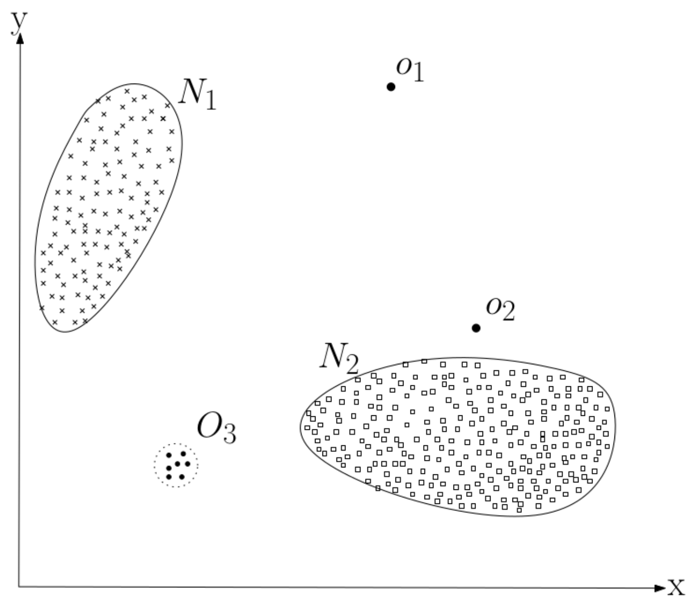


Figure 2.1: Example of Anomalies in 2-Dimensional Space [CBK09]

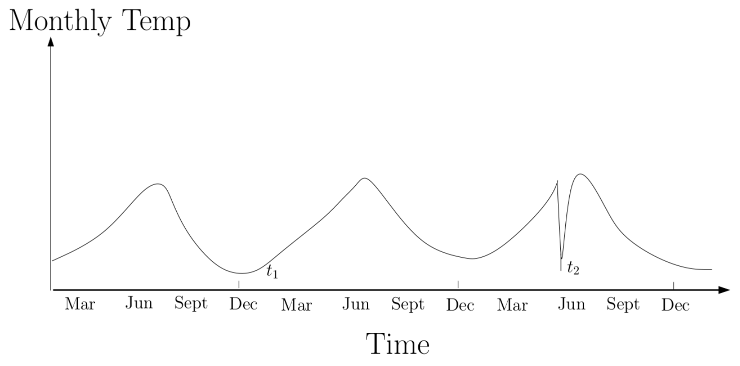


Figure 2.2: Example of a Context-based Outlier t_2 [CBK09]

Point Anomalies are the most popular type of anomalies. They are single data instances which can, in respect to the rest of the data, be considered as anomaly. In our chosen application domain, examples are sudden high loads and software failures due to programming errors, which can cause anomalies in cpu load. Examples for this category of anomalies are shown in Figure 2.1. o_1, o_2, and each point in O_3 deviate from the normal regions N_1 and N_2.

Context-based Anomalies are most popular in sequential and time series data, as underlying dependencies between the data points are required for this type of anomalies. A single data instances is marked as context-based anomaly by the context they occur in (e.g. their neighborhood) [CBK07].
As demonstrated in Figure 2.2, a context-based anomaly can be a pointbased inlier, as a point anomaly can be a context-based inlier. The data point t_2 has the same value as t_1, which makes it in context of the full data set an inlier, but in context of its neighborhood an outlier.

Collective Anomalies are collections of data instances, which deviate from the rest of the data set. Independent of each other they appear normal, but by their joint appearance they

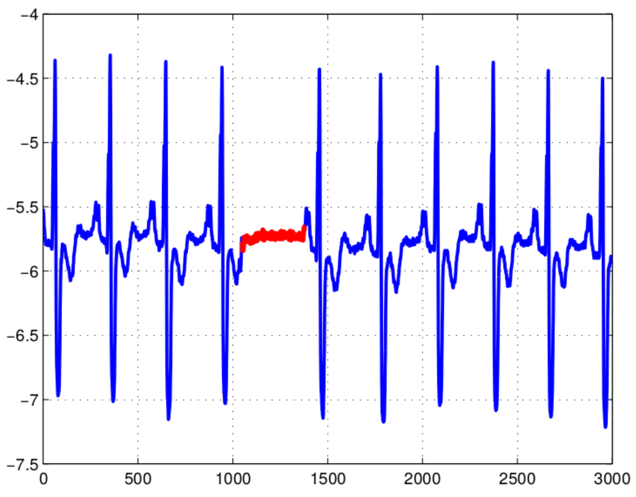


Figure 2.3: Example for a Sequence of Collective Anomalies [CBK09, GAG$^+$00]

become anomalous. As the electrocardiogram in Figure 2.3 shows, on a search for point anomalies the red marked group of collective anomalies would not be detected. However, the anomalous sequence consists of average data values, which clearly do not follow the general heartbeat pattern. Their joint appearance indicates in this specific case an Premature Atrial Contraction [CBK09]. In the domain of server monitoring, a collective anomaly could for example indicate a buffer-overflow.

2.2 ANOMALY DETECTION MODELS

An anomaly detection problem is solved by an anomaly detection model. Detection models follow the same principles, here presented as the Basic Detection Model. The main component of a model is the detection method, which depends on multiple factors listed in Section 2.2.2. For construction of such a detection model certain problems and tasks are inevitable, which are described in Section 2.2.3.

2.2.1 Basic Detection Model

In Figure 2.4, the fundamental structure of an anomaly detection model is illustrated. The central core is the anomaly detection method, also called outlier detection technique. This component results from the combination of knowledge disciplines and application domains as algorithms from machine learning, data mining, and statistics are applied on application domains as system diagnosis, computer networks, banking statistics or medical sensor diagnostics [AY01, CBK09].
The problem definition of the anomaly detection task results from the application domain. It defines which data is provided, for which kind of anomalies is searched and which kind of results are required. For this reason, an anomaly detection method is always strongly coupled to its application.
The input for the outlier detection technique originates from the application domain. It is also

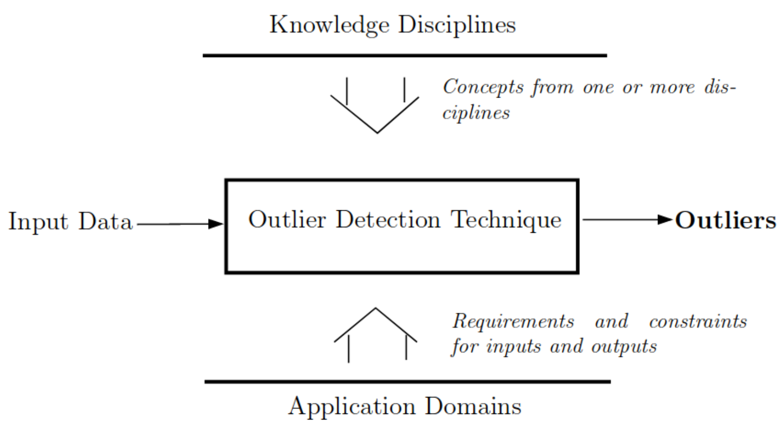

Knowledge Disciplines
Concepts from one or more disciplines
Input Data
Outlier Detection Technique
Outliers
Requirements and constraints for inputs and outputs
Application Domains

Figure 2.4: Basic Composition of an Anomaly Detection Model [CBK07]

referred to as collection of data instances, features, samples, case, entities or points. In it, the anomalies are searched using the anomaly detection method. The output of the detection technique can take two different forms: either score or label.
Scores represent the degree of outliernes for each given data instance, but do not offer any clear distinction between anomalous and normal data as boundaries are commonly fluid. But they provide the possibility of analysis by ranking and comparing anomalies by severity [AY01].
Binary labels divide the data points in the two groups of either anomalous or normal, but they provide fewer information about a points degree of outliernes than a score.
By itself, scores are the more common output, but labels are the preferred final output. For the transition from score to label, domain-specific thresholds (e.g. cut-off threshold) are applied. This results in a loss of information, but the possibility of making decisions based on the results becomes easier [AY01].

2.2.2 Fundamental Decisions on the Detection Method

For the construction of an anomaly detection model, the detection problem has to be described as detailed as possible. The domain of anomaly detection contains innumerable different facets, which sometimes exclude each other. In order to start anomaly detection, the following decisions must be made:

Fundamental Nature of Data A models anomaly detection method depends fully on the type of data. It might for example be continuous, temporal, graph, spatial and/or distributed data. Each domain has its own specialized methods and sometimes multiple such domains come into question. A decision on the most useful domain has to be made.

Data Dimensionality The method also depends on the dimensionality of the data. Data can occur in form of either multiple data variables (multivariate) or in form of one variable (univariate). Which kind is supported, depends on the detection method.

Level of Detection The level of anomaly detection in the data set should be specified. Either is the detection of a full anomalous data composite (e.g. one anomalous time series) possible or the detection of data instances within this composite (e.g. anomalous data instances within a time series).

Context Relevance The influence of the data context on the detection process varies with each detection method and allows to search for collective, but also for point anomalies.

Mathematical Model Dependency Detection Methods can impose underlying mathematical models on the data (parameterdriven) or can deduce limits and properties from underlying data and "let the data speak" (datadriven).

Existing Domain Knowledge Most anomaly detection methods use one of three fundamental methods: Some have no prior knowledge of the data, some model both anomalies and normals, some model only normals or only anomalies. The choice depends on the data which kind is most accurate [HA04].

Supervision Type The choice of a detection method strongly depends on the availability of labeled data. Detection methods can either be learning from given labeled data (supervised), not using any labeled data (unsupervised) or use a combination by using either normal labels or anomaly labels (semi-supervised). In general,labeled data is hard to obtain, as it is often manually created. Using labels also bears the problem, that not all kinds of possible anomalies are always present in a training set. Unsupervised techniques are more freely applicable, but commonly suffer from higher false-alarm rates [GGAH14].

2.2.3 Challenges of Model Construction

When constructing an anomaly detection model, several hurdles need to be tackled. They include for example the definition of normals, anomalies, and noise. They are here described in detail:

Anomaly Definition A fundamental difficulty is to define, when a data instances is an anomaly and when it is normal. Commonly no clear boundary between normal and anomaly exist.

Distinction of Noise Noise is just as difficult to filter out. As anomalies are strong outliers, noise consists of weak outliers of low interest. It intermingles with normal and anomalous data and is not clearly distinguishable [AY01].

Imbalanced Data In common scenarios anomalies are seldom and underrepresented in the training data set, which increases the inclination of the detection method to categorize anomalies as normals [CBK09].

Changing the Operational System The definition of anomalies can change over time, as the application domains keep evolving. The underlying supervised system will be subject to interference, as its surroundings change. The detection system is required to adapt to latest developments. Today's normal could be tomorrows anomaly [CBK09].

Data Selection The data set should support a good distinction between anomalies and normals by presenting the underlying causes in the best possible way. Hence data exploration and feature selection are strictly necessary[CBK07].

Parameter Choice The choice of parameters affects the performance of a detection model considerably. Finding perfect parameters is tedious work, which is why in the best case parameter-free methods are applied [KLR04].

2.2.4 Measures for Anomaly Detection

In anomaly detection, there are well-established methods to measure the quality of the detected anomalies. Statistics like the confusion matrix, as well as precision and recall are presented here, enhanced by curve analysis and the use of ROC and precision-recall curves.
The measure of accuracy is not applied in anomaly detection and is not listed here, as its results would be of little relevance in the highly unbalanced domain of anomaly detection.

Confusion Matrix

The confusion matrix provides overview of the number of anomalies detected correctly (TP) and incorrectly (FP), and of the number of normals detected correctly (TN) and incorrectly (FN) requiring a gold standard, the closest approximate to the ground truth. The common structure of the confusion matrix is shown in Section 2.2.4. Dependent on the application domain, either the False Positive Rate (FPR) or the False Negative Rate (FNR) is important for optimization. A high FPR can lead to a high false-alarm rate. While a high FNR can lead to a high rate of undetected anomalies.
The confusion matrix always stand in connection with the output of the anomaly detection model. It is easily applied on binary labeled output, but it is always coupled to the threshold applied to the scores [AY01].

Precision and Recall

Precision and recall are commonly used measures in statistics [HVK17]. As the confusion matrix, it also depends on the model output. The precision is the percentage of detected outliers in a data set, which are actual outliers. Using binary labels, the precision is calculated in the following manner:

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

In case of anomaly scores as model output with given threshold t, detected outliers $S(t)$, and provided ground truth G is the precision calculated as follows [AY01] :

$$precision(t) = 100 * \frac{|S(t) \cap G|}{|S(t)|} \tag{2.2}$$

		detected anomalies	
		normal	anomaly
actual anomalies	normal	True Negative (TN)	False Positive (FP)
	anomaly	False Negative (FN)	True Positive (TP)

Table 2.1: General Structure of a Confusion Matrix

In anomaly detection, the recall is defined as the percentage of outliers in G, which have been reported. Hence it is equal to the TPR. Using binary labels they are calculated like:

$$\begin{aligned} TPR &= recall \\ recall &= \frac{TP}{TP + FN} \end{aligned} \tag{2.3}$$

While using anomaly scores with threshold t the recall and the TPR are defined as [AY01]:

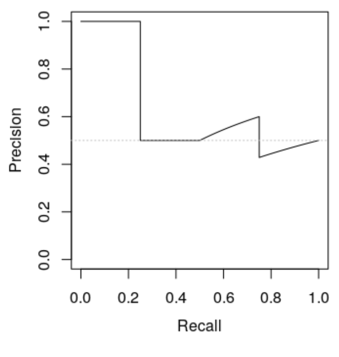
$$\begin{aligned} TPR(t) &= recall(t) \\ recall(t) &= 100 * \frac{|S(t) \cap G|}{|G|} \end{aligned} \tag{2.4}$$

Curve Analysis

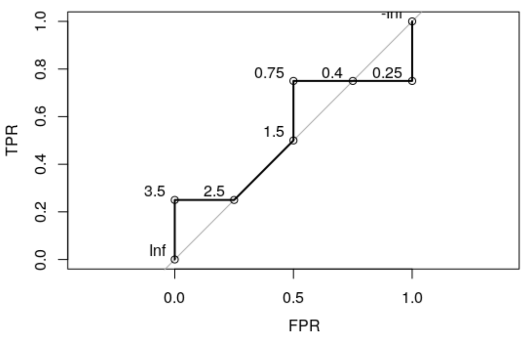
As the measures introduced above are dependent on variables in the detection model (e.g. threshold), two-dimensional plots become possible. The two most popular plots are the precision-recall curve and the Receiver Operating Characteristic curve (ROC curve). Examples of both curves are shown in Figure 2.5.

Precision-Recall Curve The shown precision-recall curve in Figure 2.5a results from the variation of the threshold value t applied on the detected anomaly scores. As with moving threshold the recall steadily increases and never descends, it is plotted on the x-axis, while the precision is plotted on the y-axis. In the best case, the curve runs close to the upper left corner, close to a precision of 100% and then moves to the right.

Receiver Operating Characteristic Curve ROC curves originate from the domain of multiclass classification and are popular for anomaly detection, as they can also tackle the separation problem between detected and undetected normals and anomalies. As seen in Figure 2.6, the detection methods tries to separate the true anomalies from the true normals by the means of scores. The ROC method tries several threshold on the scores to find the best separation. For each threshold, the TPR and FPR is measured. The result is a curve as the



(a) Precision-Recall Curve



(b) ROC Curve

Figure 2.5: Examples for a Precision-Recall Curve and a ROC Curve Based on the Same Label and Estimation

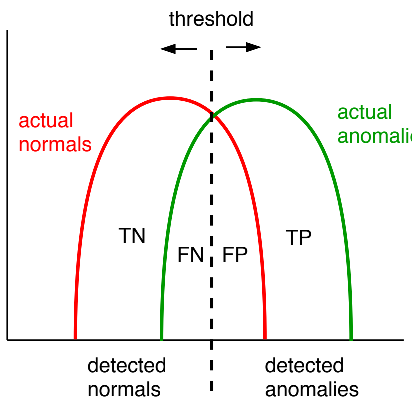


Figure 2.6: ROC Applies a Moving Threshold on the Anomaly Score Distribution

one seen in Figure 2.5b. ROC curves have FPR denoted on the x axis and the TPR denoted on the y axis. Plotting the corresponding thresholds is optional, here they are plotted on the curve. The optimal threshold has a high TPR and a low FPR, but is neither infinity (Inf) or the negative infinity (-Inf) as those thresholds can only identify one label. In the best case, the curve runs close to the upper left end of the graph.

ROC curves are usually perceived as more intuitive, as the curves can be compared more quickly and clearly. Contrary to the precision recall curve, the start point always lies at 0,0 and endpoint always at 100,100, which allows the consideration of the Area Under Curve (AUC). The AUC is a measure for the degree of seperability between normals and anomalies [AY01]. Optimal is an AUC of 100% points, as a threshold can clearly separate the scores and achieve a TPR of 100% and a FPR of 0%. An AUC of 0% indicates an inverse predictor, as the estimation always dissents with the label. An AUC of 50% indicates a bad predictor, as a meaningful score separation is not possible.

2.3 STREAM-BASED ANOMALY DETECTION

In modern times, the nature of the underlying data has changed. Nowadays, massive amounts of data are created and processed daily. Streams are data sequences, which are collected continuously and are never complete. Due to resource constraints, as limited cpu power and memory space, new priorities have to be set. For one, data instances are best treated only once to allow fast anomaly detection during runtime. Secondly, not all data instances can be kept for computation. Old data instances have to expire, while new instances are successively added to the stream. Usually sliding windows are used on the data, to keep current data in storage and to dispose old data.
In this section, we first present the basic model for such stream-based anomaly detection. Then we introduce different sliding window approaches and discuss the challenges of stream-based anomaly detection.

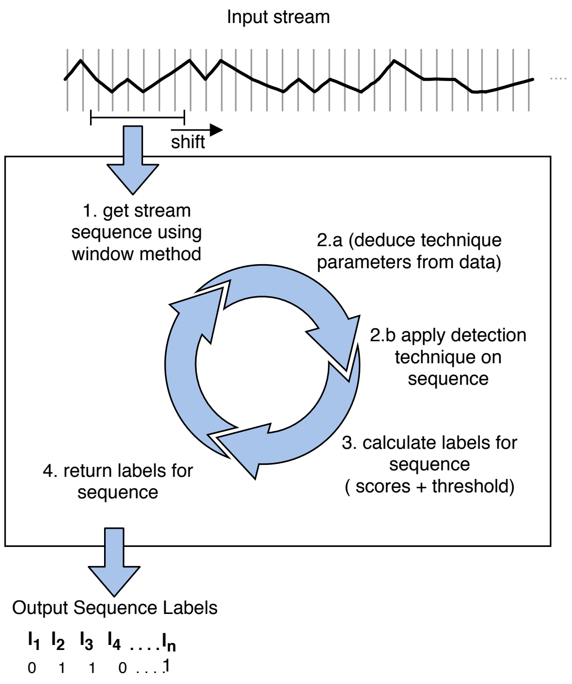

Input stream
shift
1. get stream sequence using window method
2.a (deduce technique parameters from data)
2.b apply detection technique on sequence
3. calculate labels for sequence (scores + threshold)
4. return labels for sequence
Output Sequence Labels
l1 l2 l3 l4 ....ln
0 1 1 0 ... 1

Figure 2.7: The Basic Stream-based Anomaly Detection Model

2.3.1 Basic Stream-based Anomaly Detection Model

In stream-based anomaly detection, anomalies are defined as patterns, which deviate from previous patterns in the stream [LA15]. To find such anomalies, the detection model applies a detection method on shifting stream segments. Figure 2.7 shows the resulting detection cycle as it moves over the input stream. The first step of the cycle is to get the next stream segment including the data points up for estimation and if required some reference data. In the next step, the data driven methods deduce some parameters, which are already set for parameter driven methods. In the step 2.b, the detection method is applied using these parameters. Afterwards the results from the methods are received, which can be labels or scores. On the scores thresholds are applied to receive the labels. As an output the method delivers a binary label for each input data point.
This cycle is similar to the static detection model shown in Figure 2.4. For example, the detection technique results from knowledge disciplines and application domain as well. Furthermore, the here shown steps of parameter deduction and method application with labels or scores are the same in the basic static model. The significant difference of the stream-based model is, that the output always consists of labels for the stream segment only.

2.3.2 Stream Window Methods

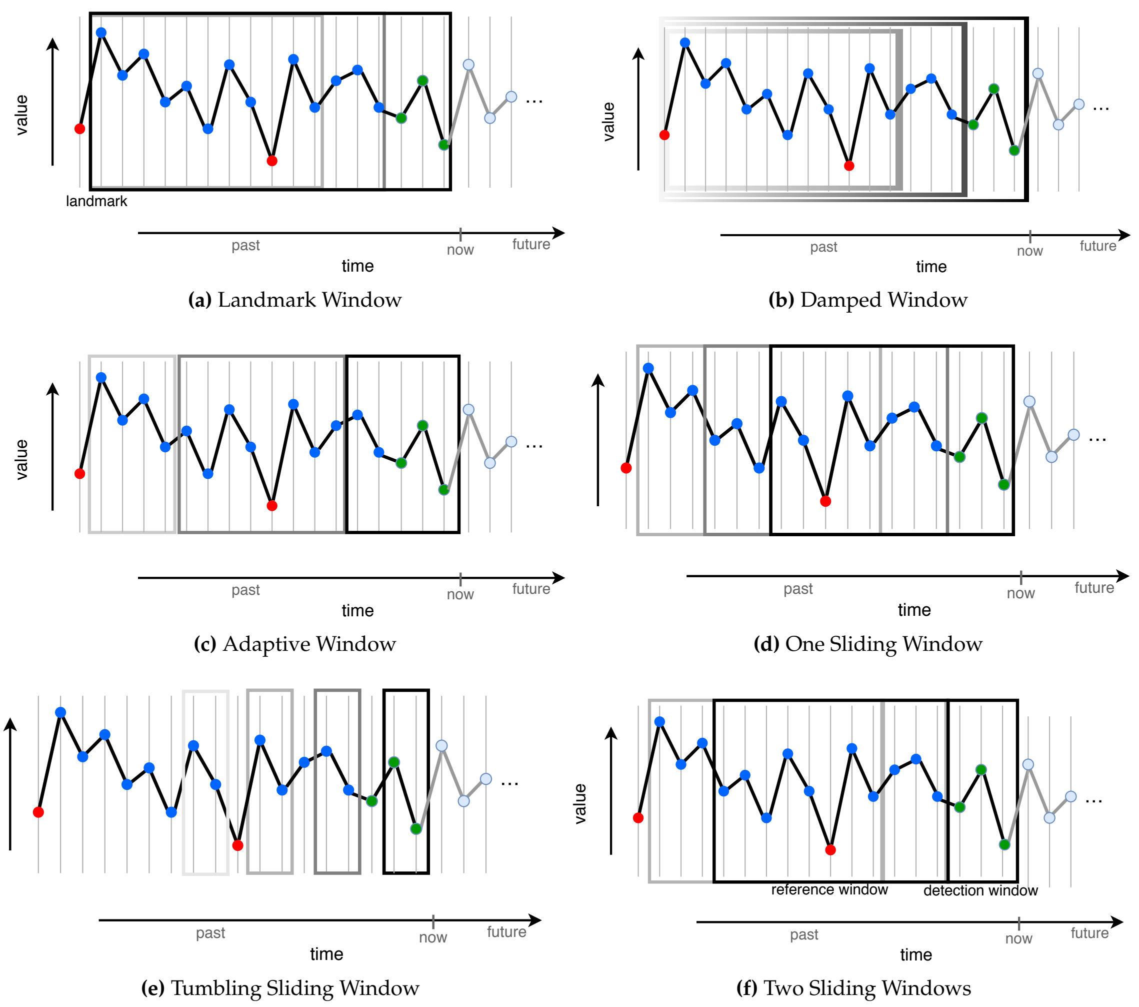
In stream-based anomaly detection, it is common to apply windows. For one, the stream requires limitation for data memory and cpu load savings. In addition, as the stream evolves, restrictions to the latest data are useful. Several such window techniques exist. Here we gathered different methods from various survey papers and publications, in specific from [AF07, SR18, GÖ03, TTL11]. As the naming was not uniform, it had to be adjusted accordingly.

In Figure 2.8, we have a visualization of each found method. In it, data points are either marked as normals (blue), anomalies (red), up for anomaly detection (green), or are upcoming values in the future (gray). Each window method settles these values differently:

Landmark Window For this method, fixed points are identified in the stream, called landmarks. The analysis is then performed on data instances between the last landmark and the current data point, hence the window is delimited by one fixed endpoint and one moving endpoint [AF07, SR18].

Damped Window This method uses weights on its data points. The older the data instance, the lower is its weight. Hence more current data instances have more influence on the calculation outcome. However, as this window method requires a supporting anomaly detection method, its application is not in all cases easily possible [SR18].

Adaptive Window While the other methods use fixed amounts of instances per window, the window size of this method is adaptive. For example, it can be coupled to the amount of fluctuations in the stream. Many fluctuations could cause more detailed examination, while few changes could cause only few estimations [SR18].



(a) Landmark Window **(b)** Damped Window **(c)** Adaptive Window **(d)** One Sliding Window **(e)** Tumbling Sliding Window **(f)** Two Sliding Windows

Figure 2.8: Variations of the Sliding Window for Stream-based Anomaly Detection

Sliding Window This window method has two moving endpoints: the last data point in the stream and the point corresponding to the window width many points before the last point. Both ends always shift with the same speed as new data instances are added, hence the count of data instances per window is constant.
For anomaly detection, the same window is used for training and for estimation. Different sizes in window are possible, too large windows can cause interesting artifacts, because as the window shifts, it can overlap with its previous position. This causes some data instances to go through anomaly detection more than once. This leads to multiple estimations of the same data point, which can deliver new information about the instances. However, this can show to be more or less useful as the resulting predictions are able to contradict each other [AF07, SR18].

Tumbling Sliding Window This window method is a special case of the sliding window mentioned before. When there are more data instances up for estimation than there are instances in the window, it is called a tumbling window, or tumbling sliding window. It is special, as some data instances are omitted from the window and are not subject to anomaly detection [GÖ03]. This can be useful for example for time series with not equidistant data points, but with equidistant anomaly estimations.

Two Sliding Windows This method uses two sliding windows: the reference window and the last window (or detection window), which are joined on one side. In the first step of anomaly detection, data-driven parameters are retrieved from the reference window. In the second step, those parameters are applied on the detection window. This technique allows each instance to pass through anomaly detection once, as long as the detection window length equals the shift length [TTL11].

Which window method should be chosen depends completely on the application scenario, as each has its weaknesses and strengths. The presented window techniques can most often be executed in two different modes: eager reevaluation and lazy reevaluation. With each update interval are new data instances added to the stream. Eager re-evaluation will run an anomaly detection method on each new data instance, while lazy re-evaluation will process data instances in batches, which causes a less computationally expensive 'jumping window'.

2.3.3 Challenges of Stream-based Anomaly Detection

Stream-based detection models often work in real-time critical systems, as they have to check for anomalies in online systems where always new data is added. Such systems work close to real-time and should require only little expenditure of time. Also in real-time are no labels available and the models have to work fully unsupervised with good detection rates.
In these environments, no parameter adjustments are possible without taking the system offline, hence the method is required to be flexible for various data characteristics. It is a common problem that the underlying data might evolve or underlie changes as a concept drift. As shown in Figure 2.9, the data can change drastically. This can effect the detection performance immensely when using window methods, as this leads to a loss of context. We see, that the latest analyzed data (as $o1$ to $o5$) does not lie in the range of the newest data (as $o6$ to $o12$) anymore and hence will be labeled as anomalies. For all techniques this drift will effect the future predictions as long as it

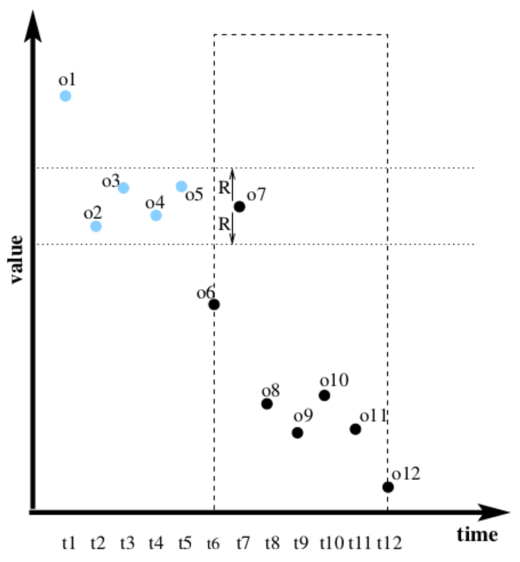


Figure 2.9: Example for a Concept Drift in a Data Stream [AF07]

is in their window. However, in techniques as the two window technique, this drift will affect the future predictions as long as it is included in the reference window, which is often far longer.

2.3.4 Measures for Stream-based Anomaly Detection

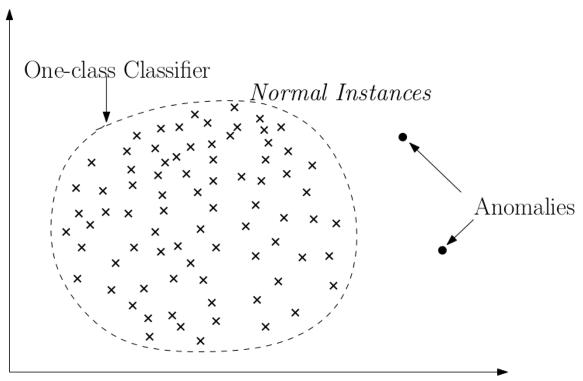
As stream-based anomaly detection models are designed to run close to real-time on uncompleted data, measuring the detection performance is not possible without further ado. A common method is to simulate the stream-based scenario on completed and labeled data and then to apply the same measurement methods for static detection methods shown in Section 2.2.4.

2.4 FAMILIES OF ANOMALY DETECTION METHODS

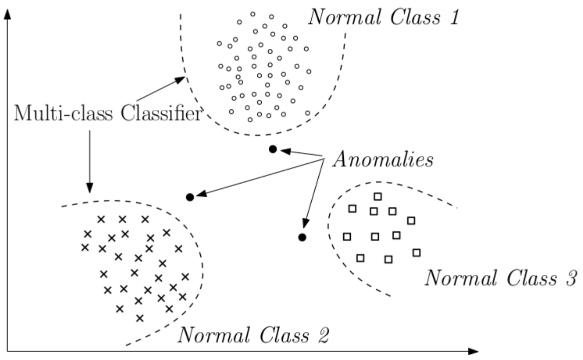
Stream-based and static anomaly detection use similar kinds of detection methods. In this section we provide an overview of the most common families of anomaly detection techniques. We have worked out a categorization of these methods. It is in parts based on [CBK09].

2.4.1 Classification-based Methods

Classification-based methods commonly work in two phases: In the training phase, a classifier is learning a model on prelabeled data instances. In testing phase, the classifier assigns labels to new data instances according to its internal model.
This process can be adapted for anomaly detection, as a classifier can work with 'normal' and 'anomaly' labels. Most commonly only 'normal' instances are modeled using one or multi class anomaly detection [CBK09].



(a) One Class Anomaly Detection



(b) Multi Class Anomaly Detection

Figure 2.10: Examples for Classification Anomaly Detection in Two-dimensional Space [CBK09]

One-class Classification In the training phase a classifier learns only data instances of the 'normal' class. As Figure 2.10a shows, the classifier models a boundary, which encircles the given instances. During the testing phase, each data instance outside of this boundary is labeled as anomaly. This approach requires special one class classifier as one-class SVMs or one-class kernel fisher discriminants.

Multi-class Classification Multi-class classification has only labels of multiple normal classes in the training data. As shown in Figure 2.10b, the boundaries for each class are calculated. Instances, which are not labeled by at least one classifier as normal, are labeled as anomaly. Extensions of this method also use confidence scores as anomaly scores. Exemplary classifiers are Neural Networks, Bayesian Networks or SVMs.

Using classification for anomaly detection has the advantage of fast labeling of new instances, as the classifier models are pre-computed once and then freely applicable. On the other hand, the basic approaches do not support scoring, as classifications only provide the labels for the tested data instances. Additionally, classification methods require accurate training data, which has to be prepared by hand. But any anomaly in the training data can lead to inaccuracies in the detection process.

2.4.2 Clustering-based Methods

Clustering methods are commonly used to identify groups of similar data instances in a data set and work in either unsupervised or semi-supervised fashion.
Anomalies can be detected with clustering under the assumption that normal data points are very similar to each other and form dense clusters, while anomalies are more secluded from other data points. The following three detection approaches make use of these properties:

Incomplete Clustering Methods This detection approach assumes, that normal data points form clusters as they occur very often, while anomalies do not. For separation, a certain family of clustering methods can be used, which do not assign every data point to a cluster. Examples of such incomplete clustering algorithms are DBSCAN [EKS$^+$96], ROCK [GRS00] or sNN [ESK02]. Here anomaly detection is only a side product of the applied algorithms.

Cluster Centroids-based Methods This approach uses algorithms, which cluster all data instances. For each instance the distance to its nearest cluster centroid is calculated as its anomaly score. Example algorithms for clustering are Self-Organized Maps (SOM) [Koh97], k-means clustering [Llo82], and Expectation Maximization (EM) [DLR77]. Additionally, semi-supervised clustering can be applied, to improve the clustering process [CBK09].

Cluster Density-based Methods To use clustering density for anomaly detection, it is assumed, that normal data instances occur in large and dense clusters, while anomalies occur in sparse clusters. Algorithms as FindCBLOF [HXD03] capture a clusters density, and assign a value to each data instance called Cluster-based Local Outlier Factor (CBLOF) considering its cluster size and its centroid distance.

Clustering-based detection methods have the benefit of their unsupervised and datadriven fashion, because only few assumptions on the data are made.
They allow fast testing, as only the comparison of a test instance to the constant number of clusters is required. The performance of the clustering process itself depends on the clustering algorithm. The two approaches of incomplete clustering and cluster centroids have the flaw of unregarded anomaly clusters. Both approaches would detect them as normal clusters and classify them as normals, instead of anomalies [KL05, CBK09, DXLL09].

2.4.3 Nearest Neighbor-based Methods

This family of anomaly detection techniques requires the definition of a distance measure between the data points to estimate neighborhoods. The distance is assumed to be a similarity measure: The closer points are to each other, the more similar they are assumed to be. One common measure is the Euclidean distance [HA04] :

$$dist_{Euklid} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.5}$$

Here defined with two i-dimensional vectors x and y.
The other common measure is the Mahalanobis distance using the covariance matrix C [HA04]:

$$dist_{Mahal} = \sqrt{(x-y)^T C^{-1}(x-y)} \tag{2.6}$$

In contrary to the Euclidean distance, the Mahalanobis distance can also be applied on correlated dimensions. But it is more computationally expensive than the Euclidean distance. With the construction of a covariance matrix a pass through the whole data set is required [HA04].
The chosen distance measure is then used in one of the two major types of nearest neighbor anomaly detection techniques: either the distance to the kth nearest neighbor is calculated, or relative densities are computed.

Using k Nearest Neighbor Distance In this type of detection method, it is assumed that points with high distance to their neighbors are outliers, while points in close neighborhoods are

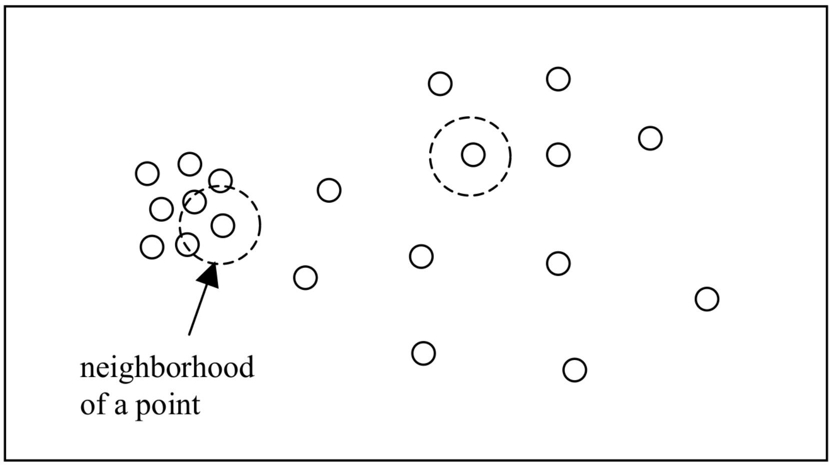


Figure 2.11: Definition of Neighborhood in Sparse and Dense Groups [ESK02]

inliers. The basic nearest neighbor-based anomaly detection technique calculates the distance of a data instance to its *k*th nearest neighbor. Whereby the distance serves as an anomaly score [CBK09]. Similar versions of this algorithm calculate the distance to the first neighbor [GMESK99] or use the sum of distances of the k nearest neighbors [EAP$^+$02, AP02].

Using Local Density This method assumes, that points in an area with low density are outliers, while points in close neighborhoods with high density are inliers. For these comparisons, the local densities are estimated using for example the Local Outlier Factor (LOF) [BKNS00]. The method estimates for each data instance a ratio between its local density and the average local density of its k nearest neighbors. A more detailed explanation of this method can be found in Section 3.2.4.
The original LOF approach is with $\mathcal{O}(N^2)$ computationally expensive [CBK09, HA04], which is why there are several optimizations for this algorithm. The Connectivity-based Outlier Factor (COF) uses for example an incremental approach[TCFC02], while the Outlier Detection using In-degree Number (ODIN) uses Nearest Neighbor lists[HKF04].

Nearest Neighbor anomaly detection has the key advantage of being purely data driven and unsupervised. There has been a lot of scientific research in this area. However, Nearest Neighbor detection methods have a weakness of operating poorly on clusters with varying density, especially with clusters of normals which have low density. As Figure 2.11 shows, the data instance in the left cluster has more neighbors within a certain distance, than the right instance. Hence, the right instance will be detected as anomaly, although the data points in the whole cluster are distributed differently. In the same way, clustered anomalies can be accidentally identified as inliers, when they have a similar high local density as normals.

2.4.4 Statistic-based Methods

The first anomaly detection methods were actually statistical methods applied on univariate data [HA04]. When using this type of methods it is assumed, that anomalies are generated by a different stochastic model than the inliers. Once a sufficient model for the normals is found, unlikely data points can be identified as anomalies. Statistical detection techniques impose a previously known model on the data, which makes them parametric. Different mathematical models are common, such as Gaussian models or regression models [CBK09, AY01].

Gaussian Models To use Gaussian models assumes that the underlying data originates from a Gaussian distribution [CBK09]. One of the most simplest techniques is the $\mu \pm k\sigma$ method. First, it calculates the mean μ and the standard deviation σ. Then, any data point outside the range of $\mu \pm k\sigma$ is labeled as an anomaly [She31].
The Boxplot Rule likewise determines minimum and maximum values for normal data using the 1.5 fold of the Inter Quartile Range (IQR). Here as well, each data point above the maximum or below the minimum are declared as anomalies [LJK$^+$00]. Other methods in this field are for example the Grubb's test, also called maximum normalized residual test [Gru69] as the student's t-test for anomaly detection in univariate data [SW$^+$98], and its related version the hotelling t^2 test for multivariate data [LW91].

Regression Models These models are mainly applied to time series. Once they are fitted to the data, the residuals between instance and its regression estimate are calculated and used as anomaly score. Some regression-based detection methods have the weakness, that anomalies in training data influence the results. Fortunately, more insusceptible methods can be applied as the method called robust regression[RL05], the Autoregressive Moving Average (ARMA) [GPT04] or the Autoregressive Integrated Moving Average (ARIMA)[BGBMY01].

Statistical detection models have the advantage of being able to operate in unsupervised settings. Discovering a good fitting statistical model for the data leads to better anomaly detection rates as the underlying distribution is remodeled. In addition, valuable knowledge about the application case is gained. However, a good fit is most often unlikely, as the generative nature of the data source is often complex. Especially in high dimensional data, finding a good fit is difficult [CBK09].

2.4.5 Histogram-based Methods

This smaller family of techniques creates histograms for anomaly detection. As shown in Figure 2.12, the basic univariate example first discretizes the data by splitting the distance between the maximum value and minimum value in equally sized bins (x-axis of histogram) and then populates them with the corresponding data instances (y-axis of histogram). The anomaly score of an instance is the population of its allotted bin. Instances of less populated bins can be declared as anomalies by the application of a threshold [AY01]. Hereby, the detection rate is dependent on the bin size. If the bin sizes are too small, less data points will fall into each bucket and the difference between abnormal and normal buckets will be less obvious. If the bin sizes are too large, more anomalies fall into normal buckets and vice versa, hence a good trade-off is required.
Different versions and extensions of this basic histogram method have been published. A multivariate approach creates a histogram for each variable in the data, then the anomaly scores for each test instance are aggregated [CBK09]. Another supervised versions of this procedure exist

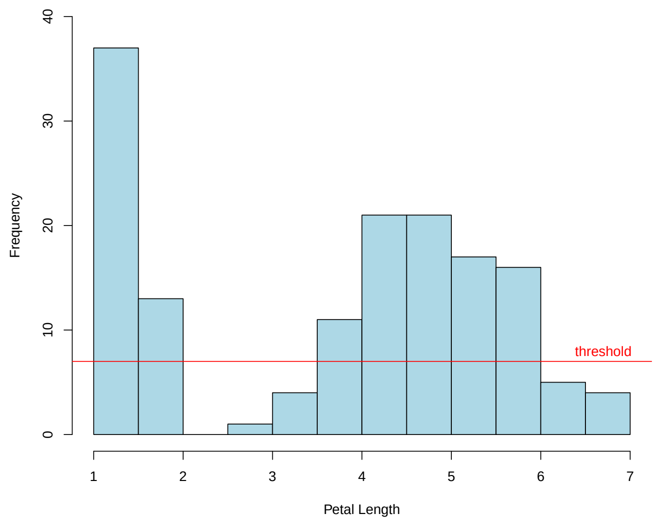


Figure 2.12: An Example Histogram for Flower Petal Length with 12 Bins

where only normal instances are used for histogram construction, then a data instance is tested by checking its membership with the created bins. If it fails, it is an anomaly [JV91, HB97].
Histogram-based anomaly detection is similar to clustering-based anomaly detection, as they both focus on density. But clustering methods partition the data instances into groups, while histogram-based methods partition the data space [AY01].
Altogether histograms have the advantage of being either supervised or unsupervised for univariate or multivariate data and are therefore very flexible in application. They have the minor disadvantage that the multivariate variant is not able to consider correlations between the variables. These methods are prone to high dimensional data, as with increasing dimensionality the population of bins decreases exponentially with number of features [AY01].

2.4.6 Neural Network-based Methods

Neural networks are popular for their high prediction performance, when successfully constructed. They can work in supervised, semi-supervised, and unsupervised manner. Supervised neural networks with accurate labels have the highest chance of success, as semi-supervised and especially unsupervised methods require more fine-tuning.
Since the literature in this field is innumerable and would go beyond this measure, we limit ourselves to introduce only the popular Autoencoder and the Self Organised Maps (SOM) [Koh97].

Autoencoder (AE) [HZ94], also called Replicator Neural Networks (RNN) or Auto-associative Networks, consists of an input layer, an output layer, and one or more hidden layers in between. The number of neurons in the input and the output layer have to be equal, but the neurons in the hidden layer(s) must be fewer to force a compression and a reconstruction step. For training, normal data instances are provided to the Autoencoder to learn the reconstruction of the exact input. Then in the test phase, if a data instance cannot be reconstructed and the reconstruction error is too high, the instance is declared as anomaly.

Several such Autoencoder architectures are possible. For image data convolutional networks [Fuk80] are advised, while for sequential data Long Short Term Mermories (LSTM) [HS97] are more qualified [CC19].

Self Organised Maps (SOM) [Koh97] cluster the normal input data and form reference vectors for each cluster during training phase. For testing, a data instance is compared to each reference vector. If it matches, it is an inlier and part of a cluster. If it fails, the instance is declared to be an anomaly [CBK07].

Other popular methods for deep anomaly detection are Neural Trees [Mar98], Multi Layered Perceptrons [AF02] or Hopfield Networks [CH+01].
Deep learning anomaly detection has the advantage of being an area of high interest. Improvements and extensions are in permanent development. However, deep learning detection techniques come with some disadvantages: For one, the methods are prone to anomalies or flaws in the training set, they perform best on accurately labeled or normal data, otherwise the model will be error-prone [KYL+16].

2.4.7 Frequent Pattern-based Methods

This family of detection techniques was introduced by [HXHD05] with the first method called Frequent Pattern Outlier Factor (FPOF). This detection methods are based on the assumption that frequently occurring patterns in the data are normals, while rarely occurring patterns represent anomalies.
[HXHD05] applied FPOF only on database transactions by reusing techniques from frequent pattern mining, but in general the application on grouped data is possible, as long as there are multiple items I for each group D. For the calculations, the measures of support is required. The support $support(x \longrightarrow y)$ denotes the count of occurrences the two items x and y have together in I. Frequent patterns $FPS(D, minisupport)$ are declared to be the set of items X of an item set I with higher support than a given threshold $minisupport$.

$$FPS(D, minisupport) = \{X \subseteq I | support(X) \geq minisupport\}. \tag{2.7}$$

Each frequent pattern based method calculates the outlier factor by the use of these methods different. For example, the FPOF score of a group t is calculates as:

$$FPOF(t) = \frac{\sum_{X \subseteq t, X \in FPS(D,minisupport)} support(X)}{|FPS(D, minisupport)|}. \tag{2.8}$$

The resulting scores for each frequent pattern-based method commonly fluctuate between 0 and 1. The closer the score of a group t is to 0, the likelier it is an outlier.
As FPOF is time-consuming and the size of the extracted frequent patterns is large, multiple improved versions of this algorithm exist, as for example the methods called Weighted Frequent Pattern Outlier Factor (WFPOF) [ZSZY07], Maximal Frequent Pattern Outlier Factor (MFPOF) [LLB10], Frequent Pattern Contradiction Outlier Factor (FPCOF) [TLC09], Weighted Closed Frequent Pattern Outlier Factor (WCFPOF) [RWHW09] and Longer Frequent Pattern Outlier Factor (LFPOF) [ZWY10].

2.4.8 Ensemble Methods

Ensembles are methods, which use multiple submethods for anomaly detection. They are known as more robust to anomalies in the input data and as more performant. Through their joint manner, they can overcome the weakness of singular unsupervised detection methods. As outlier detection ensembles are rather new and few research is available. In literature, they are divided in the two categories of sequential and independent ensembles[AS17]. In sequential ensembles methods are applied sequentially, so successor methods works on the results of their predecessors. While independent ensembles apply the same algorithm on the data or parts of data.
Here we briefly introduce an independent ensemble technique called Isolation Forest (IFOR). Its basic idea is in some way similar to a random forest, only it is unsupervised. By the use of multiple trees the overall performance is heightened. The isolation forest is composed of isolation trees, which partition the data with random axis-parallel cuts until all points are isolated. The assumption is, that anomalies are isolated in fewer splits as they lie spare, where normals require far more splits as they lie closer together. An instances anomaly score is the average over its tree depth in all isolation trees [AY01].
Isolation forests are similar to histogram methods, as they also partition the data space (see Section 2.4.5). In comparison does the isolation forest approach avoid the challenge of finding a good bin size by choosing its partitions randomized [AY01]. With the randomization arises the challenge of tree imbalance, as for a data pool with N instances the creation of trees with depth N becomes possible. Most often, additional parameters are required like a maximal tree depth to receive more balanced isolation trees.
Other ensemble anomaly detection techniques, as [GT06], use models trained on varying parameters or combine the output of different detection techniques as [NAG10, KKSZ11]. This family of techniques is especially efficient where the detection success depends on choices in the model design, as for example multiple parameter settings can be tried on the data. For example, in high-dimensional data, the choice of a data space can be covered by ensembles that use multiple sub spaces.

2.5 CONCLUSION

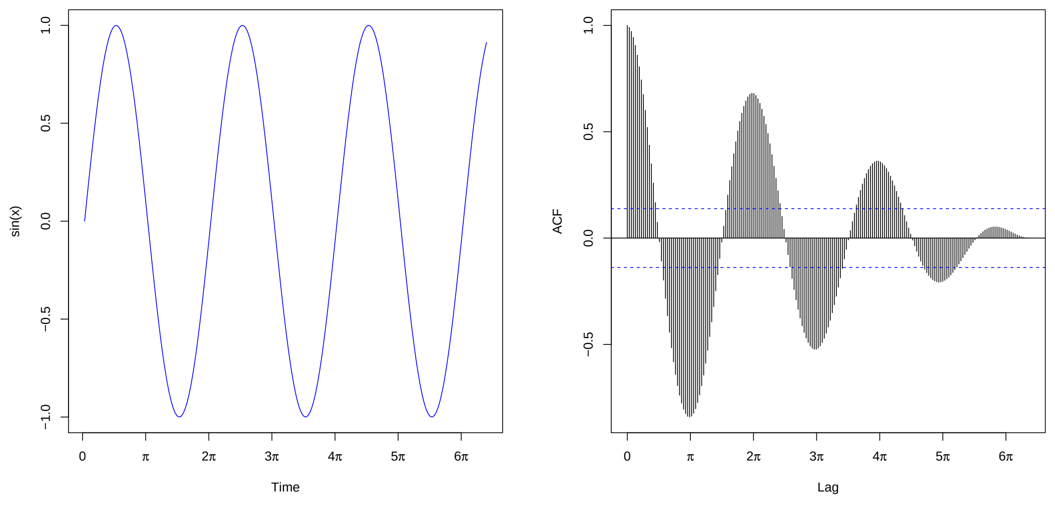
In this chapter, we have introduced the basics of anomaly detection. We first presented the three basic types of anomalies: Point-based, Collective and Context-based Anomalies. Then we elucidate a general anomaly detection model minimally represented in Figure 2.4. We briefly describe some important design decisions as parameter choice and data selection. Then common methods of performance evaluation are introduced.
In the next section, we address the specific domain of stream-based anomaly detection. The shifting detection model for streams has a compact representation in Figure 2.7. Then we introduce six different window techniques for the mentioned model and describe additional problems of stream-based anomaly detection. The last section of this chapter treats families of detection methods, which can be used on stream-based or on completed time series. The basic principles of eight different families are briefly explained and some example algorithms or sub categories are listed.

3 FUNDAMENTALS OF ANOMALY DETECTION IN TIME SERIES DATA

A time series is a sequence of data instances continuously collected over time. Some examples are daily rainfall amount data or the Dow Jones stock market index. This type of data is wide-spread. It occurs in countless domains as in economics, astronomy, medicine, computer science, physics, and in the social sciences. [Wei06]
Elements of a time series have a shared temporal context, hence adjacent instances are dependent on each other. This interplay of different temporal factors is of high interest. The domain of time series analysis focuses solely on their determination. These dependencies also allow the generation of forecasts and the detection of anomalous states. [Wei06]
The data instances can, but do not have to, be equidistant to its neighboring data points. In the related domains of event detection and change point detection, different time intervals are used. For simplicity, this thesis works only with equidistant neighbored points in time.
In this chapter, we will first introduce some basic methods specific for time series as decomposition and correlation detection. In the next two sections, we explain specific static and stream-based anomaly detection methods for timeseries in full detail.

3.1 TREATING TIME SERIES FOR ANALYSIS

The time dependency and order is special for time series data. They may contain further information, that can be used for better anomaly detection. Based on these interdependencies, we can make use of the autocorrelation function (Section 3.1.1), which detects the seasonality in the data. Based on this, we can remove trend and season via decomposition (Section 3.1.2) of a time series to receive the inexplicable values. Finally, we can display the decomposed values in a more insightful manner (Section 3.1.3).



(a) Example Series of $sin(x)$ for $[0, 6\pi]$ **(b)** Resulting Auto Correlation Plot

Figure 3.1: Example for an Auto Correlation Function on Sinusoidal Data

3.1.1 Seasonality Detection

The auto correlation function allows us to detect how the past and future data points are related by calculating the correlation between the time series itself and its lagged version. The autocorrelation r_t for a time series X is calculated as follows:

$$\begin{aligned} c_t &= \frac{1}{n} \sum_{s=max(1,t)}^{min(n-t,n)} [X_{s+t} - \overline{X}][X_s - \overline{X}] \\ r_t &= \frac{c_t}{c_0} \end{aligned} \tag{3.1}$$

As an intermediate step the autocovariance c_t is calculated using: n as the number of elements in X, s as an element in X, t as the chosen lag, and $\overline{X}$ as the mean of X [VR13].
Using these formulas with a varying lag, an autocorrelation plot can be created. The autocorrelation plot for an exemplary sine curve (Figure 3.1a) is shown in Figure 3.1b. As depicted, the correlation of a point to itself is always 1. It is the highest possible correlation. In case of a negative correlation the value is at minimum -1. As the overlap between the time series $[t, n - t]$ decreases with growing lag t, the correlation is approaching zero.
We can see from the graph, that the correlation of the sine function is highest for every multiple of 2π, hence we assume this to be the strongest seasonality of t. Common correlations in natural data are weekly, daily or hourly periods.

3.1.2 Decomposition

A decomposition uses the time context of data points to extract seasonal, trend, and remainder information. In particular, we use the seasonal-trend decomposition method (STL) by [CCMT90]

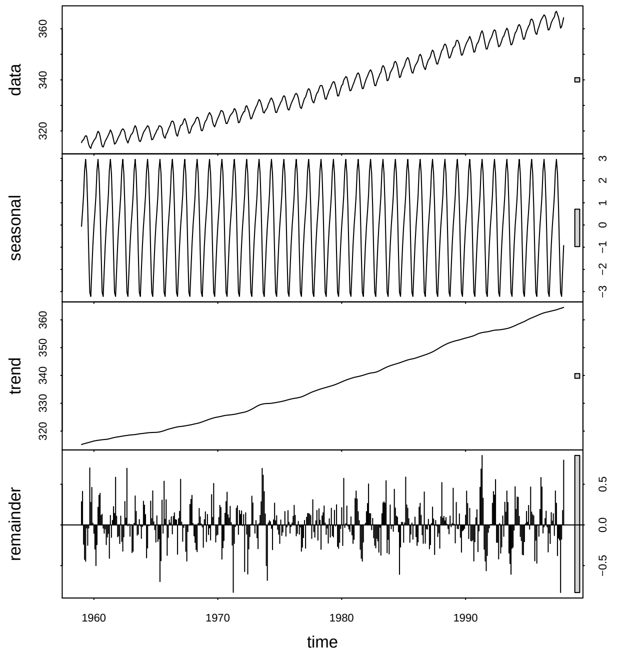


Figure 3.2: An Example Decomposition Using Grass CO_2 Intake Data

based on the locally-weighted scatterplot smoothing (also called loess smoother) [Jac00]. STL can process even long time series in short time and is able to handle missing values. Also it can run in robust mode, where anomalies have only effect on the remainder.
STL hast the two parameters of trend window width and seasonality window width, which determine how rapidly trend and season can change, hence STL is adaptable to each time series. As a disadvantage, it does not treat calendary dependencies, for example for a yearly seasonality the 29th February has to be erased.
As a full explanation of STL would exceed the boundaries of this thesis, we keep the explanation brief. The STL algorithm consists altogether of two loops: an inner and an outer loop. The algorithm starts with the initialization of robustness weights set all to 1. Then the outer loop first executes the inner loops, with seasonality and trend smoothing, then adapts the weights for the next cycle. The weights are set to be stronger for average data values, while they are set weaker for aberrant and transient values. The inner loop consists of five steps: detrending, season subseries (e.g. series of all Januaries) smoothing, a low-pass filter, detrending, deseasonalizing and trend smoothing [CCMT90].
It can run in additive mode or multiplicative mode. The additive mode should be chosen, if the seasonal change has the same magnitude across the time. In this mode, the full time series Y_t is handled as the result of the sum of trend T_t, season S_t, and remainder e_t.

$$Y_t = T_t + S_t + e_t \tag{3.2}$$

While the multiplicative mode, using the product of trend T_t, season S_t, and remainder e_t, should be chosen, when the seasonal change increases over time.

$$Y_t = T_t S_t e_t \tag{3.3}$$

Figure 3.2 shows the result of an example decomposition. In the first image segment, we see the decomposition input ('observed'), below is the calculated overall trend ('trend') with a slightly increasing curve, followed by the segment for the seasonal ('seasonal') change, each spike is part of its own seasonal window. The remainder ('random') is the last segment, which contains the data which cannot be explained by season or trend. The remainder should vary by the zero point. It can include noise as well as anomalies, hence that's what the anomaly detection is going for.
Of course decomposition should only be applied, if the anomaly detection method does not use the seasonality and remainder information itself.

3.1.3 Data Exploration

It is strongly advised to investigate in data exploration before starting with anomaly detection, as it can uncover useful traits in the data. While this work relies on expert knowledge and the advised choice of data streams, nonetheless we are conducting an investigation of the data. We use a plotting method exemplary depicted in Figure 3.3. Here we plot the value distribution for each time series on the diagonal. This allows to detect suspicious value accumulations and shortages, furthermore it support the selection of a detection method. Depending on how strongly the distributions are pronounced, distribution-dependent methods (eg. Gaussian-based Statistical Methods) can be applied.
The part below the diagonal shows the pairwise plots of the time series independent of time. This method of representation allows a new insight into the underlying interrelations as strong correlations become visible in the form of clusters.
The upper half of the plot, shows the value of the pairwise correlation of the time series itself to support fast evaluation. High correlation values indicate an interdependence between the time series, hence this form of presentation allows the uncovering of relations within the host system.

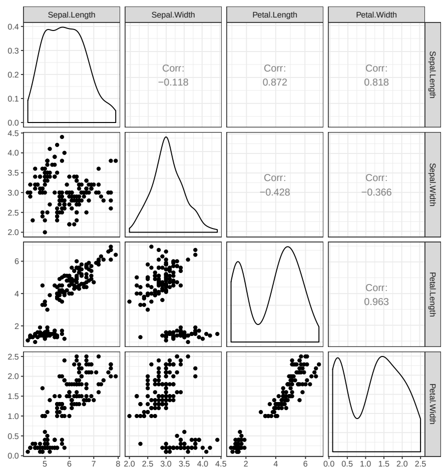


Figure 3.3: Exemplary Correlation Plot Table for the Iris Flower Data Set

3.2 STATIC DETECTION METHODS FOR TIME SERIES DATA

In this section, we want to introduce some static detection methods as described in Section 2.4 in detail as they are used in the following chapters. The underlying data for static detection always consists of completed time series. The methods have found application in various domains.

3.2.1 $\mu \pm k\sigma$ Method

This method was published by [She31] and originates from 1931. It was used for quality estimation in product control using time series data. The method allows the detection of anomalies, as it sets the upper and lower limit for normal data points:

$$\begin{aligned} Min &= \mu - k\sigma \\ Max &= \mu + k\sigma \end{aligned} \tag{3.4}$$

μ is the mean of the time series, σ is the standard deviation, and t is a factor based on experience. As k is set to 3 by default in the publication, the method is sometimes called $\mu \pm 3\sigma$ method [CBK09].

3.2.2 Boxplot Rule

The Boxplot rule, also 1.5xIQR rule, has been applied on medical data for outlier detection by [LJK+00] and [SL05]. Figure 3.4 shows such an exemplary boxplot.
The original paper applies in the publication a Box Cox transformation on the data to bring it in a Gaussian shape. Then the rule estimates the upper quartile $Q1$ and lower quartile $Q3$ and calculates the inter quartile range (IQR), also called the middle-50%. In the next step, the rule

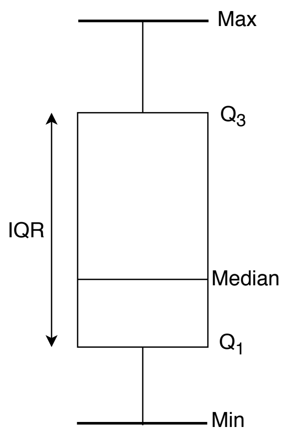


Figure 3.4: An Example Boxplot for Univariate Data

calculates the upper and lower boundaries for the normal data:

$$\begin{aligned} Min &= Q_1 - 1.5IQR \\ Max &= Q_3 + 1.5IQR \end{aligned} \tag{3.5}$$

All data instance outside of these boundaries are declared to be outliers [SL05, HFLP01].

3.2.3 Incomplete Clustering with DBSCAN

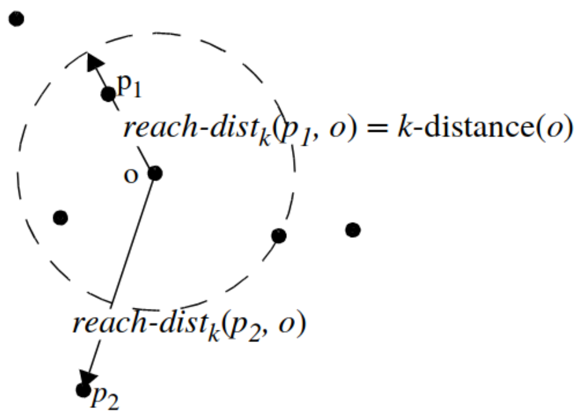
This method uses special clustering algorithms, which do not assign every data point to a cluster. In this specific case we use the Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm, published in 1996 by [EKS+96].
DBSCAN requires two parameters: ϵ, a points neighborhood radius, and $minPts$, the minimum number of data points required to form a cluster. The DBSCAN algorithm reformulated by [SSE+17] selects a random unvisited data point from the data set and checks its ϵ-reachability. If the checked data point p has fewer than $minPts$ neighbors within ϵ-distance, it is labeled as noise and the next point is checked. If p has more than $minPts$ neighbors, it is a core point as it is in the middle of a cluster. Then each ϵ-neighbor of p is checked. If the neighbor q is unlabeled or has been labeled as noise before, it is now relabeled as part of the cluster, as it is a border point. If q is also a core point, its neighbors will be recursively checked until all border points enclosing the cluster are found.
The anomaly detection technique takes all points that DBSCAN labeled as noise as outliers [ÇDÇD11]. Anomalies are only a byproduct, as it is not optimized for anomaly detection.

3.2.4 Local Outlier Factor (LOF)

This technique was published by [BKNS00] in 2000. It is a Nearest Neighbor detection technique, which estimates the local density by calculating the ratio between the estimated density of a point p and the averaged density of its k nearest neighbors. The LOF is defined using multiple formulas, hence it is best to explain it step by step. The calculation uses the $k\text{-}distance(o)$, which is the distance to the kth neighbor of a point o. Per definition, this distance can include more than k neighbors, in case multiple points have the same distance. As each k neighbor for each data point is different, the distance measure is not symmetric.
The first formula is the definition of the reachability distance $reach\text{-}dist_k(p,o)$ of p to a point o using the maximum of the k nearest neighbor distance $k\text{-}distance(o)$ and the distance between both points $dist(p,o)$:

$$reach\text{-}dist_k(p,o) = max\{dist(p,o), k\text{-}distance(o)\} \tag{3.6}$$

Figure 3.5 demonstrates this measure using $k = 3$. The reachability distance between p_1 and o is the k-distance, as $d(p_1,o)$ is smaller. While $reach\text{-}dist_k(p_2,o)$ is $d(p_2,o)$, as the k-distance is smaller.
Then the local reachability density lrd is calculated, as the inverse of the averaged reachability



p_1
$reach\text{-}dist_k(p_1, o) = k\text{-distance}(o)$
o
$reach\text{-}dist_k(p_2, o)$
p_2

Figure 3.5: The Reachability Distance as Maximum for $kdistance(o)$ and $dist(p,o)$ using $k = 3$[BKNS00]

distance of all points in the k-neighborhood $N_k(p)$ of p:

$$lrd_k(p) = 1/\frac{\sum_{o \in N_k(p)} reach\text{-}dist_k(p,o)}{|N_k(p)|} \tag{3.7}$$

At last the LOF score itself is calculated as the average of the ratio of the lrd of p and its k-neighbors:

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|} \tag{3.8}$$

When the lrd of neighbors and of p are similar, is the LOF score close to 1. If the local density of p is lower than its neighbors density, then the score will be higher than 1. If $lrd_k(p)$ is higher, will the score be lower than 1. As we assume that data points with lower local density than its neighbors are outliers, a threshold on values greater than 1 is applied.

3.2.5 Cluster Centroid Distance

The detection method in this specific version was introduced by [PD12] in 2012. It uses k-means clustering [Llo82] to group the data points unsupervised in k clusters. For initialization, the algorithm uses a random set of k means. In the first phase, it calculates the Euklidean distances from each data instances to the means. In the second phase, it assigns all data points to the clusters with the nearest centroid mean. These two steps are repeated until the means are stationary and no data instances changes its cluster or until a certain amount of iterations has been achieved. Then for anomaly detection, all distances to the corresponding centroids are calculated and a threshold is applied to detect the outliers [PD12]. Other versions of this method use expectation maximization clustering [SBE$^+$02] or Self-Organising Maps [Koh97].

3.2.6 *k*th Nearest Neighbor Distance

[BR98] used this method in 1998 to detect minefields in graphical data. It solely uses a points Euklidean distance to its kth neighbor as anomaly score, as the distance already gives an estimate of the proximity of the next k neighbors and the points neighborhood density. The higher the distance to the kth neighbor the sparser is the points vicinity and the likelier is it to be an outlier.

3.2.7 Histogram-based Outlier Score (HBOS)

This scoring method was published by [GD12] in 2012. It is an advanced detection technique, which creates histograms for multivariate and univariate data in either static or dynamic mode. Which mode is chosen, depends on the underlying data. Different point distributions may occur in real life, hence two different modes have been implemented by [GD12].
At the beginning, in both methods d univariate histograms are created for all d dimensions in the data. In static mode, the value space between the minimum and the maximum point is split in k even bins. A common choice for k is $k = \sqrt{N}$, for N being the number of points. The score $hist_i(p)$ for a point p is the its bin height, the total number of data instances with which it falls into a bin. In dynamic mode, first a sorting technique is applied, to then group all instances N in bins of size N/k. As the number of observations is the same for all bins, the frequency score $hist_i(p)$ can be calculated by dividing the number of observations per bin by the bin width. An exception is made for multiple data points with the same value. In this case, a bin is able to contain more than k instances.
In dynamic mode, $hist_i(p)$ depends on the data point distribution between the maximum and the minimum point, whereas in static mode, it depends on a fixed grid between those points. Whereby, the fixed grid is known to be the weakness of the static method, as it does not adapt on the data. To support the multivariate data format, all histograms are normalized to have a maximum height of 1. Then the anomaly score for each data p instance is calculated by applying:

$$HBOS(p) = \sum_{i=0}^{d} log(\frac{1}{hist_i(p)}) \tag{3.9}$$

The resulting score can maximal assume the value 0. The higher the frequency score $hist_i(p)$, the lower is the resulting HBOS score. Hence the final score of outliers is higher, than the final scores of normal data instances. The static method achieves scoring in linear time $\mathcal{O}(N)$, while the dynamic methods scores in $\mathcal{O}(Nlog(N))$ time.

3.3 STREAM-BASED DETECTION METHODS FOR TIME SERIES DATA

This section will explain some stream-based anomaly detection methods in detail. They work in a similar manner as static methods, except that they are often combined with stream windows and hence use only parts of the data and are not complete.

3.3.1 Isolation Forest (IFOR)

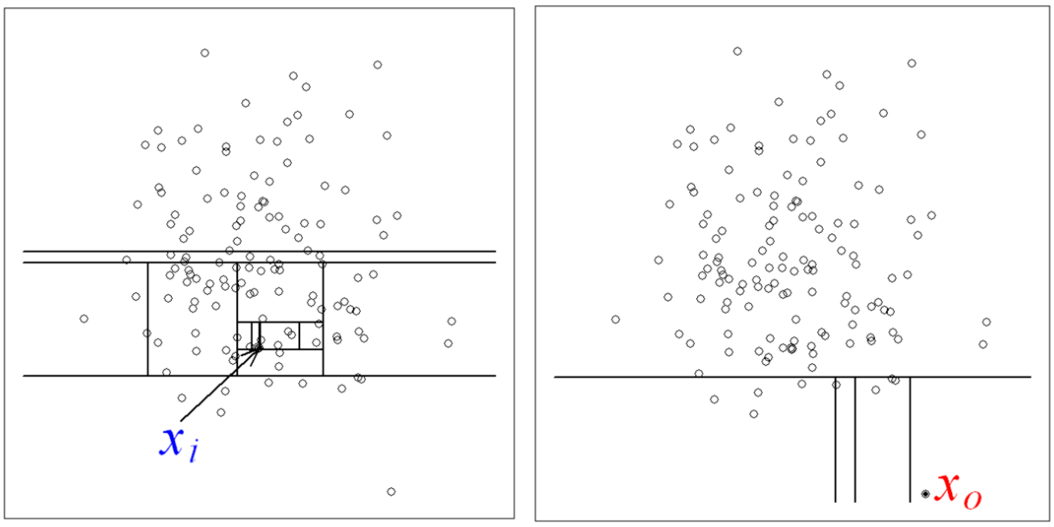
The Isolation Forest (IFOR) is an ensemble technique published by [LTZ08] in 2008, which was later also published for stream data[DF13]. For anomaly detection, it uses the advantage of anomalies to be different and more distant from the other data. In training phase, trees are constructed on the data as shown in Figure 3.6a and Figure 3.6b. Repeatedly, the data is randomly partitioned using random dimensions and random splitpoints. Thereby, a tree is created, which is limited in its depth by a user.defined maximum tree depth It is assumed, that as a result, anomalies like x_o are closer to the root of the tree, while normals as x_i are located deeper in the tree. Multiple such isolation trees are used as their combination increases significantly the detection performance. To test a data instance, the score $h(x)$ is calculated by counting all edges from root to its position in the tree for all trees. The expected path length is the average $E(h(x))$ of the path length of a data instance for all trees. As the average height of a tree grows with $log(n)$ with n being the number its instances, the average can be estimated with the following formula:

$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{3.10}$$

Thereby $H(i)$ is the harmonic number, calculated as the sum of $ln(i)$ and the Euler's constant e. With these prepared values the anomaly score can be calculated:

$$score(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{3.11}$$

The anomaly score varies between 0 and 1. The higher the score, the likelier is a data instance an anomaly. If a score is below 0.5 it is assumed to be a safe inlier.
This calculation is of linear complexity and has the high advantage of being valid for even high dimensional problems [LTZ08].



(a) Isolation of Inlier x_i **(b)** Isolation of Outlier x_0

Figure 3.6: Example for the Differing Isolation Tree Depth of Two Data Points [CBK09]

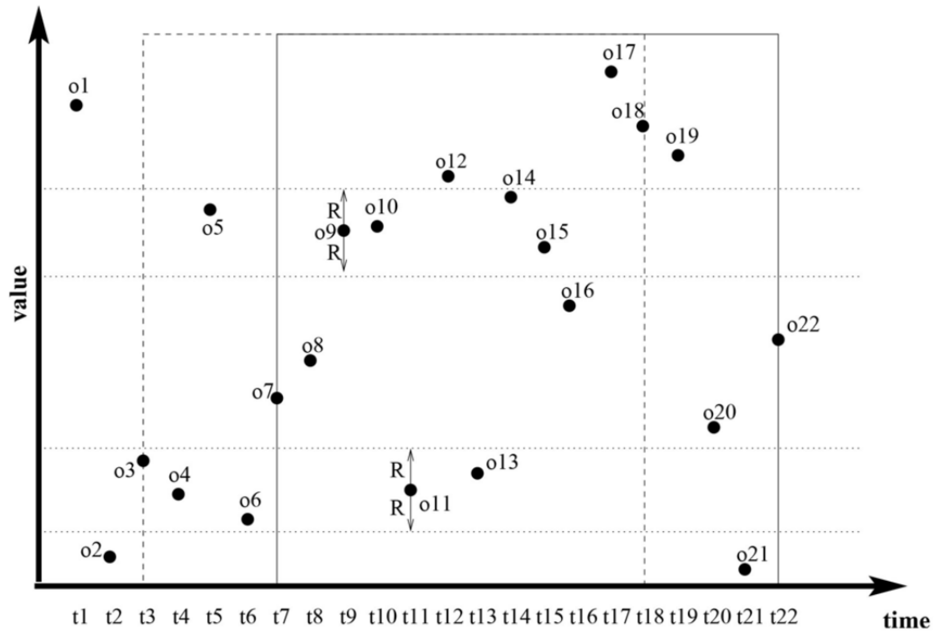


Figure 3.7: Example for Distance-based Outliers Using Sliding Windows [AF07]

3.3.2 Stream-Neighborhood-based Method

The Stream-Neighborhood-based Method is a simple theoretic example of [AF07] from 2007. Where it is explained using the image from Figure 3.7. We see two sliding sliding windows, the previous (dashed) and the current one (solid). From these windows, we choose a certain range for estimating the anomaly score of point o by calculating $o \pm R$. In Figure 3.7, we see a demonstration o_9 and o_{11}. As a result we receive an extract from the value range. Then all predecessors (also called neighbors) of o within the required window are counted as its anomaly score. A data instance is defined as an outlier, if the score is lesser than a threshold k, otherwise it is an inlier. Dependent of the window position, the detection results can change and contradict each other. For example, with $k = 3$ the point o_9 is in both the previous and current window and outlier. As it hast at most one predecessor, o_5, in its value range and never more than three. Different for o_{11}. In the previous shift, the point was an inlier, as is had the neighbors o_3, o_4, and o_6. In the current window it is an outlier with no preceding neighbors in its value range.

The algorithm allows to define safe inliers considering the data instances which follow after a considered data point. Safe inliers are data instances which have at least k succeeding neighbors within the window and R range. Of course, not in all scenarios are data points for considering the succession available.

This method is computationally efficient, but has the minus that the method is only defined for univariate data. In [AF07] it is also demonstrated that this method is very prone to a concept drift, as neighborhoods become very sparse during a shift and the method would label most data as outliers [AF07].

3.3.3 Frequent Pattern Outlier Factor (FPOF)

As declared in Section 2.4.7, this method was introduced by [HXHD05] and makes use of the support measure from frequent pattern mining. To recall, frequent patterns $FPS(D, minisupport)$ are declared to be the set of items X of an item set I with higher support than a given threshold $minisupport$.

$$FPS(D, minisupport) = \{X \subseteq I | support(X) \geq minisupport\}. \tag{3.12}$$

The FPOF score for t is calculated by dividing the support value for each frequent pattern in t by the number of frequent patterns.

$$FPOF(t) = \frac{\sum_{X \subseteq t, X \in FPS(D, minisupport)} support(X)}{|FPS(D, minisupport)|} \tag{3.13}$$

The resulting scores range from 0 to 1. Since it is assumed that outliers are rare and therefore have low frequency patterns, they have low FPOF scores.

3.4 CONCLUSION

In this chapter, we have described the methods and measures for anomaly detection in time series data in detail. They are the building stones for the experiments in the next chapter.
To recapitulate, we first discussed the preceding time series treatment, by explaining the seasonality detection via auto correlation function, the season and trend decomposition of time series, and finally we introduced a special correlation visualization method.
In Section 3.2, we explained static anomaly detection techniques in detail. Whereas in Section 3.3 the stream-based anomaly detection methods are described in more depth. We have tried to select different methods from different categories, but there is not as much diversity in the stream-based anomaly detection domain, as in the static domain.
All together we have introduced twelve methods, as shown in Table 3.2. Of the eight introduced detection method families in Section 2.4, we have implementations of six method families for time series data. Two method families were not used. The classification-based Methods were not implemented, as they require labels for learning. The Neural Network-based methods were not implemented due to their high configuration effort. Two of the selected methods have been published (Isolation Forest and FPOF) in the stream-based anomaly detection domain, as in the static anomaly detection domain and hence are listed twice in the table. In total, the introduced methods include seven multivariate and five univariate detection methods, of which two methods work with univariate and multivariate data. We tried to detect as many different types of anomalies as possible, but unfortunately there are imbalances in the literature. For example, only HBOS could be discovered as a method for collective anomaly detection, while point-based and context-based anomaly detection techniques are far more popular. When examining the table, note that point-based anomalies do not exist in the stream-based environment, as the sliding window techniques always restrict the anomaly estimation to the time context. Also we show here a different table to assess the detection methods. As summarized in Section 3.4, three anomaly detection techniques have label output, while the other eight methods have scores as output. However for evaluation purposes, we had to transform the label output to numeric binary scores using 0 and 1, as the ROC evaluation method only takes numeric input.

Label Output	**Score Output**
$\mu \pm k\sigma$	HBOS (static)
Boxplot Rule	HBOS (dynamic)
Incomplete Clustering	LOF
	Cluster Centroid Distance
	Isolation Forest
	Stream-Neighborhood-based Method
	kth Nearest Neighbor Distance
	FPOF

Table 3.1: List of Anomaly Detection Methods By Type of Output

	Static Detection Methods			**Stream-based Detection Methods**
	Point-based Anomalies	**Context-based Anomalies**	**Collective Anomalies**	**Context-based Anomalies**
Nearest Neighbor-based		• LOF †‡ • k Nearest Neighbor distance †‡		• Stream-Neighborhood-based Method †
Statistic-based	• $\mu \pm k\sigma$ † • Boxplot Rule †			
Clustering-based	• Incomplete Clustering ‡	• Cluster-Centroid Distance ‡		
Histogram-based			• HBOS (static) ‡ • HBOS (dynamic) ‡	
Ensemble-based	• Isolation Forest ‡			• Isolation Forest ‡
Frequent Pattern Mining	•FPOF ‡			• FPOF ‡
‡ multivariate † univariate				

Table 3.2: A Categorization of the Introduced Detection Methods

4 EVALUATION

In this chapter, we bring the previously described methods to use and conduct out experiments. We first describe the full experimental setup, then we will show and discuss the results of the experiments. All in all, five experiments were conducted. The focus lies on the first two experiments, where we analyze the ROC AUC first for the univariate, then for the multivariate detection methods. The third experiment is about the similarity of the detected anomalies between all applied detection methods. In the fourth experiment, we make changes to the detection window size and examine the effect on the ROC AUC. The fifth experiment tackles the requirement of finding detection methods with a runtime of lower than one minute for a single estimation. The time measurement also gives a hint of the parameter dependency of the calculation effort.

4.1 EXPERIMENTAL SETUP

In order to describe the experimental setup, we first elaborate the data set in use and its labels. Then some information and analysis about the distribution of labels will follow.

4.1.1 Data Set

Here we provide some information about the data set origin, the feature selection process based on correlation and give some details about labels.

Data Origin

The data set is provided by the company SQL Projekt Ag. They monitored database servers systems and collected different properties from the cpu (cpu), the machine state (monstate), from various caches for each host (tempdb cache and defaultdata cache). To reduce the amount of data the database operations were aggregated to a one minute granularity.

SQL Projekt Ag identified two host systems as important. As we use these hosts in this thesis, but have to keep them anonymous, we will call them Host I and Host II.

Feature Selection
The provided data had to be optimized for effective anomaly detection. The better the quality of the underlying data, the more accurate is the anomaly diagnosis. First, we omitted uninformative data (e.g. cpu idle and column ids) as they do not provide additional information. Further on, we omitted data carrying too few information. Time series with a standard deviation close to zero were excluded, since they do not provide significant information. Usually one of the next steps is to remove incomplete data. But as some time series have their start point five months later than the other time series, we first checked the time series for their importance for the anomaly estimation by calculating the correlation over the common time period shown in Table 4.1. Additionally, we calculated the correlation of the labels to the time series, shown as the two bottom rows and, since the table is symmetric, the two last columns on the right of Table 4.1. For this purpose, we have created two temporary time series from the labels. One has the anomalies as zero and the normals as one, the other artificial time series has it reversed. This approach is made possible by the nature of our data. According to the statement by SQL Projekt Ag, the anomalies are noticeable by increases in cache, cpu load, and monstate data. We can see, that the artificial time series with 1 as anomalies (Peaks) correlates the most with the time series user_busy, io_busy and Rollbacks,while the artificial time series with anomalies as 0 (Lows) has negative correlations with the other time series.
Building upon this, we have experimented with the time series having a label correlation higher than 0.03. Different combinations for multivariate data were formed, multivariate anomaly methods were applied and the ROC AUC was examined. As a result, we remained with the Rollbacks and Inserts from the monstate data, as their ROC AUC together with io_busy and user_busy was the highest. Using TempDbObjects did not increase the ROC AUC.
We also examined the cache data. It did not contain any correlation with the labels or with any other time series, hence it was left out. Finally, we tackled the problem of unequal time series length. While the four chosen time series have equal length for Host II, the monstate data for Host I starts five months later. We omitted these five months also from io_busy and user_busy for Host I. Hence, the data streams from Host I and Host II in use have different length.

Annotated Labels
We received anomaly annotations for both hosts to evaluate the detection performance. In total 2329 of the 402.784 data points are annotated as anomalies for Host I, which is 0.00578%. Host II in comparison has 168 annotated anomalies for 623.044 data points, which is only 0.00027% of the full time series.
In Figure 4.1, the four univariate time series are shown using a correlation plot matrix described in Section 3.1.3. If you look at the distributions on the diagonal, you see there is a distinction in distributions between the anomalies (red) and the normals (blue) for user_busy and io_busy in the data. Hence, it should be possible for us to have detection methods, which make this distinction for us.
The data points of Rollbacks and Inserts are more scattered and seem only to cluster around zero, hence the distribution plot is not so meaningful in this representation.
In the scatter plots, we see that a separation between anomalies and normals is present, but not very prominent. In each of the scatter plots, the anomalies intermingle with the normal data points. This will have effect on the ROC AUC of the detection methods, as false estimations are likelier. However, as we cannot assess the data in four dimensional space, we can not examine the full interplay of the variables and cannot make any certain statements.
Since we have outlying clusters of normals and anomalies present in the data set, the detection

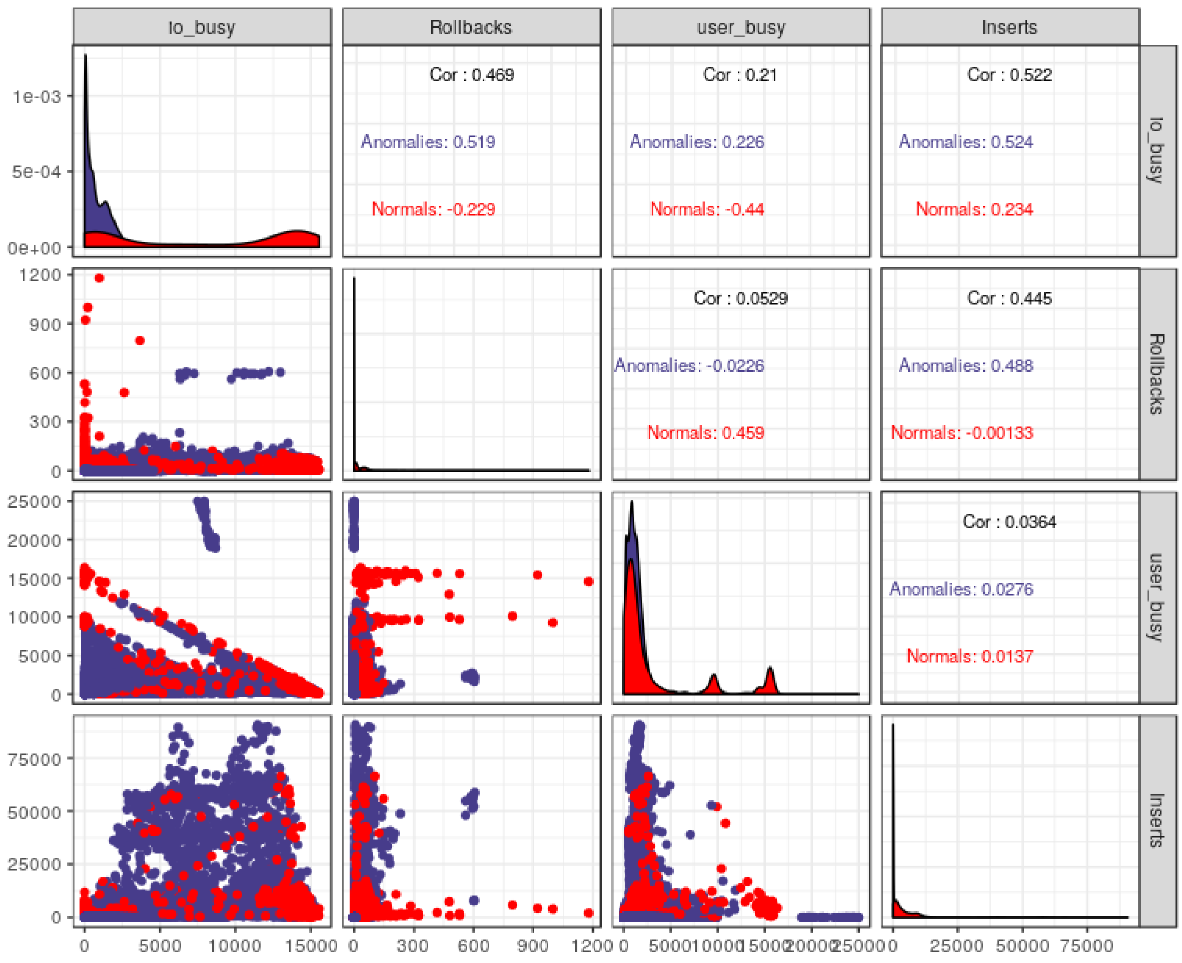


Figure 4.1: Pairwise Correlation Plots for the Four Essential Undecomposed Time Series of Host I

performance for the anomaly detection methods vulnerable to them (e.g. LOF, HBOS) will probably be lower. The upper right half of the plot concerns the pair-wise correlations. Besides the calculated pearson correlation from Table 4.1, the pairwise correlation between the normal and anomalous data points is shown. Here we see Rollbacks, Inserts, and io_busy have a high correlation between the anomalies, what could explain their contribution to the improved ROC AUC.

	user_busy (cpu)	**io_busy (cpu)**	**Inserts (monstate)**	**Updates (monstate)**	**ULCFlushFull (monstate)**	**Deletes (monstate)**	**Rollbacks (monstate)**	**NumDeadlocks (monstate)**	**Connections (monstate)**	**TempDb Objects (monstate)**	**PhysicalReads (defaultdata cache)**	**LogicalReads (defaultdata cache)**	**PhysicalReads (tempdb cache)**	**LogicalReads (tempdb cache)**	**Correlation Anomaly Labels to Peaks**	**Correlation Anomaly Labels to Lows**
user_busy	**1.00**	0.21	0.04	0.03	0.02	0.03	0.05	0.06	0.26	0.01	0.15	0.07	0.01	0.05	**0.17**	-0.11
io_busy	0.21	**1.00**	**0.52**	**0.43**	**0.37**	**0.52**	**0.47**	0.03	0.23	**0.33**	0.07	0.15	0.02	0.03	**0.12**	-0.16
Inserts	0.04	**0.52**	**1.00**	**0.91**	**0.31**	**0.46**	**0.45**	-0.01	0.08	0.25	-0.02	0.08	0.01	0.00	0.04	-0.09
Updates	0.03	**0.43**	**0.91**	**1.00**	0.28	**0.39**	**0.44**	-0.01	0.06	0.21	-0.02	0.05	0.01	0.00	0.03	-0.08
ULCFlushFull	0.02	**0.37**	**0.31**	0.28	**1.00**	**0.74**	0.21	0.00	0.06	0.16	-0.01	0.06	0.00	0.00	0.02	-0.06
Deletes	0.03	**0.52**	**0.46**	**0.39**	**0.74**	**1.00**	**0.31**	0.00	0.09	0.24	-0.01	0.09	0.01	0.01	0.03	-0.08
Rollbacks	0.05	**0.47**	**0.45**	**0.44**	0.21	**0.31**	**1.00**	-0.01	0.07	**0.33**	-0.04	0.05	0.01	0.00	**0.15**	-0.20
NumDeadlocks	0.06	0.03	-0.01	-0.01	0.00	0.00	-0.01	**1.00**	0.03	-0.01	0.02	0.01	0.00	0.01	0.00	0.00
Connections	0.26	0.23	0.08	0.06	0.06	0.09	0.07	0.03	**1.00**	0.05	0.13	0.06	0.02	0.05	0.00	-0.02
TempDbObjects	0.01	**0.33**	0.25	0.21	0.16	0.24	**0.33**	-0.01	0.05	**1.00**	-0.02	0.04	0.00	0.00	0.04	-0.07
PhysicalReads I	0.15	0.07	-0.02	-0.02	-0.01	-0.01	-0.04	0.02	0.13	-0.02	**1.00**	0.13	0.10	0.01	0.01	0.01
LogicalReads I	0.07	0.15	0.08	0.05	0.06	0.09	0.05	0.01	0.06	0.04	0.13	**1.00**	0.01	0.04	0.00	0.00
PhysicalReads II	0.01	0.02	0.01	0.01	0.00	0.01	0.01	0.00	0.02	0.00	0.10	0.01	**1.00**	0.01	0.00	0.00
LogicalReads II	0.05	0.03	0.00	0.00	0.00	0.01	0.00	0.01	0.05	0.00	0.01	0.04	0.01	**1.00**	0.00	0.00
Anomaly Labels to Peaks	**0.17**	**0.12**	0.04	0.03	0.02	0.03	**0.15**	0.00	0.00	0.04	0.01	0.00	0.00	0.00	**1.00**	-1.00
Anomaly Labels to Lows	-0.11	-0.16	-0.09	-0.08	-0.06	-0.08	-0.20	0.00	-0.02	-0.07	0.01	0.00	0.00	0.00	-1.0	**1.00**

Table 4.1: Correlation Table for each Time Series in Question of Host I

Methods Using Only Reference Window for Learning	**Methods Using Reference and Detection Window for Learning**
$\mu \pm k\sigma$	HBOS (static/dynamic)
Boxplot Rule	Isolation Forest (normal/ stream-based)
Cluster Centroid Distance	Stream-Neighborhood-based Method
Incomplete Clustering	LOF
kth Nearest Neighbor Distance	Stream-Neighborhood-based Method
	FPOF

Table 4.2: Categorization of the Implemented Detection Methods and Their Used Window Technique

4.1.2 Implementation of Methods

As we have a completed data set with label annotations, we simulated the stream-based scenario by feeding 'new' unlabeled incoming data to the detection method. The resulting anomaly scores had their optimal thresholds chosen by the ROC measure, the resulting labels were matched with their gold standard. Finally, the detection performance was calculated over the full time series. We decided to chose the two window technique from Figure 2.8f, as it fits the requirement for an early warning system best. For our scenario, we require only one estimation for a time point and also need data for the detection method to train on.
Using the style of the two window setup, we are able to apply originally static detection methods on the stream. The use of static methods allows us to use more diverse detection methods and also to apply established and approved methods beside the newer stream-based methods.
As some static detection methods require the data points for detection to be part of the calculation process (e.g. HBOS), we adapted the two window method in these cases. The adapted version uses a combination of reference window and detection window for learning. Which method uses which kind of window method is listed in Table 4.2. As shown, are the statistical and clustering-based methods using the separated window, while the other techniques use the joint window version. The joint window version might have the side effect, that the data points up for detection also influence the detection of the other data points. Hence, depending on the detection window size, the anomaly diagnosis might change. However, as we work with very small detection window sizes in comparison to the full data set, we assume this effect to be small.
For the implementation of the detection methods described in Section 3.2 and Section 3.3, we made use of the programming language R. As shown in Table 4.3 existing implementations for Isolation Forest, FPOF, and LOF came to use. Other methods as Incomplete Clustering, Cluster Centroid Distance or kth Nearest Neighbor Distance are based on supportive packages outside of the anomaly detection context. The remaining methods were self implemented in R.

4.1.3 Hardware

The code of this thesis was implemented and tested on an X1 Carbon with an i5-3427U and 8GB memory on a Ubuntu 18.04.2 LTS. The experiments were conducted on Linux Server with two Six-Core AMD Opteron Processor 2435 and 33 GB memory on a Ubuntu 16.04.5 LTS using 11

Detection Method	Code Origin
$\mu \pm k\sigma$	self-implemented in R
Boxplot Rule	self-implemented in R
LOF	DDoutlier package v.0.1.0 in R
Incomplete Clustering	self-implemented using dbscan package v.1.1-3 in R
Cluster Centroid Distance	self-implemented using flexclust package v.1.4-0 in R
k Nearest Neighbor Distance	self-implemented using FNN package v.1.1.3 in R
HBOS (static)	self-implemented in R
HBOS (dynamic)	self-implemented in R
Isolation Forest	isofor package v.1.0.0 in R
Stream-Neighborhood-based Method	self-implemented in R
FPOF	fpmoutliers package v.0.1.0 in R

Table 4.3: Code Origin for Applied Detection Methods

of the 12 cores. The code was parallelized using the 'doSNOW' package version 1.0.18. In both systems R in version 3.5.3 was used.

4.2 ANALYSIS OF ROC AUC

This experiment examines the ROC AUC of the detection methods. In total, we examine five methods performing on the univariate data, including three solely univariate methods ($\mu \pm k\sigma$, Boxplot Rule, Stream- Neighborhood-based Method) and two multivariate methods (LOF, kth Nearest Neighbor Distance), which also allow univariate input. Furtheron, we examine eight multivariate detection methods (LOF, Incomplete Clustering, Cluster Centroid Distance, kth Nearest Neighbor Distance, HBOS(static/dynamic), FPOF) performing on the multivariate data, which was combined from the io_busy, user_busy, Inserts, and the Rollbacks time series. To perform this experiment, we used the data described in Section 4.1.1 and simulated the stream-based scenario. We chose a constant reference window size of three weeks to predict the anomaly scores for each method. For Host I are 372.544 estimations required. We applied a weekly decomposition on the combination of reference window and detection window and performed the anomaly estimation on the remainder. Additionally, we applied a grid search with varying search ranges to find a maximum ROC AUC for each detection method. As the search grids had to be limited, finding global maxima is not guaranteed. For each method, we selected a threshold with the smallest Euklidean distance to the optimum point of 100% TPR and 0% FPR of the ROC curve. That means, we did not prefer either a high TPR or a low FPR. Depending on the preferences, another selection is possible.
To ensure the reproducibility of the experiments, the optimal parameters and threshold obtained via ROC curve and grid search are listed in Table 4.4 for the univariate methods and in Table 4.5

for the multivariate methods. We have depicted the ROC curves, from which the ROC AUC values are derived, in Figure 4.2 and Figure 4.3. For the univariate data in Table 4.4, we can see that the most suitable time series for anomaly estimation has shown to be Rollbacks. The ROC AUC results for this time series all surpass the results for io_busy, and user_busy. For the Rollbacks results, we see in Table 4.4 and Figure 4.2, that the $\mu \pm k\sigma$ method and the Boxplot Rule, even by being very similar in their technique, achieved slightly different results. This can be caused by the different use of the mean or median, or by the coarse parameter grid with size 0.5, through which the actual maxima were not found.
In comparison, the LOF method did not manage a single run through a full time series in three days. We assume that its complexity did not scale well with our large number of anomaly estimations. The Stream-Neighborhood-based Method did achieve low to mediocre results, this can be cause by a too small search grid. The parameter space for this method is very large. However, due to the high runtime, a sufficient search would be very time-consuming. While a single time series run through takes approximately forty minutes, testing only thirty different parameter combinations takes twenty hours, what exceeds the capacities of this work.
The best univariate method for each time series has shown to be the *k*th Nearest Neighbor method with a ROC AUC of 96.69% points on Rollbacks. In Figure 4.2, we see that it performs very similar to the $\mu \pm k\sigma$ method and the Boxplot Rule.
For the multivariate methods shown in Table 4.5 and Figure 4.3, we had problems with the calculation times of some methods. The computation times for FPOF, Cluster Centroid Distance, LOF, and for some parameters HBOS (static) were too high and exceeded three days of computation. The parameter optimizations for these calculations had to be aborted. However, from the remaining methods, some achieved high ROC AUC. The *k*th Nearest Neighbor Distance method exceeds with 95.55 % points. The second best method is the dynamic HBOS method with 95.39 % points. The higher ROC AUC for the dynamic method compared to the static method is likely caused by the flexible nature of the dynamic HBOS method, as it can adapt to the data, while the static method cannot. Other methods, as the Isolation Forest or HBOS(static), achieved lower ROC AUC. Due to the higher calculation time of multivariate methods compared to univariate detection methods, the search grid for these methods were coarser as well. The same problem existed with the Incomplete Clustering method. It is the only method, which achieved bad results with a ROC AUC of 51.82 % points. It is not capable to manage a separation close to the labels and resembles a random estimator. The bad performance is caused by the insufficient parameters. The search grid had to be limited due to its very high calculation times. We used the common method to compute the epsilon parameter. The different distances between the data points for several three week windows as well as for the full data set were plotted and the the knee points were determined. However, none of the received epsilon parameters resulted in a ROC AUC higher than 51.82%. Very likely, we were not able to detect the sweet spot for this method.
To conclude, we have seen that the best performing methods come from different method families and also deliver score as well as label output. The best performing methods include scoring methods, which use data point neighborhoods for estimation(*k*th Nearest Neighbor Distance method, Stream-Neighborhood-based method), and also labeling and scoring methods which apply splits on the value range (dynamic HBOS, $\mu \pm k\sigma$, Boxplot Rule). Hence, we found a whole set of different techniques, which perform well on database server data. We cannot advise a single family of techniques, but several of them: Histogram-based methods, Nearest Neighbor-based methods, as well as statistical detection methods performed well in this thesis. The multivariate methods, performed less efficient, probably due to too few parameter fittings, as their calculation time was higher.

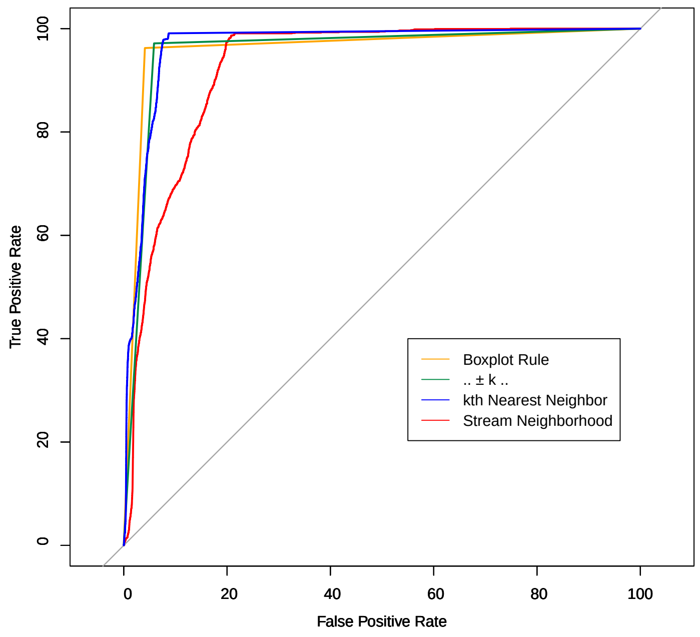


Figure 4.2: ROC Curves for the Univariate Detection Methods on the Rollbacks Time Series

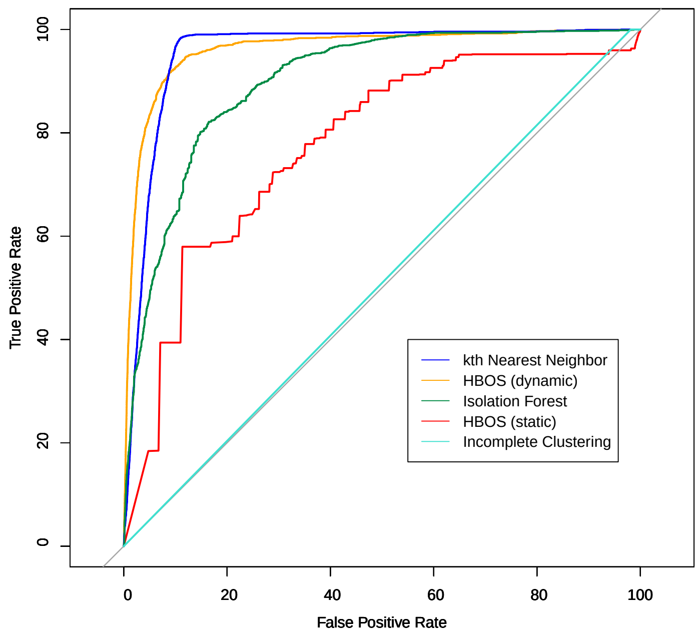


Figure 4.3: ROC Curves for the Multivariate Detection Methods

Detection Method	Parameter Search Range	Optimal Parameter			Parameter Description	Chosen Threshold			Calculated ROC AUC		
		io_busy	user_busy	Rollbacks		io_busy	user_busy	Rollbacks	io_busy	user_busy	Rollbacks
$\mu \pm k\sigma$	(0.5, 1.0, 1.5,..., 40)	0.5	40.0	0.5	Factor for Standard Deviation	0.5	0.5	0.5	70.63	50.00	95.64
Boxplot Rule	(0.5, 1.0, 1.5,..., 30)	1	0.5	0.5	Factor for IQR	0.5	0.5	0.5	74.44	57.61	96.08
LOF	(1, 2, 3, ... 30)	-	-	-	kth Neighbor	-	-	-	-	-	-
kth Nearest Neighbor Distance	(1, 2, 3, ..., 30)	29	17	20	kth Neighbor	1.856	0.29	0.0075	90.15	62.70	96.69
Stream-Neighborhood-based Method	(1, 20, 40, ... 200), (100, 200, 300, ... 1000)	1, 100	1, 100	1, 400	Neighbor Count, Neighborhood Width	18321	1475.5	12323.5	90.42	65.46	92.53

Table 4.4: List of AUC-Optimized Parameters for each Univariate Detection Method

Detection Method	**Parameter Search Range**	**Optimal Parameter**	**Parameter Description**	**Chosen Threshold**	**Calculated ROC AUC**
LOF	(1, 2, 3, ... 30)	-	Factor for Standard Deviation	-	-
Incomplete Clustering	(0.25, 0.35, 0.5) (4,6,8)	0.25 4	Neighborhood Radius, min. Cluster Size	0.5	51.82
Cluster Centroid Distance	(4,5,6,7,8)	-	Number of Clusters	-	-
*k*th Nearest Neighbor Distance	(1,2,3,.., 30)	2	kth Neighbor	48.848	95.55
HBOS (static)	(100, 200, 300)	100	Count of Bins	-9.96812	83.04
HBOS (dynamic)	(10, 20, 30, ... 100)	10	Count of Values per Bin	4.6344	95.39
FPOF	-	-	Minimum Support	-	-
Isolation Forest	(10, 20, ..., 100) (10)	90 10	Count of Trees, Samples per Tree	0.55	82.44

Table 4.5: List of AUC-Optimized Parameters for each Multivariate Detection Method

4.3 SIMILARITY IN DETECTED ANOMALIES

In this experiment we examine the similarity between the detected anomalies from the previous experiment. Hence, the methods use the parameters and thresholds listed in Table 4.4 and Table 4.5. It is conducted on the univariate and multivariate data. For univariate data, we used only the Rollbacks time series as it performed best in the first experiment. We exam to create a heatmap. The used similarity measure is shown in Equation (4.1). We calculate the number of time points being detected by both methods m and n as anomalies in a data set I and divide it by the number of elements in the data set $|I|$:

$$sim(m, n) = \frac{m(I) \cap n(I)}{|I|} \tag{4.1}$$

Note that this examination does not work based on labels and hence, it does not handle true or false positives. The resulting heatmap, shown in in Figure 4.4, can give information about two aspects. For one, we can examine how many anomalies a detection method found at all, and compare it with the number of anomalies in our labels. Secondly, we can examine how similar two detection methods are, by examining the similarity in the detected anomalies. All in all, we expect the detection techniques of the same detection method families to resemble each other. As some methods are based on similar assumptions to detect anomalies, we expect to find similarity also between methods of different method families. When considering the labels, we know that 2.340 of the 372.544 data points are anomalies, which is 0.00628% of the data set.

The diagonal of the heatmap shows for each method the percentage of detected anomalies in the data set. The darker the color, the fewer anomalies has the detection method found. Most methods detect less than 1% of the data points as anomalies. Incomplete clustering, recognizable by the yellow field in the top right corner, does not have a good parameter configuration. It detected too many data points as anomalies, that is probably the reason why it scored a low ROC AUC. Here, we see it detected 1.914% of the data point as anomalies and thus differs greatly from the other methods. The static HBOS method has the same problem. With 0.692% points, the method has the second most anomalies. It also suffered from low ROC AUC.
We can also see on the diagonal, that the best scoring detection methods, $\mu \pm k\sigma$ (0.062%), the Boxplot Rule (0.044%) and the kth Nearest Neighbor Distance method (0.080%) detected the fewest anomalies and are the closest to the 0.00628% of the labels.

When examining the similarities between the methods, it is possible to see that the similarity depends on the detected anomalies of a method. Where fewer anomalies are detected, as in the Boxplot Rule, the overlap with other detection methods is lower, because they can share less common anomalies. As you can see with the incomplete clustering method, where many anomalies are detected, the similarity with other detection methods is likelier. Also, the opposite is possible. For example, the static HBOS method has only few in common with methods besides the Incomplete Clustering, even by detecting 0.692% as anomalies.
When examining the univariate methods at the bottom left of the heatmap, we see that the $\mu \pm k\sigma$, the Boxplot Rule, the kth Nearest Neighbor Distance method and the Stream Neighborhood-based method are very similar. The methods find almost the same anomalies as the Boxplot Rule (0.43%

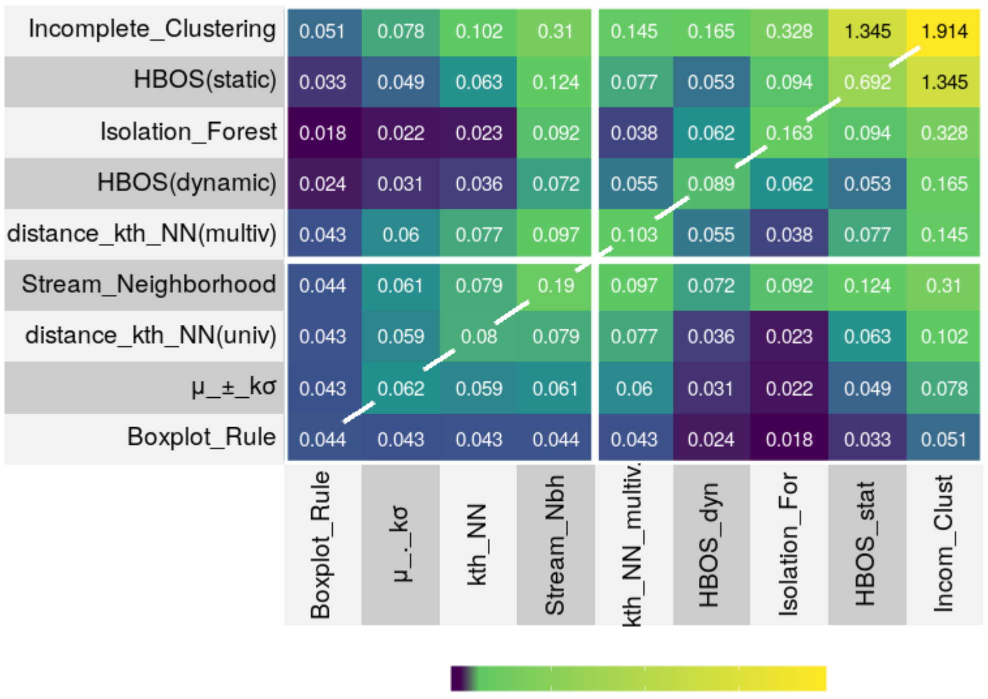


Figure 4.4: Heatmap for the Similarity of Detected Anomalies Between all Detection Methods, as in Equation (4.1), in Percent

of 0.44%), and also detect some additional datapoints as anomalies. Even in these additional anomalies the remaining methods are very similar and only deviate by 0.003% points. When considering the multivariate methods as well, we can detect, that the *k*th Nearest Neighborhood Distance method performs very similar to the univariate methods, leaving the bad performing Incomplete Clustering method aside.
When comparing the detected anomalies by the *k*th Nearest Neighbor Distance method for univariate and multivariate data, another peculiarity becomes visible. Their overlap of 0.077% points is unexpectedly low for the multivariate version. It detected 0.103% of data points as anomalies, compared to the univariate method with 0.08% of anomalies. This difference is probably caused by the additional dimensions in multivariate data, which influences the neighborhoods of data points. This difference, might also be the cause for the different ROC AUC for multivariate and univariate data. In our first experiment, the method achieved 96.69 % points on univariate data, but only 95.55 % points on multivariate data. With this, the univariate methods outperform the multivariate methods by only 1.14 % points for the chosen search grids.
When assessing the similarities between the multivariate data only, the results are not as clear as for the univariate data due to less parameter fitting and hence less optimal detection of anomalies. We see that the static HBOS, the dynamic HBOS and the Isolation Forest method have less in common with the univariate methods, but are more similar to each other. The Isolation Forest method can find close to half of the anomalies detected by the dynamic and static HBOS. This similarity can be caused by their similar fashion, as each method splits the value range for score calculation. Isolation Forest does it by random, while the HBOS methods work in a more prescripted fashion. However, the similarities are by far not as strong as in the univariate methods.

4.4 EFFECTS OF THE DETECTION WINDOW SIZE ON THE ROC AUC

In this experiment, we examine the effect of changes on the detection window size. A detection window size of one, as in the other experiments, assumes a minutely anomaly estimation interval. We check here, if making halfhourly, hourly, daily and weekly anomaly estimations, changes have effect on the estimation results. While minutely estimations require high calculation time, larger detection window are also relevant for use in practice, as the estimations are fewer, less calculation expensive and are faster in the optimization process.
As data for the univariate methods, we used the Rollbacks time series, as it performed best in the first experiment. As methods, we chose techniques with the lowest runtime. Using all detection methods for examination would be very time expensive. The reference window was fixed to three weeks, as before. The search grid is the same as shown in Table 4.4 and Table 4.5. The results for this experiment are shown in Table 4.6. In the last column, we see the different ROC AUC. For each window size for each method the ROC AUC is above 90% points, the results for each method lie close to each other. The maximal standard deviation has the Boxplot Rule with 1.56% points, while the *k*th Nearest Neighbor has the lowest standard deviation of 0.46 % points. These deviations in detection performance can be caused by the decomposition, which produces with increasing amount of data different separations between remainder, season and trend. Also it is possible, that the coarse search grid resulted in non-optimal results for each method. The values found are therefore only an approximation of the optimum.
The highest ROC AUC for each method is achieved with a detection window size of 10080 for the

Detection Method	Detection Window Size	Best Parameters	AUC in %
$\mu \pm k\sigma$	10080 (1 Week)	k = 0.5	95.56
	1440 (1 Day)	k = 8	95.84
	60 (1 Hour)	k = 4	98.51
	1 (1 Minute)	k = 0.5	95.64
Boxplot Rule	10080 (1 Week)	k = 0.5	95.56
	1440 (1 Day)	k = 24	98.18
	60 (1 Hour)	k = 5.5	98.71
	1 (1 Minute)	k = 0.5	96.08
kth Nearest Neighbor Distance	10080 (1 Week)	k = 30	97.66
	1440 (1 Day)	k = 16	97.37
	60 (1 Hour)	k = 26	96.80
	1 (1 Minute)	k = 20	96.69

Table 4.6: ROC AUC in % for each Univariate Detection Method on Host I Rollbacks

*k*th Nearest Neighbor distance method, making weekly estimations, and of 1440 for the statistical methods $\mu \pm k\sigma$ and the Boxplot Rule. However, as the differences between a minutely and a weekly estimation are minor, we can advise to use detection window sizes higher than one if required.

4.5 RUNTIME MEASUREMENT

This experiments concerns the required runtime for the experiment in Section 4.2. As we want to perform the anomaly estimations close to real time, we have the requirement of using less than a minute for a single prediction. Additionally, runtime plays an essential role for parameter optimization. The longer a single estimation takes, the longer takes a run through the full data set.
For this experiment, we analyzed all runtimes for the search grid in Section 4.2 from the first experiment. We collected the minimum and maximum for each detection method on io_busy, user_busy, Rollbacks and the multivariate data set. This way, we can examine the behavior of methods on different time series, as well as parameter dependencies of the runtimes. Using minimum and maximum as a measure allows with low effort to represent viable ranges of possible runtimes for each method. By using averages, information would get lost. Since the measured values depend on the hardware (see Section 4.1.3), the reproducibility of this experiment is limited. As described in the previous experiments, we limited the calculation times to three days for a single time series run through. The results for the univariate methods are shown in Section 4.5 and for the multivariate methods in Section 4.5.
In the table for the univariate data, we see that most methods require far less than a second for a single estimation. Only LOF did not terminate within three days, thereby we can only estimate a single prediction to take longer than 0.696 seconds. The other methods have slight differences in their runtime for each time series, hence we assume them to be runtime dependent of the used time series. Additionally, we can see, that for each method the differences between minimum and

maximum are on each time series below two minutes. Leaving LOF aside, we conclude, that none of the univariate methods is parameter dependent concerning the runtime.
For the multivariate methods, we have three methods which did not terminate within three days: LOF, Cluster Centroid Distance and FPOF. The remaining methods show varying behavior in runtime. The *k*th Nearest Neighbor Distance runtime is not parameter dependent, as its calculation time barely varies with different parameters. In comparison, the Isolation Forest has variations of close to two hours. The dynamic HBOS method has variations of up to fifteen hours. While the Incomplete Clustering method has a variation of up to one and a third day for our chosen set of parameters. For the static HBOS method, the variations were high enough to exceed our limit of three days for some parameters. With these high variations and longer runtimes of at least two hours, the multivariate methods have shown to be time consuming and calculation expensive. This is affects the parameter optimizations and increases the required time drastically.
All in all, we observed that the univariate detection method, as well as the *k*th Nearest Neighbor Distance Method for Multivariate data, have shown to be of shorter and consistent runtime, than the other multivariate anomaly detection methods.

Detection Method	io_busy		user_busy		Rollbacks	
	min	max	min	max	min	max
$\mu \pm k\sigma$	40.20 mins 0.0065 sec	41.02 mins 0.0066 sec	36.60 mins 0.0059 sec	38.02 mins 0.0061 sec	36.11 mins 0.0058 sec	36.21 mins 0.0058
Boxplot Rule	37.49 mins 0.0060 sec	39.03 mins 0.0063 sec	40.06 mins 0.0064 sec	40.27 mins 0.0065 sec	39.17 mins 0.0063 sec	39.31 mins 0.0063 sec
LOF	> 3 days > 0.696 sec	> 3 days > 0.696 sec	> 3 days > 0.696 sec	> 3 days > 0.696 sec	> 3 days > 0.696 sec	> 3 days > 0.696 sec
kth Nearest Neighbor Distance	48.69 mins 0.0078 sec	48.83 mins 0.0078 sec	50.94 mins 0.0082 sec	51.61 mins 0.0083 sec	45.78 mins 0.0073 sec	45.90 mins 0.0074 sec
Stream-Neighborhood-based Method	35.98 mins 0.0058 sec	37.81 mins 0.0061 sec	36.81 mins 0.0059 sec	38.18 mins 0.0061 sec	37.98 mins 0.0061 sec	38.07 mins 0.0061 sec

Table 4.7: Maximum and Minimum Calculation Times for Full Time Series Run Throughs and for Single Anomaly Estimations of each Univariate Detection Method

Detection Method	min	max
LOF	> 3 days > 0.696 sec	> 3 days > 0.696 sec
Incomplete Clustering	16.14 hrs 0.156 sec	2 days 0.463 sec
Cluster Centroid Distance	> 3 days > 0.696 sec	> 3 days > 0.696 sec
*k*th Nearest Neighbor Distance	2.20 hrs 0.021 sec	2.25 hrs 0.022 sec
HBOS (static)	1.04 days 0.244 sec	> 3 days > 0.696 sec
HBOS (dynamic)	2.57 hrs 0.025 sec	15.99 hrs 0.154 sec
Isolation Forest	2.52 hrs 0.024 sec	5.72 hrs 0.055 sec
FPOF	> 3 days > 0.696 sec	> 3 days > 0.696 sec

Table 4.8: Maximum and Minimum Runtimes for Full Time Series Run Throughs and for Single Anomaly Estimations of each Multivariate Detection Method

5 CONCLUSION AND OUTLOOK

This thesis has examined the performance of various anomaly detection techniques on database server data to create an early warning system. The early detection of anomalies in sever loads allows fast treatment, for example through reboots, and avoids service loss and customer dissatisfaction. To find the best fitting anomaly detection method for such problems, we simulated a stream-based scenario on four important time series of a database server: the three univariate time series io_busy, user_busy and Rollbacks, as the multivariate time series consisting of io_busy, user_busy, Rollbacks and Inserts. We tested several detection methods from different method families and determined the ROC AUC values based on provided labels for each method. This measure allows to find anomalies with high TPR, but low FPR. A successful detection of the anomalies was possible.

In total, five methods for the univariate data and eight methods for the multivariate data were investigated. We applied a two-parted sliding window, which used three weeks of reference data to estimate the anomaly score for the next minute. Each time series required in total 372,544 anomaly estimations. Unfortunately, three multivariate methods and one univariate method exceeded the calculation time limit of three days for an anomaly estimation on a full time series. Hence, no result are available for these methods. The best time series for univariate estimations has shown to be the Rollbacks time series. Probably, it is the best indicator for an erroneous server state, as it provides the highest degree of selectivity for the detection methods. For the univariate data, best performed the *k*th Nearest Neighbor Distance method with a ROC AUC of 96.69 % points on the univariate Rollbacks data stream. The Boxplot Rule and the $\mu \pm k\sigma$ method scored similar results. The univariate methods had all similar runtimes for detection and required close to 40 minutes for a pass through a full time series. Here, only slight dependencies on the underlying time series were noted.

The multivariate methods did not exceed the univariate methods in ROC AUC. The multivariate *k*th Nearest Neighbor Distance method scored 95.55% points. While, the dynamic HBOS method scored second best with 95.39% points. Some methods, as the Incomplete clustering and the static HBOS method, scored lower ROC AUC. Due to high runtimes, the parameter optimizations for these methods have been limited to a small quantity.

The multivariate methods required for a full time series estimation more than two hours. Only the *k*th Nearest Neighbor Distance method was parameter independent. Other methods, as the static HBOS method, exceeded for some parameters the 3 day limit. In total, the univariate data

outperformed the multivariate data by a difference in ROC AUC of 1.14 % points. Better results for the multivariate data are likely, as their search grids had to be kept very coarse due to their longer runtimes and higher counts of parameters. Unfortunately, the time limits of this thesis did not allow a more detailed search. Also, when we chose a set of time series as the multivariate dataset, we chose time series which resulted in the highest ROC AUC for most detection methods. However, this does not mean to be the best selection for the *k*th Nearest Neighbor Distance method. A better performance with a different set of time series is likely.
Compared to the labels, we had to conclude, that all detection methods detected too many anomalies. According to the labels, only 0.00628% of the data points were actual anomalous. The closest came the Boxplot Rule with 0.044% found anomalies. To recall, the method achieved with 96.08% points a high ROC AUC score as well. Further on, we noted a high similarity in the detected anomalies between the univariate detection methods and the the multivariate *k*th Nearest Neighbor Distance method. These methods shared a big majority of their found anomalies and scored high ROC AUC, even though they come from different method families. Indeed, the statistical and the nearest neighbor-based detection methods scored best in the experiments.
We also made experiments in terms of practical application. For example, we examined the results from estimations with varying detection frequency. While the other experiments used only minutely estimations, we investigated here hourly, daily, and weekly estimations on the Rollbacks time series using the three fastest anomaly detection techniques. This has shown to be relevant, as minutely predictions are expensive in time and computation. Less frequent estimations may already be sufficient for some applications cases. There have shown to be minor deviations in ROC AUC, which are probably caused by the changing window sizes undergoing decomposition. Fortunately, we were able to determine that lower estimation frequencies result in similar ROC AUC as minutely estimations.
While this work has already covered some areas in the field of anomaly detection, there are many more possibilities in this domain. This thesis is part of an ongoing research process, hence we will highlight here some research topics, that would be interesting for further investigation.

Extensive Research in Nearest Neighbor-based Anomaly Detection Methods In this thesis, the *k*th Nearest Neighbor algorithm has shown to be the best performing on given data. It achieved the highest ROC AUC in the univariate data, and scored high in the multivariate data in consistent time. Not only the optimized parameters had high estimation performance, other parameters scored high as well and it is likely that the best global parameter was not yet found (see Appendix C). While the algorithm from the same family, LOF, was too calculation expensive, more evolved methods from this method family, as COF [TCFC02] or ODIN[HKF04], seem very promising to score even better. In this thesis, we chose LOF as a deputy. Adding more algorithms to the selection, would have needed many more calculations, what would have exceeded the time limits for this work.

Parameter Free Anomaly Detection Methods In this thesis, the eye of the needle were the long calculation times, which extended the parameter search by a multiple. Therefore, it is guaranteed, that a parameter free detection method would require less optimizations and with this it would deliver results much faster. Such methods were described in a few papers as in [KLR04]. The domain seems very promising, as these algorithms are solely data-driven. The research in this work has in this regard not yet led to satisfactory results, hence we advice a deeper leading research here.

Remove or Correct Detected Anomalies for the Use in the Next Anomaly Estimation The detection window method used for our experiments is prone to anomalies, as even detected anomalies in the reference windows can influence the estimations on the detection window. These anomalies can shift the mean or median or can be neighbors to other anomalies and hence form anomaly clusters, which stay undetected. Removing or correcting such detected anomalies, may lead to better anomaly detection. However, as this highly influences the detection methods, it requires additional research.

BIBLIOGRAPHY

[AF02] MF Augusteijn and BA Folkert. Neural network classification and novelty detection. *International Journal of Remote Sensing*, 23(14):2891–2902, 2002.

[AF07] Fabrizio Angiulli and Fabio Fassetti. Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 811–820. ACM, 2007.

[AP02] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer, 2002.

[AP14] Hermine N Akouemo and Richard J Povinelli. Time series outlier detection and imputation. In *2014 IEEE PES General Meeting | Conference & Exposition*, pages 1–5. IEEE, 2014.

[AS17] Charu C Aggarwal and Saket Sathe. *Outlier ensembles: An introduction*. Springer, 2017.

[AY01] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM, 2001.

[BGBMY01] Ana Maria Bianco, M Garcia Ben, EJ Martinez, and Víctor J Yohai. Outlier detection in regression models with arima errors using robust estimates. *Journal of Forecasting*, 20(8):565–579, 2001.

[BKNS00] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[BR98] Simon Byers and Adrian E Raftery. Nearest-neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442):577–584, 1998.

[CBK07] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Outlier detection: A survey. *ACM Computing Surveys*, 14:15, 2007.

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[CC19] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[CCMT90] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition. *Journal of official statistics*, 6(1):3–73, 1990.

[ÇDÇD11] Mete Çelik, Filiz Dadaşer-Çelik, and Ahmet Şakir Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*, pages 91–95. IEEE, 2011.

[CH$^+$01] Paul Crook, Gillian Hayes, et al. A robot implementation of a biologically inspired method for novelty detection. In *Proceedings of the Towards Intelligent Mobile Robots Conference*, 2001.

[DF13] Zhiguo Ding and Minrui Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17, 2013.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[DXLL09] Lian Duan, Lida Xu, Ying Liu, and Jun Lee. Cluster-based outlier detection. *Annals of Operations Research*, 168(1):151–168, 2009.

[EAP$^+$02] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[EKS$^+$96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[ESK02] Levent Ertoz, Michael Steinbach, and Vipin Kumar. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on clustering high dimensional data and its applications at 2nd SIAM international conference on data mining*, pages 105–115, 2002.

[Fuk80] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[GAG$^+$00] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[GD12] Markus Goldstein and Andreas Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: Poster and Demo Track*, pages 59–63, 2012.

[GGAH14] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.

[GMESK99] Sigurour E Guttormsson, RJ Marks, MA El-Sharkawi, and I Kerszenbaum. Elliptical novelty grouping for on-line short-turn detection of excited running rotors. *IEEE Transactions on Energy Conversion*, 14(1):16–22, 1999.

[GÖ03] Lukasz Golab and M Tamer Özsu. Issues in data stream management. *ACM Sigmod Record*, 32(2):5–14, 2003.

[GPT04] Pedro Galeano, Daniel Peña, and Ruey S Tsay. Outlier detection in multivariate time series via projection pursuit. 2004.

[GRS00] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[Gru69] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[GT06] Jing Gao and Pang-Ning Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 212–221. IEEE, 2006.

[HA04] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004.

[HB97] Paul Helman and Jessie Bhangoo. A statistically based system for prioritizing information exploration under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 27(4):449–466, 1997.

[HFLP01] Paul S Horn, Lan Feng, Yanmei Li, and Amadeo J Pesce. Effect of outliers and nonhealthy individuals on reference interval estimation. *Clinical Chemistry*, 47(12):2137–2145, 2001.

[HKF04] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE, 2004.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HVK17] Jordan Hochenbaum, Owen S Vallis, and Arun Kejariwal. Automatic anomaly detection in the cloud via statistical learning. *arXiv preprint arXiv:1704.07706*, 2017.

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[HXHD05] Zengyou He, Xiaofei Xu, Joshua Zhexue Huang, and Shengchun Deng. Fp-outlier: Frequent pattern based outlier detection. *Comput. Sci. Inf. Syst.*, 2(1):103–118, 2005.

[HZ94] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10, 1994.

[Jac00] William G Jacoby. Loess:: a nonparametric, graphical tool for depicting relationships between variables. *Electoral Studies*, 19(4):577–613, 2000.

[JV91] Harold S Javitz and Alfonso Valdes. The sri ides statistical anomaly detector. In *Proceedings. 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 316–326. IEEE, 1991.

[KHM+18] Martin Kropf, D Hayn, D Morris, Aravind-Kumar Radhakrishnan, E Belyavskiy, A Frydas, E Pieske-Kraigher, B Pieske, and G Schreier. Cardiac anomaly detection based on time and frequency domain features using tree-based classifiers. *Physiological measurement*, 39(11):114001, 2018.

[KKSZ11] Hans-Peter Kriegel, Peer Kroger, Erich Schubert, and Arthur Zimek. Interpreting and unifying outlier scores. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pages 13–24. SIAM, 2011.

[KL05] Eamonn Keogh and Jessica Lin. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and information systems*, 8(2):154–177, 2005.

[KLR04] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215. ACM, 2004.

[Koh97] Teuvo Kohonen. Exploration of very large databases by self-organizing maps. In *Proceedings of International Conference on Neural Networks (ICNN'97)*, volume 1, pages PL1–PL6. IEEE, 1997.

[KYL+16] Gyuwan Kim, Hayoon Yi, Jangho Lee, Yunheung Paek, and Sungroh Yoon. Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems. *arXiv preprint arXiv:1611.01726*, 2016.

[LA15] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 38–44. IEEE, 2015.

[LJK+00] Jorma Laurikkala, Martti Juhola, Erna Kentala, N Lavrac, S Miksch, and B Kavsek. Informal identification of outliers in medical data. In *Fifth international workshop on intelligent data analysis in medicine and pharmacology*, volume 1, pages 20–24, 2000.

[LLB10] Feng Lin, Wang Le, and Jin Bo. Research on maximal frequent pattern outlier factor for online high dimensional time-series outlier detection. *Journal of convergence information technology*, 5(10):66–71, 2010.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[LW91] Jen-Pei Liu and Chung-Sing Weng. Detection of outlying data in bioavailability/bioequivalence studies. *Statistics in Medicine*, 10(9):1375–1389, 1991.

[Mar98] Dominique Martinez. Neural tree density estimation for novelty detection. *IEEE Transactions on Neural Networks*, 9(2):330–338, 1998.

[NAG10] Hoang Vu Nguyen, Hock Hee Ang, and Vivekanand Gopalkrishnan. Mining outliers with ensemble of heterogeneous detectors on random subspaces. In *International Conference on Database Systems for Advanced Applications*, pages 368–383. Springer, 2010.

[NKSH09] Sami Nousiainen, Jorma Kilpi, Paula Silvonen, and Mikko Hiirsalmi. Anomaly detection from server log data. Technical report, 2009.

[PD12] Ms SD Pachgade and Ms SS Dhande. Outlier detection over data set using cluster-based and distance-based approach. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6), 2012.

[RL05] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

[RWHW09] Jiadong Ren, Qunhui Wu, Changzhen Hu, and Kunsheng Wang. An approach for analyzing infrequent software faults based on outlier detection. In *2009 International Conference on Artificial Intelligence and Computational Intelligence*, volume 4, pages 302–306. IEEE, 2009.

[SBE$^+$02] Rasheda Smith, Alan Bivens, Mark Embrechts, Chandrika Palagiri, and Boleslaw Szymanski. Clustering approaches for anomaly based intrusion detection. *Proceedings of intelligent engineering systems through artificial neural networks*, 9, 2002.

[She31] Walter Andrew Shewhart. Economic control of quality of manufactured product. 1931.

[SL05] Helge Erik Solberg and Ari Lahti. Detection of outliers in reference distributions: performance of horns algorithm. *Clinical chemistry*, 51(12):2326–2332, 2005.

[SR18] Mahsa Salehi and Lida Rashidi. A survey on anomaly detection in evolving data: [with application to forest fire risk prediction]. *SIGKDD Explor. Newsl.*, 20(1):13–23, May 2018.

[SSE$^+$17] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):19, 2017.

[SW$^+$98] C Surace, K Worden, et al. A novelty detection method to diagnose damage in structures: an application to an offshore platform. In *The Eighth International Offshore and Polar Engineering Conference*. International Society of Offshore and Polar Engineers, 1998.

[TCFC02] Jian Tang, Zhixiang Chen, Ada Wai-Chee Fu, and David W Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 535–548. Springer, 2002.

[TLC09] Xianghong Tang, Guohui Li, and Gang Chen. Fast detecting outliers over online data streams. In *2009 International Conference on Information Engineering and Computer Science*, pages 1–4. IEEE, 2009.

[TTL11] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for streaming data. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[VR13] William N Venables and Brian D Ripley. *Modern applied statistics with S-PLUS*. Springer Science & Business Media, 2013.

[Wei06] William WS Wei. Time series analysis. In *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*. 2006.

[WFP99] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE, 1999.

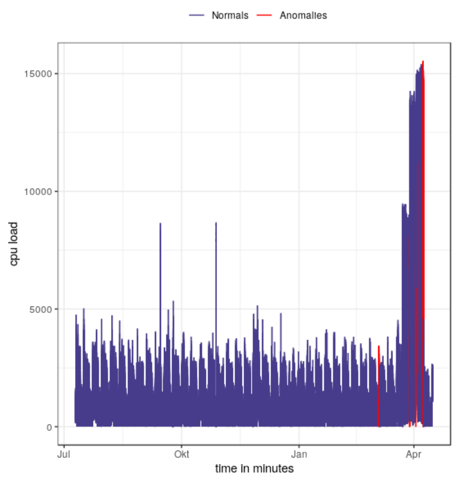
[WYK$^+$19] Yinan Wang, Ryota Yoshihashi, Rei Kawakami, Shaodi You, Tohru Harano, Masahiko Ito, Katsura Komagome, Makoto Iida, and Takeshi Naemura. Unsupervised anomaly detection with compact deep features for wind turbine blade images taken by a drone. *IPSJ Transactions on Computer Vision and Applications*, 11(1):3, Jun 2019.

[ZSZY07] Xiao-Yun Zhou, Zhi-Hui Sun, Bai-Li Zhang, and Yi-Dong Yang. Fast outlier detection algorithm for high dimensional categorical data streams. *Ruan Jian Xue Bao(Journal of Software)*, 18(4):933–942, 2007.

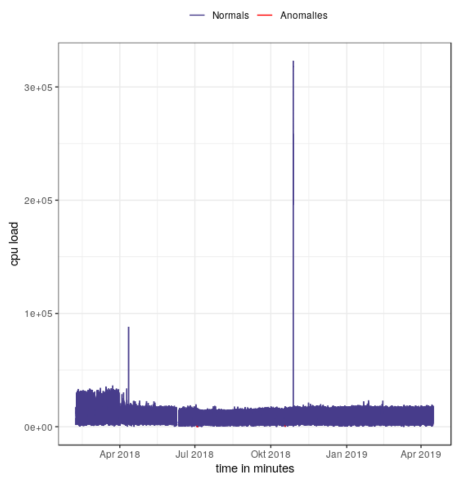
[ZWY10] Weiwei Zhang, Jianhua Wu, and Jie Yu. An improved method of outlier detection based on frequent pattern. In *2010 WASE International Conference on Information Engineering*, volume 2, pages 3–6. IEEE, 2010.

A DEPICTION OF THE ANNOTATED ANOMALIES FOR EACH UNIVARIATE TIMESERIES

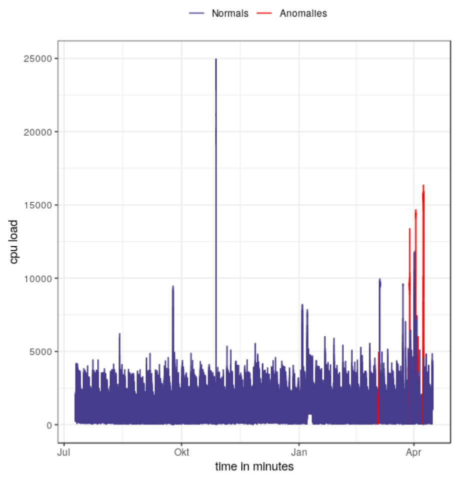
Here, we present a detailed view of the labeled time series. The three univarate time series io_busy, user_busy and Rollbacks for both hosts are depicted in Figure A.1. The normals are drawn blue and the anomalies in red. As can be seen in Figure A.1a and Figure A.1b, the order of magnitude differs for io_busy, user_busy, and Rollbacks of Host I and Host II. Host I has a mean of 1147.53 in io_busy, while Host II io_busy is with a mean of 6852.90 higher in average.
Analyzed panel by panel, in Figure A.1b and Figure A.1f is no direct visible connection between the curve progression and the few annotated anomalies. In Figure A.1a and Figure A.1c, such a correlation might be guessed in combination of the visible peak in April and the more frequent occurring anomalies.



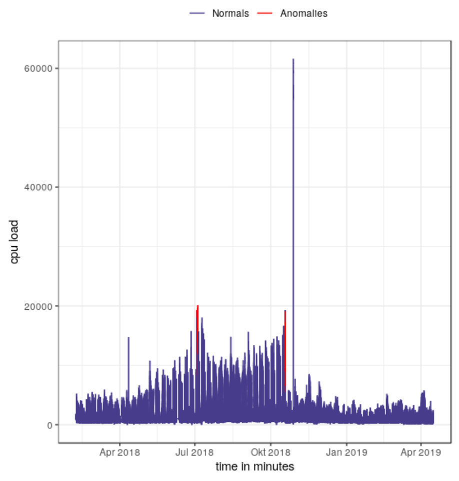
(a) Host I io_busy



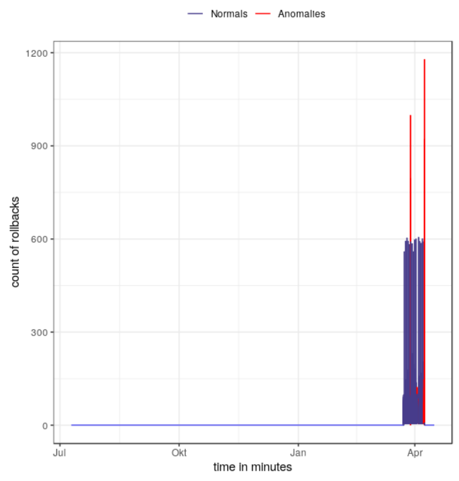
(b) Host II io_busy



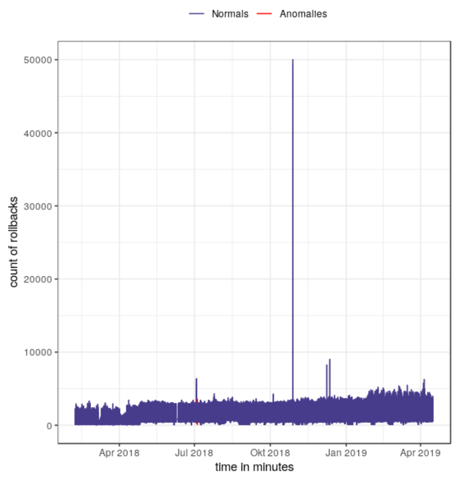
(c) Host I user_busy



(d) Host II user_busy



(e) Host I Rollbacks



(f) Host II Rollbacks

Figure A.1: Depiction of the Annotated Anomalies (Red) in each Univariate Time Series

B DIMENSION REDUCTION FOR VISUAL ANOMALY ESTIMATION

As the data in use is four-dimensional, a depiction is not possible. However, in four-dimensional space clusters can appear and dissolve which is in a two dimensional depiction as in Figure 4.1 not obvious.
Reducing the dimensions is a common way to keep the essential part of information, but to break the data down to two dimensions. We employed two algorithms: Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE), to receive the two main components of the data for depiction. For PCA the components have a explanatory value of 72.8% of the data set, the resulting image is shown in Figure B.1. For t-SNE, with a perplexity of 30 is shown in Figure B.2. However, as PCA transfers data based on linear combinations and t-SNE based on nonlinear combinations, the algorithms do not keep deviating information, hence they themselves can be used as an anomaly detection method. That the red anomalies in Figure B.1 lie within the normal data points is caused by the fact, that the deviating information has not been retained. Hence the arrangement of the data points in their four dimensions is difficult to imagine.

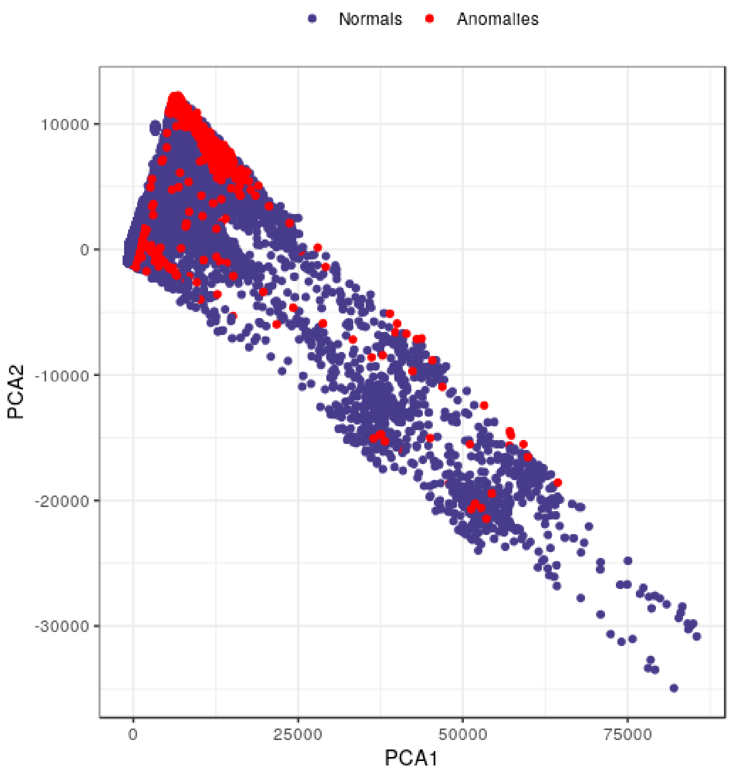


Figure B.1: Depicition of the Multivariate Data using PCA Dimension Reduction on the Multivariate Data of Host I

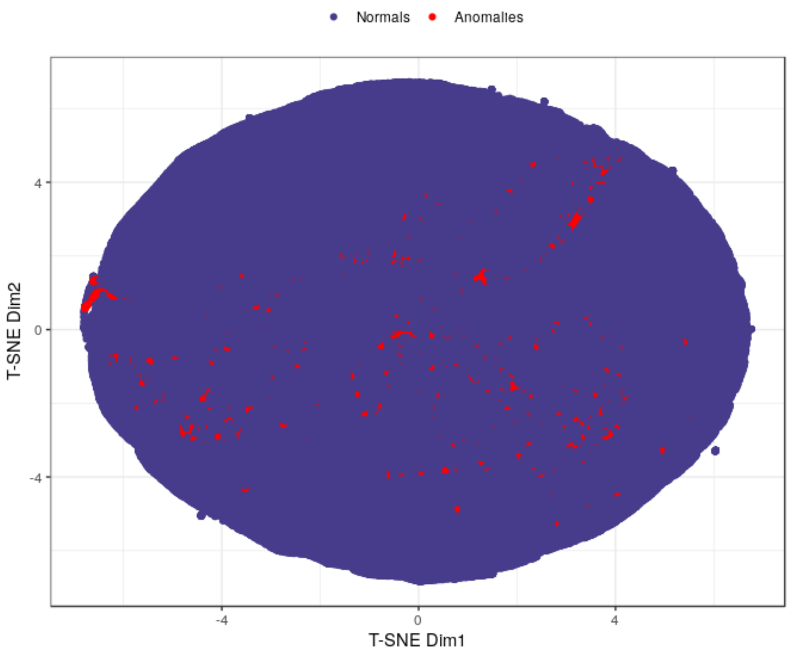


Figure B.2: Depicition of the Multivariate Data using T-SNE Dimension Reduction on the Multivariate Data of Host I

C LIKELIHOOD OF A LOCAL OPTIMUM FOR THE *K*TH NEAREST NEIGHBOR METHOD

Of the examined methods, the *k*th Nearest Neighbor Method has shown to be the method with the highest ROC AUC. However, the found parameter of 20 is most likely not at a global maximum. As shown in Figure C.1, many other neighbors achieve similar ROC AUC scores. In total, a single prediction includes three weeks, which are 30.240 data points and possible neighbors. Since an extensive search through these neighbors would be very time and resources consuming, we have limited ourselves here to the next thirty neighbors. It is very well possible that a k with higher ROC AUC exists in the remaining 30.210 unexamined parameters. The multivariate method found compared to the univariate method with $k = 2$ a more prominent maximum. Still, we cannot exclude that higher maxima in $k > 30$ exist.

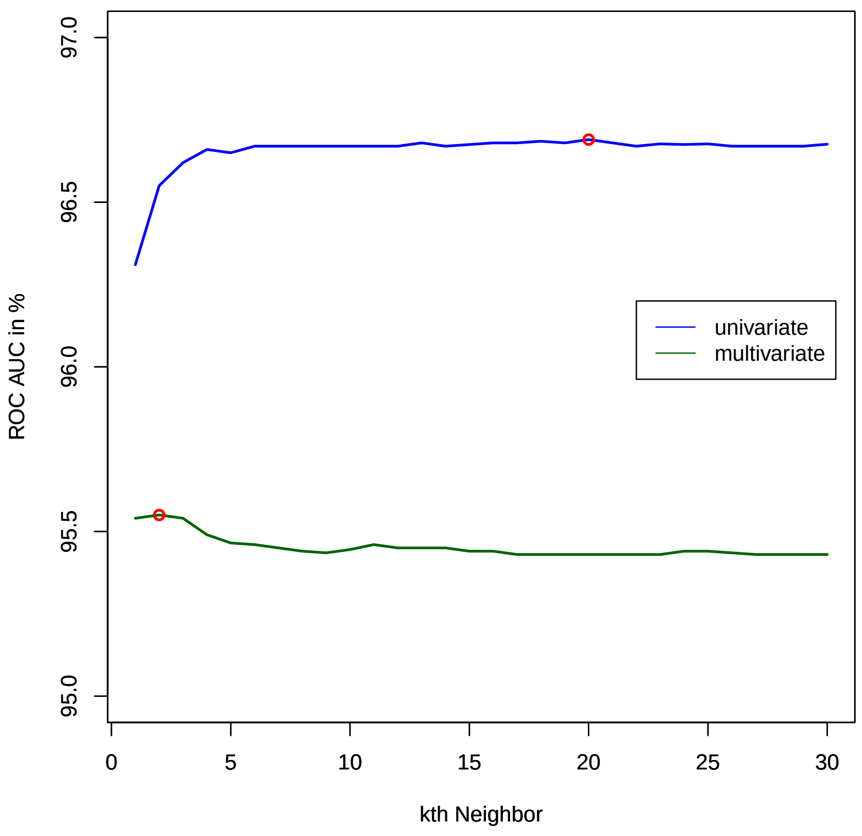


Figure C.1: ROC AUC of the *k*th Nearest Neighbor Distance Method for Different *k*

LIST OF FIGURES

LIST OF TABLES