Ensemble Based Unsupervised Anomaly Detection

ERIK ALSTERSJÖ
ALEXANDER P. THEOFANOUS

MASTER´S THESIS

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Ensemble Based Unsupervised Anomaly Detection

Erik Alstersjö
adi10eal@student.lu.se
ealstersjo@gmail.com

Alexander P. Theofanous
zba08ath@student.lu.se
theofanous.alexander@gmail.com

Department of Electrical and Information Technology
Lund University

Supervisors:
Ph.D. Zhi Zhang
zhi.zhang@eit.lth.se

Ph.D. Jianhua Cao
jianhua.cao@clxcommunications.com

Examiner:
Prof. Maria Kihl
maria.kihl@eit.lth.se

March 20, 2017

"It doesn't matter if you pass the semester by getting 50% or 95%. Passing is passing"

- Engineer

"Research is what I'm doing when I don't know what I'm doing."

- Wernher von Braun

"The real problem is not whether machines think but whether men do."

- B. F. Skinner

"Computers are useless. They can only give you answers. "

- Pablo Picasso

"You tried your best and you failed miserably. The lesson is: Never try"

- Homer Simpson

"To err is human - and to blame it on a computer is even more so."

- Robert Orben

"In computing, turning the obvious into the useful is a living definition of the word 'frustration'."

- Alan Perlis

"If everyone is thinking alike, then somebody isn't thinking."

- George S. Patton

"Democracy is an abuse of statistics."

- Jorge Luis Borges

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

A methodology as well as a suggested solution to the problem of unsupervised anomaly detection for contextual anomalies is presented. Using a combination of statistical and clustering approaches, an ensemble of algorithms provide automatic anomaly detection in an Application-to-person networking environment which can be scaled to different domains using hierarchical time series data.

The aim of this thesis is to further advance the field of anomaly detection and to provide conclusions with regards to the usability, maintainability and trustworthiness of unsupervised anomaly detection frameworks. Applications in the domain of unsupervised anomaly detection are hard to evaluate, thus methods as well as future work, which can be used to further create unmitigated assertions about any data set, is investigated.

An introduction to the concepts underlying anomaly detection as well as an implementation of the concepts are presented. Principles of machine learning are applied using static thresholds and assumptions about the data set being monitored. No active learning or dynamic adjustments of the anomaly detection framework is applied with the drawback of limiting the resulting classification but still providing clear and robust insights into the analyzed data.

It is shown that purely statistical or naive probabilistic assumptions about any data monitored is inconclusive in producing a fair estimation of anomalies. For a setting where the utility of an anomaly detection framework are not adamant to the survival of a monitoring system, the proposed solution works adequately. Since the results have not been validated, no conclusions can be drawn with regards to recall and precision metrics.

Popular Science Summary

Anomaly detection is a wide concept with many applications. In recent years a lot of work in this field has been done. We have devised a neat and novel way of finding anomalies. The work needs some polishing to be precise, but on the upside it can be used on more or less any kind of structured data where the goal is to find the answer to the following question: What are anomalies in a particular context and how are they found?

Our work has been focused on devising a monitoring system which scans data and says "Hey! Something is up!" whenever an anomaly occurs. The data which is scanned could be anything: the amount of telephone calls with your parents during a week or how loud those conversations are. There is no limit to what you can monitor in the digital sphere - as long as the data is structured.

So lets say that you have access to structured data but you have no idea what constitutes as an anomaly. Furthermore, your task is not only to find the anomalies in conversations with *your* parents, but for *all* conversations for every single person in Europe which has a cell phone. That's a lot of data which you have absolutely no time or interest in checking manually. You also know that you can't just use your own conversations as a reference for every other person's conversation. Why?

Some people talk with their parents several times a week and sometimes they yell at each other for long periods of time. Some people change how they talk with their parents depending on what time of year it is. Perhaps there is a holiday or a birthday coming up? In order for your anomaly detection scheme to be relevant for more people than yourself, and for some extended period of time in the future, you have to create the system so that it can work in different concepts. Is time an aspect? Is cost an aspect? There are many contexts to adhere to! How can you devise a simple anomaly detection system which accounts for these, any many other unnamed, factors? And why would you?

Margaret Atwood, the Canadian lyricist, once wrote *"We still think of a powerful man as a born leader and a powerful woman as an anomaly"*. Social structures and perceived notions of what does, or does not, *seem* regular changes over time. The same can be said about telephone calls or data. Specifically cloud data, or big data. By applying anomaly detection on big data you can help transform the notion of normality in almost any context or domain. This in turn could lead to technology or services that are more suited for your (or perhaps your parents) needs.

The question of "how" is solved by statistics coming to the rescue! In our work we have used some basic statistical ideas working independently. Each on their own are rudimentary and simple - like a lonely violinist playing to a big crowd. But when joined together, the statistical concepts create an ensemble. It is the ensemble's work, or music produced, which gives you a more complete picture of what constitutes as an anomaly. In other words: the whole is greater than the sum of its parts.

Foreword and Acknowledgments

Back in the beginning of 2016, our goal was to find a master thesis project that we could collaborate on despite our different educational backgrounds. We were welcomed by company representatives from CLX Communications during a work fair, *Teknikfokus*, that was conducted by students at LTH. The company representatives directed us to an enthusiastic employee, Jianhua Cao, that made sure we could conduct the thesis project at CLX Communications between June and October in 2016 with him as our supervisor.

Before starting the master thesis project we had not come across the concepts of anomaly detection and machine learning. This meant that a lot of time was spent on learning the basics of the concepts. The lessons we learned and the result of us applying those lessons are demonstrated and presented in this report. We could not have finished the project without the help from some amazing people.

We would like to thank Jianhua Cao for his extremely good supervision and for his interest in our progress. Jianhua provided us with many good ideas and guidelines.

We would also like to give a big thanks to our supervisor at LTH, postdoctoral fellow Zhi Zhang at the department of Electrical and Information Technology. Zhi helped us keep the right path during the completion of our thesis and was very helpful in making us understand what we were doing. Zhi brought a lot of encouragement and helped us to write and publish this report as well as articles in the academic papers.

We would also like to thank the employees at CLX Communications in Lund that have showed interest in our work and helped us when we needed it. A special thanks to Jonas Lindeborg that kept an interest in our progress and brought much moral support. We would also like to thank Maria Kihl, professor at EIT, for being a fair examiner as well as providing a lot of support and assistance in the beginning phase of our master thesis project.

Apart from these fantastic people we would like to thank our friends and family that have listened to our talk about anomaly detection during all these months. You have a great part in our success.

Table of Contents

List of Figures

List of Tables

Technical Terms and abbreviations

A2P - Application to person

AD - Anomaly Detection

Algorithm - A single module within a system that executes a specific task.

Consensus score - This term describes a situation when all scoring algorithms agree, with no exception, that a value is either definitely an anomaly or not. A consensus score does not guarantee correctness, it just strongly implies that the resulting score is true. A consensus score can be seen as an unanimous vote, with no or negligible deviations, with regards to an entry's classification.

Ensemble - A collection of algorithms that operate within the same context.

Epoch time - A universally agreed upon time metric which maps any given time to the amount of seconds that has passed since the 1st January 1970 at 00:00:00 in Greenwich Mean Time and vice versa. An epoch time stamp of 1234567890 can thus be converted to the "human readable" time of 13th February 2009 at 23:31:30. The epoch time stamp of the date December 31st 1999 at 23:59:59 would thus be converted to 946684799.

MAD - Median absolute deviation.

Matching historical records - This term is a label for a subset of historical records that have been extracted from the entire database containing all available historical records. The matching can be done in several steps or layers such as that all points in the matching historical records match an inputted entry ϵ with regards to a set of keys and a particular time or date.

Scheme - An abstraction, that can be seen as a flowchart, describing how an algorithm or ensemble goes from accepting input values to producing meaningful and final output or results; a white-box representation of the anomaly detection system.

SPN - Service provider network identifier which indicates who owns the digital infrastructure facilitating communication between devices or users.

System - A complete framework for a service; The "black box", product, prototype or application that facilitates a certain function.

Time series - A collection of values or points that are ordered based on some ordinal metric. The ordering is context dependent and the points can take on a multitude of dimensions. *Hierarchical* time series have several ordinal metrics which can sort a set of points in differing ways.

XKC - X-key-combination, where X is a positive integer that represents the amount of keys that can be used to access or identify entries in a data set. Synonymous with *level.*

Chapter 1

Introduction

1.1 Background

The focus of this report is to present a solution to the problem of detecting irregularities in data, streaming over digital networks. The importance of finding this solution is increasing since digital communication has escalated over the last few years [1], [2] and is expected to rise in the future [3]. Users - e.g. individuals, organizations or sensors and actuators - of electronic devices are sending and receiving data over digital networks. The transactions of data within these networks are becoming faster, resulting from higher bandwidth as a consequence of the demand from the increasing user bases [4].

The devices can range from Personal Computers and medical equipment to embedded systems such as motion detectors. The digital communication can be performed through services that are directly associated with, or connected to, those devices. The utility of the services can range from sending and receiving telephone calls and Short Message Services to wireless internet access or Global Positioning Systems, more commonly abbreviated as SMS, Wi-Fi and GPS respectively.

The interconnection between the devices and the services that they provide can be monitored and configured through subsidiary applications. The applications can range from web browsers to simple scripts running in a background process of an operating system in any given Personal Computer [5].

The connectivity between devices, services and applications can be regarded as a digital communication platform. We define the term digital communication platform as an abstraction which asserts the following: digital communication, the sending and receiving of messages, can only be achieved by using at least one element from each of the sets of devices, services and applications respectively.

Without a digital communication platform, no digital communication between two parties or entities can be achieved. The purposes of having digital communication platforms can vary immensely. Individuals might use them in their private lives to connect with friends and families over social media while companies might use them to monitor their competitors' stock [6], [7].

The aspect of monitoring traffic or data can be important for various actors for different reasons. Identifying and reacting to the change in income figures over a specific time period or the amount of bytes sent over a Local Area Network is important for the safety and longevity of a company, an organization or system [6], [5].

A monitoring system can be deployed to survey and inspect data, whatever the data may represent, in order to find patterns or values that are anomalous [8]. Anomalies indicate that something unusual or unexpected has been observed in the data [6]. The detection of anomalies is a widely researched subject that has relevance in many fields [5].

Values, instances, behaviors or patterns that can be extracted from one or several elements of a data set are defined as anomalous when they do not conform to a defined notion of normality [9]. Digital, automatic or complex anomaly detection systems are employed when the data being monitored necessitates it. Anomaly detection systems are important because they provides insights into the data which might be too hard, infeasible or simply practically impossible for a human to provide.

Most importantly, anomaly detection systems gives the facilitator of the system a chance to react to the anomaly before it leads to unavoidable consequences. In some instances the consequences are trivial, in others disastrous. This raises the bar for an anomaly detection system to provide correct prediction, something that will be elaborated upon in section 1.1.1.

In its core, anomaly detection systems warn the user of the system that something strange is going on - nothing more and nothing less. A good anomaly detection system will alert the user about the strange occurrences as soon as possible, mitigating the cost or effort needed to solve the potential problems that the anomalies can be a consequence of [6]. But an anomaly detection system that is too sensitive to changes or irregularities in the data it monitors would negate the trustworthiness and usefulness of the system[1].

This ambiguity of deciding when, where and how a strange value becomes an actual anomaly is where the true challenge in designing anomaly detection systems lies. How does one design an anomaly detection system that spots all "real" anomalies but at the same time does not consider minor, small or expected irregularities as anomalous? How does one quantify or classify whether the transfer of data between two parties has been made properly, without anomalies?

A certain value in the data, or a certain behavior of a system, could in some cases be regarded as anomalous - and in others it might not. What constitutes as an anomaly is highly context dependent and in some cases ambiguous[10], [11]. If

[1] In the same way that the boy who cried wolf too many times led to the sheep being eaten.

the ambiguity is left unresolved it could lead to anomaly detection schemes that generate erroneous classifications. This raises some questions and highlights the difficulties in designing anomaly detection software.

How does one decide whether it is the anomaly detection scheme that is behaving incorrectly or if it is the data that is insufficient or misleading? What are the criteria for detecting anomalies? These questions have occupied researchers in many fields, most notably within the Machine Learning community [12], [13], and, depending on the context, the questions have either been answered or still remain a mystery [14].

The act of anomaly detection can be performed by manual inspection, such as an individual or an “expert” looking over a data set and marking instances as anomalous or not. When working with big or streaming data sets, such as big data or cloud data, manual inspection becomes unfeasible as the amount of data increases [15], [16]. A more feasible solution is to perform anomaly detection using software.

The output of an anomaly detection software application is a "label", a "classification" or a "score" - terms that are elaborated upon in section 2.1.4 - which informs the user whether the values, entries or points of the input data is anomalous or not. Anomaly detection software is governed by specified metrics, the actual input to the algorithm, and the algorithms themselves which dictate how the classification or labeling of anomalies is supposed to be performed.

In the rest of this report it is assumed that all AD schemes, algorithms and systems are to be viewed as operating as part of a software application. Furthermore, the abbreviation "AD" will be used instead of the term "anomaly detection".

1.1.1 Setup of AD schemes

The amount of data that a system is to process at a given time and the amount of historical data that is accessible dictate the starting point of designing an AD scheme. The characteristics of the data as well as the mechanism generating the data play a big part in establishing boundaries for the AD scheme.

In some instances it is known that the data comes in batches at specific time intervals while in other instances the input data is inputted randomly. Each entry or point in the data can have a multitude of values of different types[2].

Before a definition of an anomaly can be reached, i.e. what to look for in the data, one has to decide which values or metrics that are relevant, i.e. where or what part of the data, to investigate [6], [17]. The metrics or values in a given data set can have different meaning depending on what they represent and in some cases a combination, such as the ratio or difference, between two or more given metrics is in itself of interest while in other cases it could add unnecessary complexity to the AD scheme. As an example, if the purpose of an AD scheme is to

[2] Integers, strings, floating point values etc.

alert when a user of a social network is sending an anomalous amount of messages to other users - perhaps because the user has been hacked - then the metric of interest is the amount of messages transmitted by a user in a specific time interval.

What does a “message” represent in this context? Is the amount of messages sent the only interesting aspect or is the size of the message(s), the amount of words or bytes, also important? Is the size available as an explicit metric in the data or can it be derived? Are the metrics accessible immediately or is there a time delay before the system is aware of the values?

As mentioned earlier the notion of an anomaly is context specific [6] as will be elaborated upon in section 2.1.2. The point is that AD schemes are not fully autonomous or agnostic with regards to to the structure and domain of the data which it operates upon [18].

Hierarchical time series

The examples illustrated in this report, as well as many applications and schemes in the research community can all be seen as operating on data that is in the form of time series [5], [14]. Time series can be hierarchically organized and thus be aggregated at several different levels or in groups based on some set of features [19]. In other words, the values or points in the time series can be divided or grouped in other time series based on some metric, value, identifier or other attribute which is present in the original data.

Hierarchical time series introduce the concept of levels, meaning that a certain point or value in a data set can be seen from different "perspectives", i.e. categorized as belonging to several different levels at once.

Furthermore, a value or an entry in a time series data set, can be investigated based on several conditions: The hierarchies present in the data introduce different conditions for anomaly classifications. At one level a certain value of the data is completely normal or as expected, but at a different level the same value can be an extreme outlier. But it could also be anomalous on all levels or at none of the levels.

The values in a given data set represent something and that something dictates what levels and values that are of importance when looking for anomalies. An intelligent, perhaps artificial, entity which understands the data dictates what areas are of importance. For a more thorough elaboration about hierarchical time series and how they are utilized in our work see section 2.1.3.

AD utilization

AD schemes finds anomalies where you, the user, tell it to look using the tools that you give to it. AD schemes find anomalies - they do not prevent or directly explain what caused them [5], [20]. Furthermore it can be noted that an anomaly is not necessarily always equivalent with a catastrophe [6]. An AD scheme does not, in principle, produce an estimation of goodness; it just highlights things that are not ordinary [8]. AD schemes can however rank, or score, the range or grade of abnormality for an anomalous instance [21]. This score can be interpreted, implicitly and with caution, as a rank of goodness [6]. Below is presented a simplified and fictional example which illustrates how, what, when and why AD can be achieved.

> *A hospital in Lalaland has ten wards where each ward deals with a specific type of injury or condition (burn ward, pediatric ward, E.R, etc.). The hospital admits on average a total of 100 patients each day, every day. The hospital employs a novel AD scheme whose job is to count the number of patients admitted to each ward on a daily basis. The AD scheme monitors one metric for each ward and as such the entire hospitals total admittance. If the metric, the number of patients admitted for either the hospital or each respective ward, deviates with a factor of x from the average value, then it is regarded as an anomalous instance. Note that a total number of 100 admitted patients is the baseline for this hospital.*
>
> *Yesterday 100 patients were admitted to the hospital - just as expected. However the orthodontic ward, which usually averages 3 patients admitted per day, admitted 30. As we can see, on the level of the entire hospital's admittance nothing abnormal happened yesterday. However, the orthodontic ward received 10 times more patients than usual. If* $x = 33\%$ *any admittance number below 2 or above 4 patients would be regarded as anomalous. This further means that for all the other 9 wards at the hospital the expected total average admittance would be* $(100 - 3)/9 \approx 11$ *but yesterday it was, on average,* $(100 - 30)/9 \approx 8$. *This means that all wards except the orthodontic ward admitted on average 3 patients less than usual.*
>
> *From the information given there is no way of knowing if each single ward each had 3 less admitted than usual or if perhaps one single ward had 30 less than usual. A drop of 3 patients per ward might not be anomalous for most wards; it depends on what the threshold factor x is and what the average admittance for each ward is. One could argue that it is quite reasonable to not expect that each ward always, every day without exception or deviations, admits a constant amount of patients.*
>
> *If all the other 9 wards have a small, or non-anomalous, decrease then the whole system, the hospital, only has a single anomalous instance - the increase at the orthodontic ward. But the case could be that there was a drop of 30 patients for a single ward. That would quite reasonably be regarded as anomalous for that ward. If this is the case then we have two instances of anomalies (The orthodontic ward with +30 patients and ward Y with -30 patientes).*
>
> *The question is then: are there any more, and if so how many, anomalies in the data? In order to find out we have to iterate through the*

admittance records for each individual ward and find out where further deviations exist. We do not need to go through all wards individually if we are only interested in the patients admitted for the hospital in total. But it is not up to the anomaly detection scheme to make that decision. It is up to the intelligent entity in charge of monitoring the admittance records to decide how thorough or deep the investigation should be.

Furthermore, what caused the anomaly? Did two wards accidentally swap their records and this is just a miss in the paperwork? Did someone switch the signs, pointing to the wards outside of the hospital, confusing patients? Were patients admitted because some wards were closed down due to a reparations of instruments, obfuscating the logging system? Did the personnel in the orthodontic ward act carelessly or miss something during their screening process of the patients? These are all relevant questions but they cannot be answered by the AD scheme unless the system, in which the AD scheme is operating within, contains that functionality. However, if these problems were quantifiable and measurable then they could be added as metrics to the admittance logs. Then they in turn could be used by the anomaly detection scheme in trying to determine "how bad" the increase or decrease in admittance is, i.e., we would insert exceptions to the rule of "x% deviations are anomalies".

If the increase for the orthodontic ward was exactly the same magnitude as the decrease for, say, the burn ward and the anomaly detection scheme was somehow informed that "the orthodontic ward will receive all patients that would be admitted for the burn ward" (for whatever reason) then this would not have been an anomaly at all and whoever, or whatever, is monitoring the system would never have been alerted. The anomaly detection scheme in this example could be extended to have dynamic values for x or allow updates of x based on a set of circumstances. It is up to the owner of the monitoring system to decide what level of complexity should be employed in investigating anomalies in the data at hand [6]. The number of exceptions or special cases, if maintained manually, could become hard, to expensive and time consuming [17]. Is it worth it?

1.2 Problem formulation and motivation

Networks are becoming crowded as more data is being sent over them [3]. The increased speed, bandwidth utilization and user bases operating over digital communication platforms, as mentioned in section 1.1, is not without its problems [15], [22]. The increase can potentially lead to anomalies with regards to how, when or if the data that is being sent from one end is actually received, in a timely and properly fashion, at the other end [8]. Technological advancements with regards to anomaly detection has been a necessity in the modern era of increased digital communication [8] as well as in other fields [25]. As the flow of data increases more pressure is put on researchers to find faster and more efficient solutions to the problem of AD [12], [13] where both speed and precision are key factors [23], [24].

AD schemes that mark instances as anomalous when they are not, or anomalous instances being marked as safe, can make AD schemes unsafe, untrustworthy or redundant. In certain applications, such as in medical diagnosis, misclassifications could have catastrophic or irreparable consequences [6], [26].

Making AD automatic enables people or employees to do other beneficial tasks other than analyzing data for irregularities [17]. Automatic AD schemes, implemented in software applications, can detect anomalies in big data sets reducing the factor of human error as well as enabling faster detection[17]. This motivates the purpose of utilizing a refined anomaly detection framework that can operate autonomously with as few as possible erroneous classifications [15].

This work will focus on developing an AD scheme operating in an Application-to-person (A2P) environment. The AD scheme should detect and rank anomalies, with as few false classifications as possible, in a data set provided by the company CLX [28] where the case study was performed.

The company and the case study are described in section 1.3. The case data, described in section 2.1.3, is "unlabeled" - a term explained in 2.1.4 - and contains values with regards to some known metrics. The problem we try to solve is by no means isolated to the domain of our case study and is further deemed as important to solve in many other areas[17], [15].

1.2.1 Restrictions

Our AD scheme will operate in a dynamic, automatic and unsupervised fashion. To legitimize our solution our design choices will be based on machine learning techniques as well as established statistical measures. By applying aspects from machine learning and statistics we enforce that our solution is based on more than solely intuition.

For novelty purposes we will introduce custom modifications based on our understanding of the data set. Our algorithms will still be general enough to be re-purposed in different domain, however all parameters are tuned to evaluate our solution on the case study data set. This evaluation can therefore not guarantee that our scheme is completely compatible with other platforms or domains with our parameter setup. The parameters governing our scheme's execution are described in section 3.5.

However, we do aim to make the structure of our AD scheme as general as possible so that it can work, with minor modifications and parameter tuning, in different settings or domains. The principles of our AD scheme are not tied directly to the structure of the case data. The scheme, as well as our methodology, as presented in section 3.1, is however adapted to the given case study.

1.2.2 Purpose and aim

The purpose of this work is to implement and evaluate an AD scheme that supports unsupervised AD, a term described in section 2.1.1. The scheme will be part of a novel proof-of-concept framework. The framework should support the testing, simulation and visualization of the implemented AD techniques. These functions are incorporated into the framework in order to gather results regarding any arbitrary AD technique's effectiveness.

As for future applications, our AD framework could act as a test-bench for evaluating and improving either our own or other existing AD techniques. The software implementation is general and can be used in many systems. However all parameters are optimized for performance on CLX's case data, as described in section 1.3.1.

Our findings and results can be used for further research in the field of AD to guide researchers as to which techniques can be used when implementing AD schemes. Our goal is to show that combining probabilistic models of time series data is sufficient in producing AD software by highlighting its benefits and limitations for future work.

General inquiry

Can a custom made, unsupervised, automatic AD scheme be implemented by combining novel anomaly detection algorithms in an ensemble manner? Furthermore can this system generate anomaly scores whose reliability surpasses that of a static AD system, while operating in a real-time environment?

1.3 CLX Communications

The case study is done in collaboration between the company CLX Communications, hereafter shortened as CLX, and the Department of Electrical and Information Technology at Lund University[27]. According to their own authority, CLX offers solutions for delivering voice, messaging and data services as well as customer experience management. They also offers cloud-based communication and mobile data services and solutions [28].

AD could play a big role in minimizing costs and maximizing customer relations for a company such as CLX. By detecting anomalies in the systems and frameworks maintained by CLX, the company can make sure to resend, repair or reroute messages delivered through their services. Today CLX's AD system uses a rule based approach which requires threshold tuning. A non-parametric approach that provides control of trade-offs between true and false anomalies will greatly improve the company's operational efficiency.

The system which our case study revolves around, whose corresponding data set is described in section 2.1.3, will be summarized below. Please note that for legal reasons we will not account for all the details of the system. Instead we will give a generalized description which fits our case study, but is general enough to be used as reference for readers whose interests lies in other domains than that of CLX's.

1.3.1 System overview

This section provides a cursory introduction to the case study and is complementary to section 2.1.3 which provides a more technical description.

A service provider facilitates digital communication between users. A service provider makes sure that its customers can send and receive messages, to and from other users. The other users could be customers of the same or any other service provider, but only as long as both users are customers of service providers within the same country.

Service providers operate in many different countries all over the world. Each country has several, or at least one, service provider operating within the country. Each service provider allows the communication between users using a set of applications A. Thus user X can send a message to user Y if both users are communicating through, or via, any a_i where a_i is a part of A. The service provider(s) can be seen as an intermediary that keeps track on how many messages are being sent and received through their network of applications. Each service provider network, abbreviated as "SPN", keeps track on, or counts, two things:

1. the number of messages that are **sent from** customers of the service provider,
2. the number of messages that are **received by** customers of the service provider

Each SPN keeps track of the number of messages sent and received for each respective application that the SPN provides, i.e. each SPN is responsible to keep track of the traffic of each individual application. Approximately every 15 minutes the counters are reset. Before the counters are reset the values are recorded in a log. Each entry in the log contains, apart from the time just before the reset, information regarding:

- What application was used, an application identifier that we will call "type",
- Which service provider was used,
- Which country the service provider is located in,
- The sum of each counter as described in the list above

Since each entry contains information about all the identifiers (country, SPN and type), which we will kall keys, there can exist several entries sharing the same country, SPN or application. However, all entries are unique and distinct with regards to their combinations of all keys, e.g., several entries can have the same country of origin and SPN, but then those entries, all respectively, must have a unique type.

Every 15 minutes the log is appended with several entries detailing the volume, or flow, of communication. The amount of entries depends on how many countries, SPN's and types that the entire digital infrastructure contains.The total flow increases and decreases depending on how active the users are.

The activity of the users varies depending on a multitude of factors which are sometimes relatively impossible to predict. The different types, or applications, in the system are connected to social media sites which means that major sport events or breaking news could highly influence the traffic flow - both on a local, national and even global scale [6]. In short, traffic flow has a close relationship to external, from the system's perspective, events which cannot be controlled or mitigated.

The concern for CLX is whether the flow increases or decreases too rapidly or in any other way becomes disrupted. A service provider might be overloaded with messages from one or several applications, possibly disrupting the accessibility of the information sent which in turn might render users to switch applications or service providers.

As the overseer of the communication flow, CLX has the ability to note when and where anomalies occur - assuming the logs are monitored by an AD system. Before the anomalies turn into problems, CLX has a window of opportunity to warn the service providers of the disruption and recommend the re-routing of traffic. An AD system could also be able to show if customers are leaving one service provider for another, or if a certain application is becoming used less frequently, enabling CLX to make smarter business decisions or predictions.

1.4 Outline

In chapter 1 we present the reader with a general concept of AD. The chapter initially conceptualizes and illustrates, from a general perspective, what AD is and which problems AD can or cannot solve. Challenges in designing an AD system as well as limitations and possibilities of AD systems are summarized. The chapter is concluded with a more narrow overview of the specific problem that this thesis tries to solve, omitting purely technical constraints.

Chapter 2 provides an in-depth description of the investigated system. The chapter will also provide all technical details and concepts that are relevant for our implementation such as related work and overview of the data set used in our case study.

Chapter 3 illustrates the practical aspects underlying our final solution. The methodology and functionality, for all relevant parts of our implementation, is described. A run-down of the ensemble and its preceding elements are presented.

Chapter 4 presents the resulting outcome from our implementation. The chapter serves the function of presenting all of the results that are relevant given the implementations assumptions and expectations.

We finish our report in chapter 5 where the limitations and proposed future work for our implementation is presented. The discussion highlights the restrictions that our implementation proposes on any actor who wishes to further use or improve our implementation.

Chapter 2

Theory

2.1 Technical information

As will be further elaborated upon in chapter 3, our chosen implementation is highly influenced by an array of different studies and works whose focus has varied between different domains, settings or types of anomaly detection. This section serves the purpose of emphasizing the concepts extracted from these previous works as well as putting them in a relevant context.

2.1.1 Categories of Anomaly Detection

Anomaly detection is usually divided in three categories: Supervised, semi-supervised and unsupervised anomaly detection [5]. The general difference between the categories is how algorithms, operating in its respective category, treat its reference data, also known as the training or learning data.

The reference data can be labeled meaning that each instance or point is labeled as either "Anomalous" or "not anomalous", usually represented with a distinct value such as 0 or 1. As one might expect, labeled data is in many cases hard to come by - especially if one expects the labels to be correct or trustworthy [29]. The reference data values serves as a baseline for comparison with the new data - the data which is to be assessed for anomalies [8].

Supervised

In supervised anomaly detection the anomaly detection scheme is given as input a selected set of reference values [5], [21]. The supervised anomaly detection scheme knows immediately that any incoming data that does not match the values or patterns of the reference data is anomalous. The credibility of supervised anomaly detection schemes relies on the reference data being correct. The correctness is a judgment that an expert individual or other entity makes before execution of the scheme or algorithm.

A benefit of supervised anomaly detection is that it can give reliable and legitimate results that clearly outline when, where and if the input data is anomalous since it is assumed that the boundaries, or threshold, governing the classification of anomalies is known beforehand. This makes supervised techniques relatively easy to evaluate. However, every time the input data evolves, i.e. new patterns emerge that are not supposed to be deemed as anomalous, so must the reference data be updated. Otherwise legitimate data will be deemed anomalous and as a consequence render the anomaly detection scheme useless. This in turn involves yet another verdict by an expert entity. A supervised anomaly detection scheme is not suitable for analyzing data which can have several or varying levels of "correctness" unless the iterative involvement of experts and updates of the reference data is maintainable.

Unsupervised

In unsupervised anomaly detection the reference data, if any, is unlabeled [8]. The job of an unsupervised anomaly detection scheme is not to compare new data with "correct" data. Instead, its job is to to find values or patterns, within the given input data, that does not fit in with the the values or patterns present in the rest of the input data [5], [21].

Unsupervised techniques are completely agnostic to the concept of correctness with regards to detecting anomalies. For an unsupervised technique the classification of anomalies is based only on how the intrinsic properties of the data varies. If the data deviates from the model that represents non-anomalous values too much then the point, value or pattern, which contributes to the deviation, is anomalous [6].

The benefit of unsupervised anomaly detection techniques is that it can find deviations and patterns without being given a single reference value or label to adhere to or comply against. The drawback is the agnosticism towards the function or the domain in which the anomaly detection scheme is used upon.

Unsupervised anomaly detection schemes do not have "cheat sheets" that distinctively tell what to mark as anomalous or not. Thus the results or the output might be surprising or wrong depending on the nature of the input data and the mechanism that generated it.

An unsupervised detection scheme might label values or patterns as anomalous while those same values or patterns are actually expected to arise, i.e. a human expert would probably not label them as anomalous, and vice versa. The credibility of an unsupervised anomaly detection technique lies in the intrinsic algorithm's precision and the the choice of algorithms used to identify anomalous instances [6], [14]. The inability to use only the training data to validate results is a major difficulty when designing unsupervised anomaly detection schemes [17].

Semi-supervised

Semi-supervised anomaly detection can be seen as a mix of supervised and unsupervised anomaly detection schemes in that it uses both labeled data in conjunction with unlabeled data [21]. Semi-supervised schemes use a mechanism that enables them to learn the behavior of both "correct" and anomalous data. This decreases the amount of expert interactions and updates that could hinder purely supervised techniques. Semi-supervised techniques are closely correlated to machine learning since the learning mechanism creates models of the data which is to detect anomalous points [29]. The scheme adapts, or learns, over time how or what to distinguish as important for its AD functionality.

In more simplified terms, semi-supervised AD works in the same way as supervised AD detection but instead of being given only specific values or patterns as a baseline, the semi-supervised scheme produces, gradually over time, its own labels which the scheme re-purposes to adapt its classification, or labeling, mechanism[1]. Semi-supervised techniques still require some initial labeling but it greatly decreases, or even completely automates, the amount of updates of its baseline [18], [29]. However, the credibility for semi-supervised techniques relies on the model, i.e. the learning mechanisms, being a fair and accurate description of how the data is supposed to evolve.

2.1.2 Types of anomalies

There is no generalized definition of what constitutes as an anomaly in time series data because of the volatility and ambiguity of what the data can represent [14], [21]. There are however different types of anomalies which indicate, to a user of anomaly detection systems or to a reader of anomaly detection literature, which methods are more suitable than others when deducing what anomaly detection scheme to employ for a certain application [5], [8].

A value or point in time series data can have "neighbors", a term which is ambiguous in its meaning [14]. The definition of a neighborhood, or a neighboring value, depends on the domain and the dimension that any value is represented in [11], [29]. In the context of linear time, a neighborhood could be some values right before and right after a specific value x. A neighborhood could also be similar values or values that share the same attributes as the particular value x, independent of time [30].

As an example, the point x_i belonging to X has a time stamp indicating what time of the day it shows up in the data stream. Further assume that the entries in X are sorted based on time. The neighborhood of x_i could then be x_i including the values x_{i-1} and x_{i+1}, i.e. the two values closest to x_i in the time domain. Another neighborhood to x_i could be defined as all other points in X that have the same weekday (Monday, Tuesday, etc.) that x_i has.

The notion of a neighborhood could further be abstracted based on what X actually contains and what kind of values x_i represent. Is X every single entry of the case data? Is X every single entry of the case data (see section 2.1.3) where $Key_1 = AA$? These questions enforce that heuristics lay much of the groundwork

[1] A feedback loop.

behind generating neighborhoods and detecting anomalies within or around them [29].

Point anomaly

When looking for point anomalies it is assumed that the requirement for spotting anomalies is based on comparing a single value, within a data set, to a baseline [5]. No regards are taken to whether the neighboring values of the single value, or the point, that is being examined at any time should or should not influence the classification of the currently investigated value or point [21].

Context anomaly

When looking for contextual anomalies it is assumed that the requirement for spotting anomalies is based on comparing a single value as well as its neighboring values, within a data set, to a baseline [5]. A context anomaly can be seen as an extension of a point anomaly where instead of just focusing on one particular value at a time and nothing else, one focuses on the particular value as well as the area, i.e. the neighborhood or the context, that the particular value resides in [21].

A context anomaly asserts that within a certain context a point is anomalous. Contextual anomalies are sometimes referred to as conditional anomalies [31] since the value of the point itself is not enough to deduce whether the point is anomalous - a condition regarding neighboring values has to be met as well.

Collective anomaly

When looking for collective anomalies it is assumed that the requirement for spotting anomalies is based on comparing several values, a whole neighborhood, to a baseline [5]. It is further assumed that each individual value or point, in and of its own, holds no meaning in the context of anomaly detection in a data set; It is the order or sequence of values that matter [8].

Collective anomalies can be seen as an extension of both contextual and point anomalies in that particular points, in a particular context, together define whether the points in a neighborhood are anomalous. The difference between collective and contextual anomalies are that you can assert whether a single value or point is anomalous as a contextual anomaly whereas you need a sequence of points to assert collective anomalies, i.e., one cannot classify a single point or value as an collective anomaly.

2.1.3 Case data

Our case study is based on log data, which we hereafter will call the case data, from CLX's A2P system which is given a compendious description in section 1.3.1. The case data contains 15103480 entries where the first and last entries are dated to 2014-08-26 22:11:25 and 2016-05-06 08:32:03 respectively. Each entry has six columns. A small subset of the data set is illustrated in table 2.1. The values in each column describes a particular attribute of the data. The attributes in the table are, in column order from left to right:

- # - The index of the entry in the log data set.
- **Timestamp** - the epoch time[32] of when an entry was generated.
- $Key_1 \models$ **Country** - the country of the receiver of a message
- $Key_2 \models$ **SPN** - the service provider network of the receiver of a message.
- $Key_3 \models$ **Type** - the message type of an entry.
- $Metric_1 \models$ **Number of attempts** - the number of messages sent out to that country and SPN.
- $Metric_2 \models$ **Number of confirmation** - the number of interactions from the receiver.

The keys can be seen as identifiers describing ownership of each entry while the metrics represent the values of interest. Each entry in the case data represents aggregated values for the metrics for a set of keys. The set of keys in this regard is the combination of the values for all three keys, namely $Key_{1,2,3}$. This combination will be referred to as a 3KC. For each individual time stamp and the set of entries with that time stamp, each entry will contain a unique combination of key-values with regards to the combination of all possible 3KC's. Please note that 3KC's are omitted from the case data if $\exists Key_{1,2,3}$ where $Metric_1 + Metric_2 = 0$.

From each entry a combination of keys can be extracted. In other words a 3KC can be broken down or rearranged. This illustrates the hierarchical structure of the data. In table 2.1 it can be seen that the key combination of $Key_1 = JU \wedge Key_2 = hgofg \wedge Key_3 = 1$ occurs three times, while the key combination $Key_1 = JU \wedge Key_2 = hgofg \wedge Key_3 = 2$ occurs two times. Conversely, the key combination of just $Key_1 = JU \wedge Key_2 = hgofg$ (which can be interpreted as a 2KC) occur 5 times while the key combination with only $Key_1 = JU$ (a 1KC) occurs 6 times and the key combination that only contains $Key_3 = 1$ (1KC as well) occurs 3 times.

As such the keys themselves can be aggregated or permuted to create X key combinations where the value of X is dictated by how many, not which, of the keys $Key_{1,2,3}$ that is chosen.

Which, and in what order, keys are combined indicates what level in the hierarchical structure that is being examined. We refer to figure 3.1 for a visual representation. In our example data there are $k = 3$ keys which can be arranged in 15 permutations according to equation 2.1.

$$Levels(k) = \sum_{i=1}^{k} \frac{k!}{(k-i)!} \tag{2.1}$$

Table 2.1: Example of case data

#	Time stamp	Country	SPN	Type	Attempts	Confirmations
1	1446152509	JU	hgofg	1	6	3
2	1446152509	JU	hgofg	2	5	3
3	1446152509	JU	aabaa	2	5	3
4	1446153133	GZ	hgccf	7	9	1
5	1446153409	JU	hgofg	1	6	3
6	1446153409	JU	hgofg	2	7	5
7	1446154309	JU	hgofg	1	8	4
8	1454611002	AB	gkhmk	3	5	3

Another way of interpreting key combinations that omit certain keys is that of a key combination which uses all keys, but the values of one or two keys can be anything. Thus one can interpret the example of "the key combination with only $Key_1 = JU$", mentioned above, as the following: $Key_1 = JU \wedge Key_2 = * \wedge Key_3 = *$, where $*$ stands for "any value".

We note that by organizing the data in these different structures, or views, one can see that we have exploited the hierarchical structure of the data to overcome the issue behind Blum's and Mitchell's assumptions regarding "*any two examples belonging to the same connected component [...] must have the same classification*" [33]. By treating each view, or node in the tree structure, as a single-view component we produce a higher overhead, with regards to memory usage, but with the benefit of being able to independently label all 3KC's for each available metric.

Note that the time stamp is neither seen as an explicit key or a metric value in our illustration of the hierarchical structure. This does by no means imply that it is redundant since it can, and will be used in our implementation, as a basis for looking up or extracting data, e.g. neighborhoods as noted in section 2.1.2, from specific time periods. It will also be shown that our solution disregards most of the combinations/permutations in finding anomalies since they would be redundant as per the description of context sensitive anomalies in section 2.1.2. The context in our case data is illustrated, elaborated upon and defined in section 3.2.

2.1.4 Labels and scores

A label, as per our description of anomaly detection schemes in section 1.1.1, can be regarded as the likelihood that a certain point, entry or value x in a given data set is anomalous. We will use the notation $S(x)$ in the text below to represent the value of the label for the point x. It is further assumed that x, in the context of this description, contains only one metric. In our case data each metric is regarded as being univariate and thus, the principles presented in this section can be applied on several metrics in parallel at the same time.

In some settings of supervised anomaly detection (see section 2.1.1) the labels can be binary [5], where the value $S(x) = 0$ is to be interpreted as "x is not anomalous" whereas the value $S(x) = 1$ is to be interpreted as "x is anomalous". However, in many instances the labels are decimal values such that $0 \leq S(x) \leq 1$.

The interpretation of a labeled value in the range $[0, 1]$ is susceptible to heuristics but the most common interpretation is of the following form: Lower values ($S(x) \rightarrow 0$) indicate "a low probability of x being an anomaly" whereas higher values ($S(x) \rightarrow 1$) indicate "a high probability of x being an anomaly" [5], [14], [8].

Please note that in our description the probability is only implied or indicated. It is not necessarily the case that the value of $S(x)$ is equal to the actual probability that x is anomalous - it is simply modeled as such. In a purely statistical approach, where the label is calculated based purely on probability mass or density functions, the notion of a "probability of anomaly" makes more sense. However, many applications modify or extend - using additional measures, approaches, novelties or heuristics - their calculations for more precision or more reliable results that cannot be achieved by purely probabilistic approaches [21], [20], [34], [35], [36], [37], [6]. We presume that is why the label rarely, if ever, is termed as a probability even though it can implicitly be regarded as such. Instead, the term "anomaly score" or just "score" is used [30], [14], [5].

At this point we would like to inform the reader that the research literature sometimes makes a distinction between "labels" and "scores". The term label is generally regarded as a label, such as we describe the term above, which have been generated prior to the execution of the AD scheme. The term score is generally regarded as a label, as described above, which have been generated by the anomaly detection scheme, mid- or post execution of the AD scheme. Thus it is implied that the mechanism generating labels might not be the same as the one generating scores. In other words, a label is assumed to be a correct, generally binary, classification whereas a score indicates a degree of sureness for a classification that may or may not be correct.

Classification

To reiterate: An anomaly detection scheme outputs a score for each value it has been assigned to investigate. It answers the question "Is the point, value or entry x an anomaly?"[5]. In order to produce meaningful results, for a user or monitor of the AD scheme, the score value $S(x)$ needs to answer the question unambiguously. In other words, it has to be determined whether the score value is telling the user whether it has found an anomaly or not [2].

The score $S(x)$ is, in the non-binary case, matched against a threshold value $t, 0 \leq t \leq 1$. If $S(x) \geq t$ then we say that the point, value or entry is anomalous. In this case the anomaly detection scheme produced a positive result; positive in

[2] Note that there can exist more classifications such as warnings or levels of anomalies, but here we will only illustrate the case of a binary classification

the sense that it is telling the user "yes, this is an anomaly". In the opposite case, where $S(x) < t$, a negative result is produced; negative in the sense that the scheme is telling the user "no, this is not an anomaly". Thus it can be seen that high scores imply positive results whereas low scores imply negative results [38].

Scoring and classification is two distinct steps of an anomaly detection application. The calculations and intrinsic behaviors of the anomaly detection scheme produces the score. Afterwards, the score is "boiled down" to a classification. One could very well ignore the step of classification and just output the score value itself. The need for a classification step is up to the owner of the system or application [6].

It is however, in order to more easily measure and evaluate the results of an anomaly detection scheme, convenient to have few distinct labels since it would mean that there are fewer metrics or processing steps in order to evaluate whether the scheme exhibits desirable performance. This becomes more relevant if the evaluation is done manually by a human[5].

False and True prediction

As implied in section 1.2, mentioned in section 2.1.1, and further discussed in chapter 5 the success of an anomaly detection scheme can be measured in how correct it is in its labeling. A positive result which should have been labeled as negative is called a false positive (FP), a positive result that indeed should have been labeled as positive is called a true positive (TP), a negative result which should have been labeled as positive is called a false negative (FN) and lastly, a negative result that indeed should have been labeled as negative is called a true negative (TN) [38].

Please note that the reasoning or the criteria behind the judgment of what a labeling should or should not be is derived from a set of rules, thresholds or other reference data which is assumed to be correctly estimated or based on an expert entity's opinion [5]. In table 2.2 we illustrate how to interpret these terms. The tables first row, second and third columns respectively, indicate what the output from the anomaly detection scheme can be in any general case. The first column, second and third row respectively, indicate what the actual or true classification could be for any general case.

As an example, if $S(x) > t$ then the anomaly detection scheme would label x as an anomaly - a positive output value - and as such we would be in the domain of column 2. Furthermore, if we somehow find out that this is actually not the case - that the AD scheme should have marked x as non-anomalous if operating under ideal conditions - then we would be in the domain of the 3rd row. The intersection of column 2 and row 3 singles out a single value in the table, namely that of the false positive. This is a missclassification that, ideally, should never happen.

Table 2.2: Meaning of false/true negatives/positives

Reality/Output	Label = *Positive*	Label = *Negative*
Actual = *Positive*	**TP**	**FN**
Actual = *Negative*	**FP**	**TN**

Evaluation steps

If the intersections, as described in 2.1.4 are logged in an iterative fashion for each entry of the data set, sent through the anomaly detection scheme, then a set of statistics will be produced which can be used to evaluate the scheme [38], [8], [39]. In a supervised setting, as described in 2.1.1, it is trivial to evaluate the scheme's effectiveness since one just compares the actual output with the expected output. In semi-supervised settings it is more difficult since one must use regression analysis or some other form of statistical measure to see if the output is reasonable. The most difficult AD schemes to evaluate are the unsupervised ones.

At this point we ask the reader to take a leap of faith as we claim that it is impossible to get a completely accurate and trustworthy evaluation of an unsupervised scheme. This claim is based on the question regarding why one should, could or would employ unsupervised learning to begin with: Is the performance of the scheme satisfactory with regards to the context of the domain or the end goal of the monitoring? If the answer is yes, then the scheme works just fine and vice versa. The scheme's measure of goodness, in an unsupervised setting, cannot reliably be quantified [17].

2.2 Related work

In the research community, AD is sometimes referred to as intrusion detection, novelty detection and outlier detection [44], [14]. The terms are, depending on the context of the problem formulations, interchangeable or synonymous [6]. In this section we review some of the literature regarding anomaly detection both in general and specific solutions.

2.2.1 Surveys

Throughout this report the reader will notice many references to the surveys conducted by Chandola et. al [5], Pimentel et. al [14], Hodge and Austin [6] and Patcha et. al [8]. These surveys and reviews are exhaustive with regards to the concepts and techniques underlying many AD schemes employed in the research community, describing the basics of the concept and how AD can be utilized in different domains. The surveys are broad and very general, meaning that for the context of this thesis some information contained within them is redundant. The references contained within them have however ever been invaluable and some of them are directly mirrored in this report.

In our case study we have been dealing with solely count, or discrete, data thereof the survey [21], written by the same authors as [5], has guided us as well. All surveys or reviews cited in this paragraph have a lot of overlapping information but explain the concepts somewhat differently or complementary to each other. The survey by Hodge and Austin motivates the purpose of using an ensemble in AD schemes, mainly due to their claim that combining different algorithms in an ensemble can improve performance and reliability [6].

2.2.2 Examples of approaches

This section contains short reviews of some, as we have deemed peculiar or interesting, approaches to AD found in the research community. The selection presented below is by no means exhaustive. This section aims to illustrate the wide variety of approaches conducted.

Hierarchical data

Hong et al. used data generated by customer care calls and logs from set-top boxes as case studies for their method, called "Tiresias" [36]. Tiresias is optimized to work with data that varies in sparsity, volatility and seasonality which are typical traits found in time series data, but most typically log data, for a varying set of domains. As such it could scale to different domains [45], [46].

Tiresias is an extension of previous work on "hierarchical heavy hitters" (HHH) [47], [48]. The incoming data set is filtered and clustered in regions where a certain region implies that data contained in that region has a higher probability of being anomalous. Regions with high probability of being anomalous are HHH. Tiresias works by abstracting the set of incoming data points, by organizing it in lattices, which is then divided into time units.

Once the data is separated in the time domain the identification of HHH's commences. The data is then analyzed to find patterns of seasonality. Given the time series and its seasonality, the AD scheme has all the parameters needed to find any potential anomalies. This procedure is repeated for all data sets.

The key aspects that makes Tiresias effective is that it constructs forecasting models for HHH-nodes and only checks those nodes for anomalies, ignoring the rest of the data. One of Tiresias's limitations, which applies to most AD schemes, is that it can only detect anomalies in whatever data it has access to, i.e., there is no way for the AD scheme to detect whether the data is erroneous or miscategorized.

A limitation that is specific to Tiresias is that it can only process data which is derived from an additive hierarchy. If there is a very small subset, perhaps a single point, that should be deemed anomalous, but it is in a region of low anomaly, then it will be classified as normal. Tiresias shows that clustering the input data is effective, since it limits the values needed to be processed or analyzed, but it is not always sufficient or precise enough. Tiresias, like many other algorithms, need a relatively big set of reference data to be effective [14], [11], something that in general is hard to acquire pre-execution [10], [5].

"SHARD" is a technique that analyzes input data in an ensemble manner [35]. SHARD takes a subset of the input data and analyzes it using several different methods. The authors of SHARD claim that using several methods instead of one generates fewer false positives since the ensemble is able to make more robust and general abstractions of the data than a single method can. This statement relies on the individual, or intrinsic, methods are accurate in and of themselves.

SHARD does not discriminate among the data points of the given input in the same way that Tiresias does. SHARD uses all neighboring points of the point under scrutiny during analysis. SHARD is however an offline AD scheme, meaning it cannot perform accurately in real time.

SHARD places the data in clusters or regions based on the data's label, instead of its predicted probability of being anomalous. SHARD is more computationally heavy but can expect to produce more robust, but not necessarily better, results than Tiresias. The concept of clustering the data based on its inherent labels seems as a more appropriate approach when designing AD systems for specific domains, while Tiresias (described in section seems better for a non-domain specific approach.

Network flooding

Vasilios and Fotini discusses in their article [49] an approach using an adaptive threshold to check if measurements exceeds a threshold value. This threshold is adaptive and is calculated by a mean value of past measurements. Applied to this mean algorithm is the adaptive threshold that is calculated as a percentage of the mean value. They also uses a variable to count how many consecutive measurements that exceeds the threshold. After a fixed number of consecutive exceeding measurements an alarm is triggered.

Vasilios and Fotini continues and discusses a cumulative sum algorithm that is more complex and a lot slower [49]. This algorithm is based on a logarithm likelihood ratio and another adaptive threshold. According to Vasilios and Fotini this algorithm assumes the data is distributed in a Gaussian fashion. Otherwise the adaptive threshold works in a similar way as in the mean value case.

Clustering

Portnoy describes in his article [50] a system where the algorithm starts out with empty sets of clusters and then during the process of data analysis adds incoming data points to the nearest cluster. In this way the algorithm can label the clusters and in the long run even label each data point. Clusters with few or only one data point is easy to point out as an anomaly, and some clusters containing many data points is easy to point out as normal.

Breunig et al. propose the concept of Local Outliers Factors (LOF) [37]. A LOF is a mathematical definition of a point's or value's location in the context of neighboring values. As such, it is a clustering algorithm that assigns degrees of outlierness; a high score for points that deviates from its closest, or neighboring, cluster of points.

Salvador et al. uses also a clustering approach to find anomalies in data. Their algorithm, Gecko, creates sub-clusters of the data that contain at least a certain amount of points that are non-overlapping in the time dimension. Each cluster is assigned a representative point which is used as reference to that cluster. Thereafter the clusters, or representatives, are merged with similar clusters. The number of clusters indicates how much the data varies, and in the end where the anomalies are found.

Many unsupervised AD schemes are based on clustering approaches [5]. However, care must be taken of how the clusters are generated and how they are used in subsequent calculations or analysis, as pointed out by Keigh and Lin [10]. In their article, Keigh and Lin say that the clustering of subsequence time series is meaningless, i.e., the output of these clustering algorithms is independent of the input. However, it should be noted that Keigh's and Lin's arguments supposedly only hold for subsequence time series where distance measures, between points, is based on the use of the Euclidean distance [51].

Ensembles

The concept of combining several learning algorithm's into one scheme - or in a sense, several schemes in a system - is in the machine learning community regarded as an "ensemble method", shortened as "ensemble" [17]. The motivation behind using an ensemble is that one wants to analyze data from different perspectives, i.e. looking for different things or in different ways using one set of data[33], [52], [17]. Furthermore, the reliability of one algorithm can be validated with other algorithms and in an online fashion create more accurate classifications [22].

Zhou and Goldman present a semi-supervised technique called "Democratic Co-Learning" in [52]. The technique employs a set of learning algorithms that each individually create suggestions for how to label the unlabeled data. By weighing each individual learners output the set of labeled data, used by the classifiers, is updated accordingly and the idea is that the ensemble, in a bootstrapping manner, improves its own learning abilities.

The benefit of Zhou's and Goldman's approach is that it operates in a "single-view" fashion, unlike the techniques predecessor which works in a "multi-view" fashion [33]. The difference in single- and multi-view is the amount of training data that is required; single-view models only need one set of attributes to learn its classifiers, whereas multi-view models require two or more. A benefit of using single-view approaches is that it minimizes both the amount of training data required as well as the overhead of setting up the classifiers [52].

The principles of Democratic Co-learning is that a group, or an ensemble, of several algorithms who are different in what they pertain as "non-anomalous". All algorithms have a "democratic vote" in what the final classification will be [22]. Democratic Co-learning can thus output "democratically elected anomalies" that the scheme agrees upon and whose labels are used as future reference in "upcoming elections" [29].

Chapter 3

Algorithms and implementation

3.1 Overview

This work was initiated by conducting a literature review with the intent of researching the field of AD and to get a sense of what the state of the art in the field was. Our focus and main interest was to find inspiration, knowledge and clues on how to produce an AD scheme for time series that can be used in real time. Our wish was to use a non-parametric solution that employed machine learning techniques. To gather more knowledge about machine learning we enrolled in a massive open online course on the subject [41].

Some of the literature contained methods or solutions that were not directly applicable to the domain of hierarchical time series or were to complex and domain specific to apply in our project[37], [50], [36], [35]. The complete literature review can be read in section 2.2. In the following subsections we present a general overview of our methodology.

3.2 Data structure

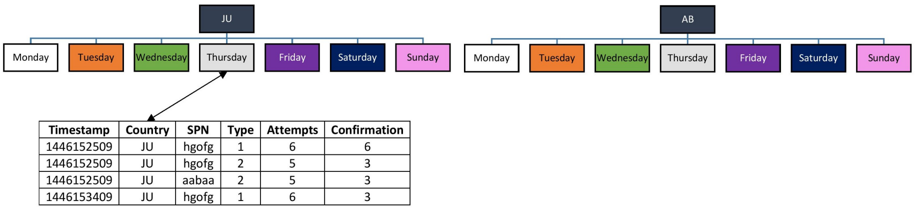
The structure chosen by Fanaee-T and Gama in [17] inspired our choice of structure. Key_1 is the root node [1]. Each value of Key_1 is always followed by a value from a set of unique Key_2-values. The Key_2-values are unique with regards to what Key_1-value that precedes it. $\forall x$,$x \in A$, $\exists y \in Key_2, y \notin (A - x)$. In other words, if Key_1 = "x" then $Key_2 = y, \forall y \in a, b, c$ and for all other values of Key_1, none of the values in (a, b, c) can follow it. In other words, the value of Key_2 implicitly dictates what the only possible value of Key1 is or can be. However, this is not true the other way around.

In the context of our case data an SPN implicitly dictates which country it operates within, since each country has a set of SPN's whose respective identification value is unique for that particular country. But we cannot know which SPN an entry belongs to just by looking at the country - unless we know that a country

[1] For the readers convenience we recommend to have read section 2.1.3 to understand the notation described in this section

only has a single SPN operating within it.

The values that Key3 can take on are limited/constant/static/fixed. Key_3 will only take on an integer value x, $x \in \mathbb{W}$, no matter what the values of Key_1 or Key_2 are. In the case data the value of Key_3 is limited to integers in the constant range $0 \leqslant x \leqslant 7$. This means that no matter what value Key_1 or Key_2 has, it is guaranteed that Key_3 will always have a value between 0 and 7. A visualization of the tree structure can be seen in Figure 3.1.



Timestamp	Country	SPN	Type	Attempts	Confirmation
1446152509	JU	hgofg	1	6	6
1446152509	JU	hgofg	2	5	3
1446152509	JU	aabaa	2	5	3
1446153409	JU	hgofg	1	6	3

Figure 3.1: Tree structure of our data

3.2.1 Initial algorithm and implementation

Parallel with the literature review we coded test benches, which became the foundation for the final framework, in order to familiarize ourselves with the data as well as learning how to code in Python, a language that we were unfamiliar with. The integrated development environment used was Jupyter Notebook [54], also known as IPython.

The case data was a single file containing over 15 million entries. The case data is further described in section 2.1.3. Using matplotlib [55], a library for Python, we were able to visualize how the different metrics of the data, as well as how our results, were distributed.

In order to determine which anomaly detection algorithms to apply we wanted to get to know the case data. When plotting the the distributions for different 3KC's we noticed that very similar distributions of values were reoccurring in similar time periods on a weekly interval. That is, the distribution of values for a 3KC on date X was almost identical to the distribution of values for the same 3KC but for the date X *minus a week* - a pattern that was repeated for several preceding weeks. This implies a mechanism that selectively finds values, in the historical data, pertaining to certain time stamps. The mechanism is described in section 3.2.2.

The distributions were right-skewed, i.e., many low values, few high values in any given time period, which can be seen in the figure 3.2.As can be seen in figure 3.2, the majority of points are located closely around the median. This arrangement was consistently seen for each metric for any combination of identifiers, keys, present in the data. That is, independent of which keys and how many of them that were combined, similar distributions emerged. When this observation was established to hold true for almost every single 3KC of the entire case data, the idea of using histograms and the median as measures for detecting anomalies was explored. Our explorations regarding these particular findings are elaborated upon in sections 3.3.3 and 3.3.5. To efficiently utilize the median for anomaly detection we used the Median Absolute Deviation (MAD) as a baseline in the Median-algorithm [42].

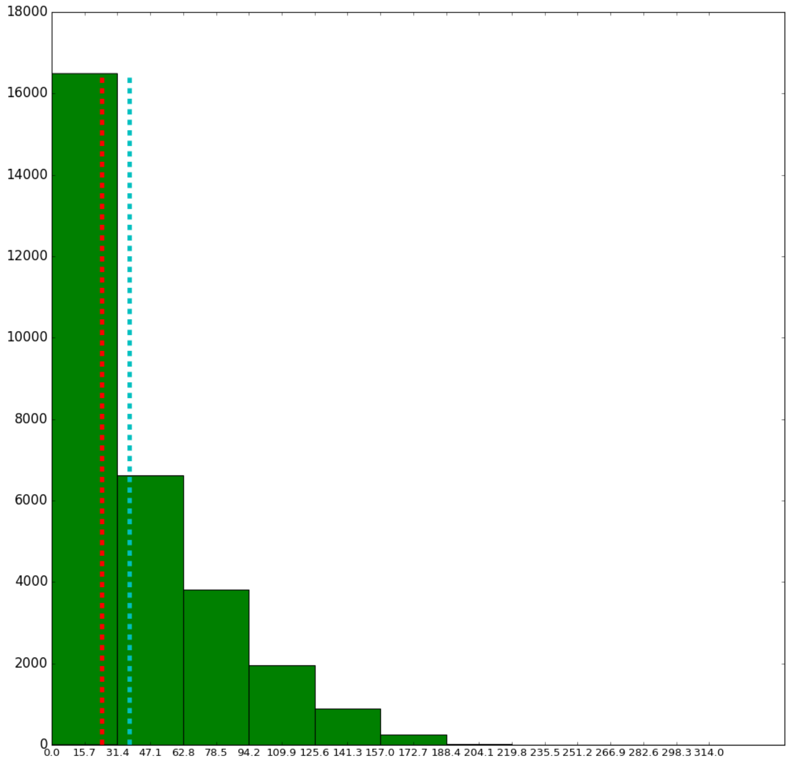


Figure 3.2: Count of number of attempts for all entries in case data for a certain country and a specific SPN. The mean (37.44) is shown by the cyan line and the median (24.0) is shown by the red line. The X-axis represents number of attempts and the Y-axis the sum of all data entries with that specific number of attempts.

3.2.2 Concept of intervals

In this section we describe the concept of fetching matching historical records. The only aspect that is omitted from the description below is that we did not fetch points that are over a year old, or 52 weeks. In other words, no consideration was taken for any historical record that is dated earlier than 52 weeks relative to the currently investigated point.

The concept of fetching entries pertaining to a certain time interval τ works in the following fashion for any given entry ϵ that is being investigated at any time:

1. The time stamp of ϵ is extracted and represented as T.
2. The identifiers, key values, of ϵ are used to find what part of the data structure, which node in the tree, to access. All entries in the historical data that match the keys of ϵ are temporarily copied to a list L.
3. All entries in L whose time stamp have the same weekday as T are kept in L, while the rest are discarded. Furthermore, all entries left in L that differ with more than $\pm\frac{\tau}{2}$ seconds from the hour, minute and second part of T is discarded.
4. All that is left in L are all historical records that match a certain key combination but which also occurred at the same time of day as ϵ - just in different weeks.

The following example illustrates the concept:

> Say that τ is 1800 seconds, which is the same as 30 minutes. Say that the time stamp T of ϵ is Friday 11th November 18:29:11 2016. Furthermore assume that the historical record's first entry for the key combination matching ϵ is Friday 1st January 00:00:00 2016. This means that all entries from all the Fridays all the way back to the 1st of January are initially extracted to L. Then, all entries from L whose time stamp has the time of day before 18:14:11 and after 18:44:11 are discarded. What is left is a list of all historical entries that match the given day of ϵ and whose time of day differs no more and no less than 15 minutes beyond the time of day of T.

The reoccurring similarity described in section 3.2.1 was not seen when there were few data points to plot, which happens in two cases. One case being that the amount of occurrences for an entry in the historical records is low, i.e., the key-combination is not being seen frequently during the lifetime of the system. The other case being that the interval of time, used as a parameter when fetching subsets of the historical records that contain the values to compare and plot, is small. A small interval time means that the set of points to fetch, that match a key combination, correlates to a smaller set of acceptable time stamps. A large interval means that the spectrum of points, if viewed as a time line, becomes wider which increases the amount of points that the spectrum encompasses compared to a smaller spectrum. In our implementation we decided upon using an interval length of 30 minutes. Interval lengths above 30 minutes generates distributions that are redundant and interval lengths below 30 minutes generated indistinguishable patterns that were in many cases impossible to match to a certain random process.

3.2.3 Probabilistic approach

We decided to extend our AD scheme using a probabilistic approach based on the assumption that the data can be modeled using univariate regression analysis. This assumption was made based on the way that we access the data using key combinations and time stamps. It meant that we can create samples of count data where each sample was indicative of the expected value.

Based on our own perception, previous knowledge of statistics and with the intent of not creating a scheme that was distinctively modeled around our case data (i.e. generic enough to be used with other data sets, but still applicable in our domain) we explicitly defined the distribution as a Poisson process. This assumption established the starting point for implementing the "Poisson"-algorithm, as described in 3.3.2.

Uncertainty regarding Poisson

At a very late stage in the case study we noticed that our assumptions about the data being generated by a Poisson process might have been too naive. We noticed that the historical data that was used in the algorithms showed a high measure of over-dispersion. This was expected since it is unknown how many outliers or anomalies that the data contains.

Over-dispersed count data occur regularly in real applications. A popular remedy to this problem, in order to model the data more properly, is to assume that the distribution is negative-binomial [40]. However, the results produced by the Poisson-algorithm were not unsatisfactory and we consciously wanted the principles governing our AD scheme to be as general as possible as well as relying on as few parameters, or background knowledge [6], as possible.

There was no attempt made to change our model based on this and we additionally expected that the results would not have been noticeably different since, as stated in [40]: "*There is no evidence that using the negative binomial distribution instead of the Poisson distribution helps when the systematic part of a regression model is misspecified*".

In order to nuance our ensemble approach we decided upon using both the Poisson-algorithm as well as the "Bucket"-algorithm, as explained in section 3.3.3. They will both count the distinct amount of the metrics and use the most frequently occurring value as a baseline. These two algorithms can be seen as an alternatives to each other since they will, intuitively, behave similarly. The difference is that the Bucket-algorithm will not use a probability mass function and the Bucket-algorithm is a clustering approach, whereas the Poisson-algorithm is a statistical approach to the problem of AD [5], [8].

3.2.4 Averages

As mentioned in section 3.2.1, the median was deemed to play a big role in establishing where non-anomalies are likely to be found. In order to efficiently use the median as an influence or baseline we developed the Median-algorithm as described in section 3.3.5. One thing that was noted in section 3.2.1 was that the most frequent value was set around the median. However, the values were not centered, or normally distributed around the median. Instead they were left-skewed meaning that the most frequent value landed, more often than not, to the right of the median and the mean. See figures 4.3 and 4.4 for visualization of the distribution.

An algorithm that uses the mean as a baseline would suffice as an alternative not only to the median-algorithm but also to the Poisson-algorithm which uses the expected value, the mean, as a parameter. However, in order to not make the combination of the median and a mean algorithm redundant, or making them "cancel" each other out, we implemented a different way for the mean-algorithm to handle the reference data.

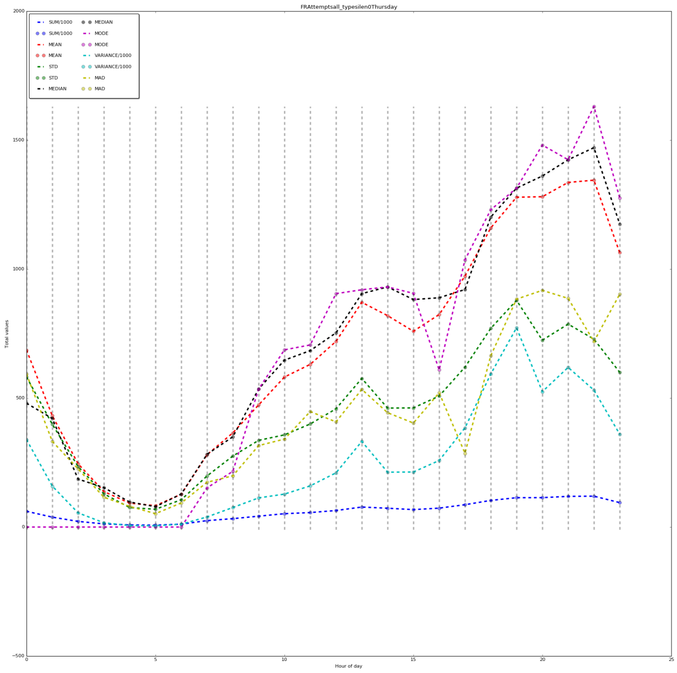


Figure 3.3: Statistics for number of attempts for a 24 hour period on a Thursday in France

Instead of matching the currently inspected value to the complete set of matching historical records[2] the Mean-algorithm creates weekly averages for set of matching historical records. This means that one can interpret the mean-algorithm as a hybrid between a clustering approach and a statistical approach. This is so since the Mean-algorithm does not consider all of the data points individually when looking for anomalies, but aggregates them in smaller clusters where each cluster is the mean for each week respectively. The Mean-algorithm assumes that the weekly means are normally distributed which can be assumed by seeing that the line, shown in figure 3.3, is more or less constant, i.e., the mean for each week is regular.

3.2.5 Designing the ensemble

The four so far described algorithms work independently of each other. The output of each respective algorithm does in no way influence any other. However all algorithms have major pitfalls or cons which need to be mitigated. This is where the idea of using an ensemble comes in. We executed the scheme using our four algorithms and collected the resulting scores, for each level and metric respectively, for each entry in the simulation data. Thus we were able to map each entry to its corresponding scores.

We noticed that in some cases all algorithms agreed upon assigning, more or less, the same score. But there were also many cases where there was extreme disagreement between each respective output. The discord in the score distribution for each entry was many times irregular and not showing discernible patterns. For example, one algorithm could give the score 0.0 while all other algorithms gave the score 1.0 and vice versa.

After investigating both high and low scored entries we were able to discern the following: The higher the similarity between scores are, the more reasonable the score seemed to be, i.e., when all algorithms produced low scores, the scored entry could be seen - when plotted together with the matching historical records - as belonging to the majority of points. The opposite was seen for entries with high scores and high agreement, i.e, it was clear that the point was a distant outlier with regards to its matching historical records.

This finding, that high agreement correlates to a seemingly correct classification, was however only clear when the set of matching historical records contained many points. For sets of sizes $\lesssim 15$ the agreement was rarely high and for sets of sizes $\gtrsim 80$ the agreement was often high, which was expected considering that the quality of a sample is probably more representative if it contains many entries[6]. As such we discerned the following two assumptions as specifications for creating a trustworthy, or reliable, ensemble:

1. The amount of entries in the matching historical records must be high

[2] Which every other algorithm in our ensemble does.

2. The variance of the scores, produced by the four algorithms, must be small

The amount of entries and the variance of the scores are combined with scores of the previously described algorithms. Details on exactly how this combination, the ensemble, produces the final output of the scheme is described in section 3.3.6. Our ensemble method, which we hereafter will call "the Normalization-algorithm", has a bias or a pitfall which arises when any, or both, of the the above assumptions do not hold. Furthermore, the Normalization-algorithm does not consider the risk of the other algorithms scoring "wrong" when labeling instances[3].

It is at this point the prerogative of how the trade-off between false positives and false negatives comes in to play. Should scores, exemplifying low reliability, be weighed towards a negative or a positive final score? Is the life of the system, that is being monitored, highly sensitive to anomalies or does the system work efficiently even if some anomalies go undetected?

Basically, and pessimistically, we have to ask ourselves what is worse: Too many false positives or too many false negatives? In section 3.3.6 we show that the Normalization-algorithm is wary of assigning high scores and more likely to assign low scores by how the variance and the size of the matching historical records are used in producing a final score. In other words we chose to implement the Normalization-algorithm such that false negatives are more fine, or forgivable, than false positives.

3.2.6 Evaluation

Once we had developed a working prototype of our anomaly detection scheme, i.e. we could iterate through all of our case data and produce scores for each entry, we sent our case data to a third party [56], described in section 4.4.1. The third party is a company that has a patented method for finding, analyzing and scoring anomalies in time series data [57]. Even though their results could not be directly compared with our result, as in a supervised fashion, their results were used to investigate our scheme's effectiveness and reliability.

The results from the third party could not be directly compared with our results because they mark and label anomalous behavior on a collective basis, i.e. over several time instances, whereas we label anomalies on a point basis. However, their results were used to investigate where and when anomalous behavior should, could or might be spotted in our system. i.e. the output of the systems are different, but they both give clues as to when and where in the data the anomalies might be occurring.

The output of their anomaly detection is a reference to a specific key, i.e. a set of entries, of the data and for how long the anomaly lasted. This output, visualized in figures 4.6 and 4.7, made it very easy for us to manually investigate the top

[3] This risk would however be pursued if operating in a supervised fashion.

20 anomalous entries found by them. First we logged the key and time period of the anomalies spotted by them. Then using this information we could, from our own system, extract all the point scores for the same set of points used to produce their scores.

In all cases of the 20 investigated anomalies, as marked by the third party, our anomaly detection scheme also gave high scores. This investigation was by no means exhaustive but we consider it more than enough to assert whether we had produced trustworthy results or not.

3.2.7 Summary

The following underlying assertions serves as a basis for our general methodology and solution:

- Historical entries without labels will be the reference data.
- An effort to produce naive estimation models for the data will be conducted.
- An ensemble of 4 static statistical algorithms combined in an ensemble manner will produce the final output(s).
- Each level in the hierarchy[4] will get a score for each metric for each entry investigated.
- The Normalization-algorithm will produce the output of the scheme and will be use the variance of scores as it primary parameter.
- Our findings will be cross-checked with a third party and our most concrete results (consensus in scores) will be treated as true results.

3.2.8 Aggregation of data

At each level, i.e. key combination, and time stamp the metrics can be aggregated. In the level containing the combination of all three keys there is no aggregation since for each unique time stamp there can only be at most one single entry that contains unique key-values for the level that uses all keys.

In a level that contain fewer key-combinations the maximum amount of entries, for a specific time stamp, is the number of unique, and valid, combinations of values for a particular level multiplied by the number of unique and valid values that a subsequent key can take on. This is why several entries share the exact same time stamp, as well as perhaps Key1 and Key2, but are still distinctive entries because of the different key3-values[5].

The only instance of two entries with the same key-values for all 3KC can thus only exist at a different time stamp. This can be seen in 2.1 when comparing row 1 and row 5. The time stamp is implicitly used as an identifier but in the data

[4] Namely the key combinations, as explained in section 2.1.3, chosen to investigate

[5] This description holds for any example where the value for all but one key is static.

structure that models and contains the case data it is not regarded as a key.

When aggregating or permuting the keys it is important to keep track of the metric values. As mentioned above, the aggregated metric values for a level with 3KC's is just the value of the metrics as they are in the entry. However, at different levels there can be several entries present for one key combination at any given time stamp. As such, the metric values have to be summed to represent the metric values for that key combination.

As an example, one can see that on the level of $Key_1 \wedge Key_3$ for the Values $Key_1 =' JU', Key_3 = 2$ the number of attempts and confirmations is the sum of the values in the metrics' corresponding column for each entry matching the key combination, namely 17 attempts and 11 confirmations (row 2, 3 and 6). However for the similar level, or 3KC, of $Key_1 \wedge Key_2 \wedge Key_3$ for the Values $Key_1 =' JU', Key_2 =' hgofg', Key_3 = 2$ the aggregated values for attempts and confirmations are 12 and 8 (row 2 and 6) respectively. Furthermore at the level of Key_1 with the value $Key_1 =' JU'$ the aggregated values are 37 and 21 for attempts and confirmations respectively (all rows except row 4 and 8).

As described in 3.1 the data exhibited evidence of being seasonal over the weeks at a certain key. We took advantage of this by dividing the historical data into keys, as described in section 2.1.3, and weekdays. That way we enabled the functionality of easily find comparable values or points that neighbors the incoming point, the point under investigation. As such, our AD scheme will be operating using similarity-based matching[6]. We will match current values against previous, historical, values within a specific context - the matching keys.

3.3 Scoring algorithms

The incoming data points were compared with historical data in five steps - four algorithm executing after another and a final step where a normalized or summarized score is conceived in an ensemble manner. All five distinctive steps are described below respectively subsection. The score from each algorithm is a decimal value between 0.0 and 1.0. The higher score, the more anomalous that algorithm considers the data point to be, as per the concept introduced in section 2.1.4.

It is important to note that even though an entry in the case data contains more than one metric, our approach treats the metrics in a univariate fashion. As such, we assume for simplicity that each entry in the case data can be divided into 2 entries (on for each metric) and, from the scheme's perspective, each metric is handled individually.

When producing the final score, as described in section 3.3.6, the scheme is operating under the same assumptions as presented in "Democratic Co-Learning" [52] which is that different algorithms have different inductive biases, for the same data, regarding what constitutes as anomalous. The biases for each respective algorithms will be elaborated upon in the sections below.

These biases are based on prior knowledge about the data which enables the setup of parameters that we describe. The methodology which underlies our chosen implementation is connected to that of "accommodation", as described by Hodge and Austin in [6]. Furthermore, our ensemble follows the "wisdom criteria"[6] proposed by James M. Surowiecki in the book "The Wisdom Of Crowds"[62], as paraphrased in [22].

3.3.1 Framework functionality overview

Figure 3.4 is a visual representation of how our AD framework operates. Please note that after a score has been produced the step of classification is actually applied by comparing the score with a threshold. The classification step only tangents the scope of this thesis, as such a shallow insight into how well our implementation classifies all entries are given in section 4.3. Please note that the incoming data point is added to the historical records *after* it has been used in each algorithm.

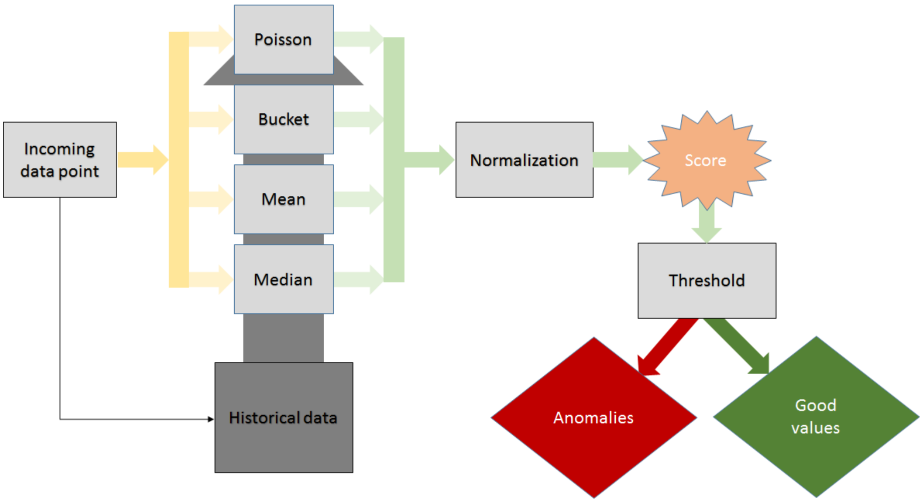


Figure 3.4: Visualization of the algorithm

3.3.2 Poisson-algorithm

The metrics in our case data can be modeled as successes in a trial. By furthermore assuming that the most frequently occurring values are the most nominal ones - and the other way around, that values which rarely or never occur are outliers - we can produce the setting required for modeling a Poisson random process. Unlike the histogram solution presented in 3.3.3 we do not cluster all values in a fixed set of 10 bins or buckets. Since the data is discrete we can however analogously say

[6] Diversity of opinion, independence, decentralization and aggregation

that the Poisson-algorithm models the data as a histogram with X bins, where X is the highest value of all entries in the data set being modeled, i.e. the matching historical records of an entry.

If the historical records contain points with extreme values, or few points to begin with, then the modeled distribution becomes hard to interpret and the analogy to the bucket algorithm less palpable. We demonstrate this by describing the modeling of the following two, somewhat similar sets:

1. 0,1,2,3,3,3,3,3,5,6,7,9
2. 0,1,2,3,3,3,3,3,4,5,6,22

Both sets are of equal magnitude - they both contain 12 values. The values in the first set range from 0 to 9, thus creating 10 bins assigning a probability to the omitted values 4 and 8 (i.e., the values within the range but not part of the actual set). The values in the second set range from 0 to 22 which creates 23 bins and assigning probabilities to the omitted values - the range from 7 to 21. The value 3 is the most frequently occurring value as it occurs 5 times. In both sets its frequency is $\frac{5}{12} \approx 0.42$ whereas all other values frequency is $\frac{1}{12} \approx 0.08$. This can be interpreted as the following: "The likelihood that a value in the set is equal to 3 is 42% while the likelihood that the value is not 3 is 8%" which holds true for both sets. We are however modeling the sets as Random Poisson variables. Therefore we reach a different conclusion since we are determining the probabilities of a value's likelihood using a probability mass functions which take the mean value of the set as a parameter. The probability mass function (PMF) for a Poisson distribution is defined in equation 3.1.

$$P\left(x, \lambda\right) = \frac{e^{-\lambda}\lambda^{x}}{x!} \tag{3.1}$$

The currently inspected value is x and the mean of the set is λ. In our example $\lambda_{set1} = 3.75$ and $\lambda_{set2} \approx 4.58$. If for example $x = 3$ then for each respective set the probabilities would be $P_1 = P\left(3, 3.75\right) \approx 0.21$ and $P_2 = P\left(3, 4.58\right) \approx 0.16$. What can be noted is that in both cases the probability is very low. Even though the value 3 is by far the most likely value the probability of it occuring is very low - even though it is the highest probability that can be acquired from the given sets[7]. This makes the probability inappropriate to use as a direct score since we want the score to be somewhat more normalized value which shows how close, or far away, a value is to being nominal. To achieve this end we utilized a feature of the Poisson PMF[8], namely that the highest probability that the PMF can assign is that when $x = \lfloor \lambda \rfloor$.

The value that the PMF outputs when the parameters are $P\left(\lfloor \lambda \rfloor, \lambda\right)$ is called the reference probability, notated here as P_{ref} and shown in equation 3.2. The

[7] The proof is left as an exercise for the reader.

[8] At least according to our own tests.

P_{ref} value is used to create a ratio between its value and the corresponding P value. If they are similar, i.e, $P(\lfloor\lambda\rfloor, \lambda) \approx P(x, \lambda)$, the ratio will be close to or equal 1 and if they differ a lot the ratio will be close to 0. To produce a final score the ratio is subtracted from 1 so as to give low scores to alike values and high scores for non similar values. The formula is described in equation 3.3, where S stands for the score value and P stands for the PMF given a value x and its historical record's λ.

$$P_{ref} = \frac{e^{-\lambda}\lambda^{\lfloor\lambda\rfloor}}{\lfloor\lambda\rfloor!} \tag{3.2}$$

$$S(x) = 1 - \frac{\min(P_{ref}, P)}{\max(P_{ref}, P)} \tag{3.3}$$

To continue our example above, $P_{ref1} \approx 0.21$ and $P_{ref2} \approx 0.19$. Now we have all the values needed to calculate the scores which are respectively, for $x = 3$, $S_1 = 1 - \frac{0.21}{0.21} = 0$ and $S_2 = 1 - \frac{0.16}{0.19} \approx 1 - 0.84 = 0.16$. For the second set there is no difference in the resulting score and the PMF value, whereas the second set gives it the absolute minimum score. If $x = 2$ the scores would have been $S_1 = 1 - \frac{0.17}{0.21} \approx 0.19$ and $S_2 = 1 - \frac{0.11}{0.19} \approx 0.42$ respectively showing that the further x deviates from $\lfloor\lambda\rfloor$, the score is higher than the PMF-value for the given x thus severely "punishing" outliers - a punishment which is escalated with the amount of anomalies in the matching set. This means that as the range of values in the matching set increases, i.e., the more anomalies that are present, the more severe is the punishment if a value is not close to the mean. This does however open up for the risk of unintended learning, a term further explained in 5.1.1.

The need for a "normalization"-step, using a reference probability, becomes more important as the ranges of values in a set increases. In large sets whose values span between, say, 0 and 200 there might be an evident nominal value to treat as a baseline. But its PMF-value, as well as its neighboring values, will be very low meaning that the probability in itself is not a good enough metric for a score. A benefit of using reference probabilities in our way is that the scoring can scale to sets whose distributions deviate from that of a Poisson random process since we do not solely rely on a probability - but how well it fits into the context of other probabilities.

3.3.3 Bucket clustering

The algorithm takes all the historical data's attempts and creates ten equally wide buckets. Each bucket is then "filled" with each point that have the value between a and b. this creates buckets of different height, as seen in Figure 3.5. The higher the bucket is the more is the likeliness of the incoming new data point to land in the bucket.

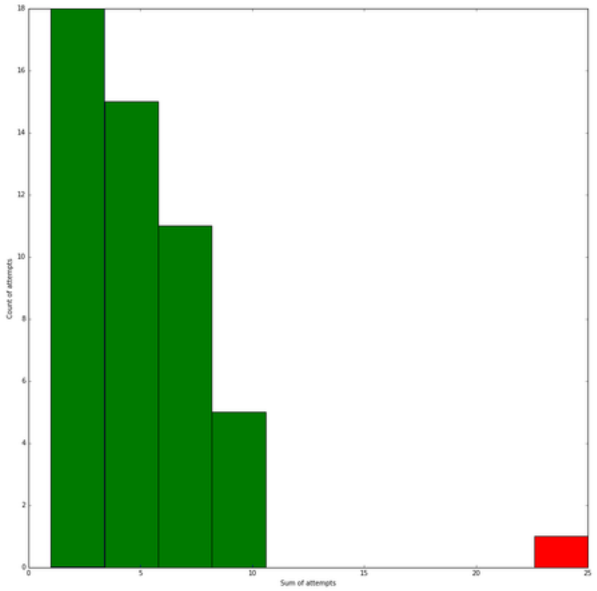


Figure 3.5: Visual representation of the bucket clustering algorithm

By calculating the height of the bucket in which the incoming data point *falls* compared with the sum of all heights a score value between 0.0 and 1.0 can be established. This value represents how many historical points that matches the incoming data point's cluster.

3.3.4 Mean

By calculating the standard deviation σ of a set of points and using it as a factor in establishing the boundaries or thresholds for out- or inliers with regards to the mean, this algorithm gives low scores to values who are close to the mean. The standard deviation is calculated in the way described in equation 3.4. In a set where the values are distributed on a large range, such that the mean is not the only obvious baseline but neighboring values as well, the need to loose the constraints set by σ is utilized by multiplying a constant factor to σ thus establishing the upper and lower bounds of how much a value can deviate from the men.

The calculation of the score is presented in equation 3.5, where x is the value being investigated, μ is the mean of the matching set and σ is the standard deviation of the same set as calculated in equation 3.4. N is the number of elements in the set and y_i represents each individual value in the distribution set.

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(y_i - \mu)^2} \tag{3.4}$$

$$S(x) = \min(\frac{|x-\mu|}{c * \sigma}, 1.0) \tag{3.5}$$

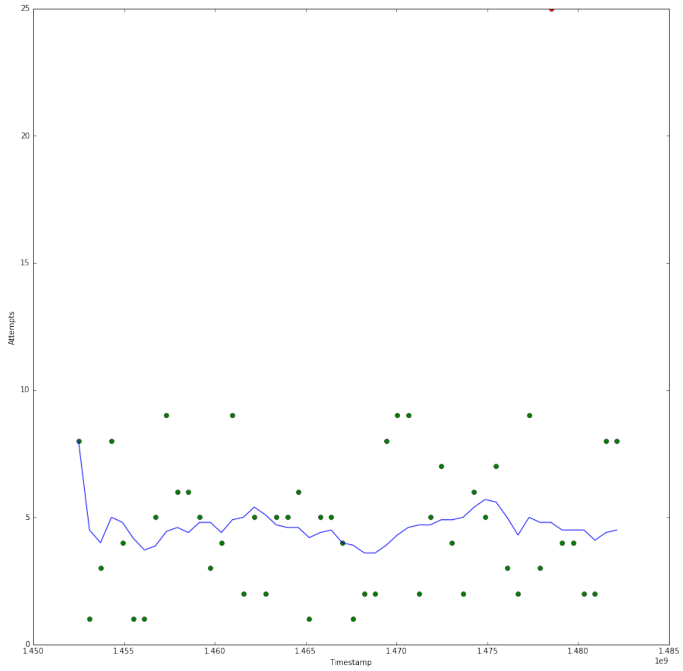


Figure 3.6: Visual representation of the mean algorithm

In equation 3.5 one sees that if the difference between the currently inspected value x and μ is larger than $c * \sigma$ the score will automatically be set to 1, independent of the actual value of $|x - \mu|$. This is a limitation in the scheme that is further described in section 5.1. In a normal distribution, the standard deviation multiplied by 1.96 covers the range of values where 95% of the data around the mean resides. In our implementation $c = 1.96$ and as such the concept of the Mean-algorithm is that "all values x which deviates more than $1.96 * \sigma$ are guaranteed to be anomalies and values which deviate less are given a score, where a low score, $S \rightarrow 0$, indicates that the value x is close to the mean and thus non-anomalous".

The set of values which the distribution is modeled around differs in one very important aspect. For all other algorithms all of the individual matching historical entries are used to create the distribution of values which serves as the baseline for comparison. The Mean-algorithm initially access the same set of points as all other algorithms, as per the description in section 3.2.2, but before a baseline distribution is created a clustering step is performed.

The clustering step in the Mean-algorithm's case is that it separates the matching historical records in weekly intervals and then takes the mean of the values pertaining to each week. The mean for each week thus models the distribution which means that the amount of values in the baseline distribution is the same as the amount of weeks that the matching historical records span independently of how many entries there actually are in the matching historical records.

If the historical records skips weeks, i.e., in some weeks there are no records whatsoever then no value for the baseline distribution is entered for the missing weeks. One could argue that instead of omitting entries as our implementation does, a mean value of 0 should be given for a missing week. This would however infer that one expects values each week, which is not necessarily true, as well as potentially skewing the baseline distribution too much to the left, towards 0, if several weeks are skipped.

3.3.5 Median

By calculating the median absolute deviation (MAD) of a set of points and using it as a factor in establishing the boundaries or thresholds for out- or inliers with regards to the median, this algorithm gives low scores to values who are close to the median. The MAD is calculated in the way described in equation 3.6. In a set where the values are distributed on a large range, such that the median is not the only obvious baseline but neighboring values as well, the need to loose the constraints of the MAD is utilized by multiplying a constant factor to the MAD thus establishing the upper and lower bounds of how much a value can deviate from the median.

The calculation of the score is presented in equation 3.7, where x is the value being investigated, m is the median of the matching set and MAD is the MAD of the same set as calculated in equation 3.6. Y represents all the values in the matching set and y_i represents each unique value of the set Y.

$$MAD = median(|Y_i - median(Y)|) \tag{3.6}$$

$$S(x) = \min(\frac{|x - m|}{MAD * c}, 1) \tag{3.7}$$

As can be seen in equation 3.7, if the difference between the currently inspected value x and m is larger than $c * MAD$ the score will automatically be set to 1, independent of the actual value of $|x - m|$. This is a limitation in the scheme that is further described in section 5.1.

In the setting of the Median-algorithm we are assuming that the data is distributed in a non-normal fashion. As such, the standard deviation, as used will not be particularly helpful since it will only produce reliable results if the median $\approx$ mean, which almost never happens in our case data. The MAD can however be used analogously in the same way as the standard deviation is used for confidence intervals. Please note that the multiplicative factor is $c = 1.4826$ for normally distributed data sets[9] in order to estimate the standard deviation[43].

We are however not interested in estimating the standard deviation but instead using the MAD to figure out if a value is an outlier relative to the median. As such c becomes the parameter that regulates the sensitivity of the Median-algorithm.

[9] A "standard" constant in the same way as 1.96 is for normal distributions.

A high value of c means that the range of values that a value can take on and not be an outlier increases and vice versa. Our framework permits setting c to any value, as mentioned in section 3.5.

In our implementation and for the results we show here we chose $c = 1.96$. Please note that this does not imply that we used a 95% confidence interval, we did however want to see if using a constant reserved for normally distributed data could work well in a non-normal, median centered, setting. When testing with much lower values of c we noted that many ordinary, as per visual inspection, points were marked as anomalous and the opposite when c was much higher. Out of convenience we stuck with 1.96.

Please note that whatever value c is, a major drawbacks or bias of the Median-algorithm is revealed with is with regards to skewing. If the entire distribution is left- or right-skewed one might reasonably expect that the distribution of points around the median, within the $c * MAD$ range, are also skewed similarly. The Median-algorithm ignores the skew and as such, might be too lenient or harsh when scoring, depending on which part of the median-centered sub distribution an investigated value falls within.

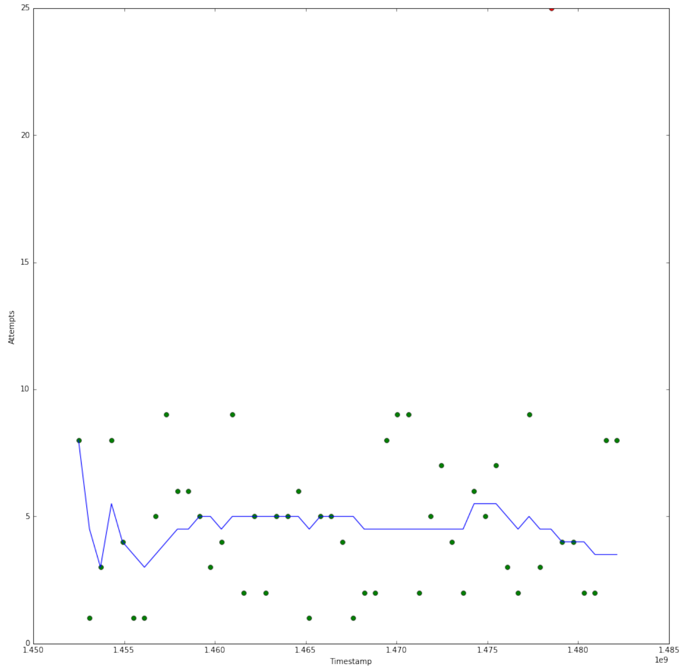


Figure 3.7: Visual representation of the median algorithm

3.3.6 Normalization

As explained in section 3.2.5 this algorithm takes as input each individual algorithm's score a_i, for each level and metric, as well as the number of elements N in each distribution set.This section will describe the actual calculation steps in producing a final ensemble score. The concepts and ideas underlying our implementation is further described in section 3.4. The Normalization-algorithm serves only as a last filter and does not take into consideration either the value being investigated nor the matching historical records.

Equation 3.8 shows how the scores are created for each level L and metric M investigated for each other algorithm's score a_i. Equation 3.8 is performed for each element, or score, a_i and each result f_i is put in the set F, which will be used in equation 3.9. $\bar{F}$ is the average of all values in F and V is shorthand for $Var[F]$ which stands for the variance of the values in F. The reason for using $\bar{F}$ is to make the calculation adhere to a single value, a baseline. The choice of the mean, instead of any other summarized representation of the scores, is motivated by the idea of having a fair representation of where all scores would, or could, converge - the common, average, ground.

$$f_i = (1 - \frac{1}{N}) * a_i \tag{3.8}$$

$$S_{L,M} = |\bar{F} - V| \tag{3.9}$$

By investigating the two above mentioned equations we see that if N is very high, then the resulting difference between f_i and a_i is marginal or negligible which emphasizes the point that a larger distribution set is less likely to "force" the final score to be low. Another factor that decreases the score is V which emphasizes the point that higher disagreement between the scoring algorithms correlates with lower final scores.

As can be seen in equation 3.9 we take the absolute value of the difference between the mean and variance of F. This is to ensure that the resulting final score is not negative. Furthermore, if $V \ll \bar{F}$ then the value of $S_{L,M}$ can "swing" up to a high score, which is justified by the principles dictated in section 3.2.5.

3.4 Score committee

The division of labor described in the beginning paragraph of section 3.3 is similar to co-training [33], except that we do not consider labeled data. Co-training is an approach which divides data sets in one or more or two sub-sets, sends the data to the algorithms which in turn teaches another, or the same, classifiers. In the same way the metrics are seen as (two) distinct sub-sets of the data. These sets are sent to each algorithm respectively as described above and each individual score or result is joined to produce a final result [29].

Our approach is has been influenced by the work behind "Democratic Co-Learning"

[52] and "Weighted Majority Rules" (WMR) [53]. However, unlike Democratic Co-Learning which utilizes confidence intervals in order to eliminate wrong predictions [52], and WMR using game theoretic approximations, our ensemble approach assumes labeling criteria holds throughout the execution of the scheme. Unlike the approach described in [52] we did not have access to labeled data and as such we presumed that usage of confidence intervals would introduce ambiguity or unnecessary complexity with regards to evaluating correctness.

We by-pass the need for confidence intervals while still introducing a weighing factor that diminishes the score based on the distribution of the algorithms' scores. The weighing factor is the variance of the scores. The variance is subtracted from the mean score and in doing that, the ensemble score is forced to be lower if there is a high disagreement between the algorithms.

The difference between co-learning and our ensemble, which we here call score committee, is that we do not let the labels influence future predictions, thus maintaining an unsupervised setting [29]. In short, we have borrowed techniques used in semi-supervised machine learning but applying them in an unsupervised setting to the best of our abilities. As noted in the subsections of section 3.3, the algorithms all have different characteristics, something that is desirable in an ensemble [17].

Our starting point in creating the AD algorithms are based on knowledge about the data which arose during our investigations of the values. From our investigations we have drawn conclusions depending on the distributions of the data. This makes our methodology analogous to a supervised or semi-supervised approach. However the definition of unsupervised AD scheme only requires no knowledge about labels [17].

By combining techniques in Co-learning, unsupervised anomaly detection and knowledge of the data set structures we have created a new way to conjoin algorithms into an anomaly detection scheme. The framework is general and will be able to fit into different systems with minor tuning of parameters. As such the novelty of our implementation is not only that it uses very basic statistical measures to provide a reasonable output but it is modular enough to work in any setting operating on time series data.

3.5 Configuration parameters

This section will account for the parameters that regulate the general execution of the entire AD system. Each point in the list below represent a parameter that can be changed before executing the scheme. For each parameter we outline our chosen values which led to the results presented in the report.

- **Index of split** - In order to test our implementation, the case data was split in to two halves. The first half contained the first 7 million entries of the case data and will hereafter be referred to as the historical data. The other 8xxxx rows was used as "simulation data" or input data used to test our scheme. The setup of our scheme was thus initiated by filling our data structure with the historical data and then afterwards the AD scheme was executed using the input data as "online" data.

- **Interval length** - The value of τ as described in section 3.2.2
- **Levels of interest** - The choice of key aggregations/permutations to consider when scanning the data. The parameter is set by indicating what keys to inspect for each successive level. If the parameter is set as an array with the following two elements (which are also arrays) $\{0, 2\}$ and $\{1-2\}$ it means that one level is to be regarded as the permutation of the first and third key in the order they show up in the case data. In our case data the first (0) and third (2) entry were the country and type. In the second level, the second and third keys are permuted meaning that the second level would look at the SPN as well as the type. Note that in our case data the 0 is implicitly inferred, as per our description in section 3.2. This means that any initial value of 0 is redundant. In other words, the example described here is equivalent with $\{2\}$ and $\{0, 1, 2\}$. Please note that this is a special case for our case data and might not fit in cases where subsequent keys, to the initial key, connect to differing values of the first key.
- **Index of levels** - In the description of "Levels of interest" above we only considered the *index* of each respective level. The default indexing of the levels are in the way they occur in the data set studied, i.e., index 0 represents whatever key that is first in the data set. This can however be modified in way which tells the framework that "the first key" is actually any other key. In our implementation this parameter is set to $\{1, 2, 3\}$ meaning that key 0 will refer to the 2nd column in the data etc. This is how and why we do not use the time stamp of the case data as an explicit key which would be the case if this parameter had the value 0 in its list. Basically, this parameter dictates not only which keys in the case data to treat as actual keys in the framework, but also in what order they should be accessed.
- **Index of metrics** - Same usage as "Index of levels" but pertaining to those columns in the case data which hold the metric values. In our implementation this parameter was set to $\{4, 5\}$, indicating that the 5th and 6th column contains the values representing the metrics.
- **Algorithms of interest** - A list indicating which algorithms to actually deploy during execution. This parameter can be omitted for the cases when the only algorithms of interest are the ones already implemented. To preserve modularity and testing this parameter can be modified. For example, one might want to change the percentile points used in our Mean- and Median-algorithms.
- **True threshold** - A percentage which indicates what the score value must be for all algorithms in the ensemble for the total score to be regarded as a true result. The percentage sets the lower and upper bound for what

the consensus score must be in order for the score to be regarded as a true positive or true negative respectively. If the consensus score is between these bounds the results are deemed "unsure". In our implementation this is set to 10% indicating that consensus scores below 0.1 are regarded as true negatives and consensus scores above 0.9 indicate true positives.

- **Confidence factor** - A value which is multiplied with the standard deviation or MAD, for the Mean- and Median-algorithms respectively, to infer upper and lower bounds for detecting in- and outliers during AD. In our implementation this is set to 1.96

Chapter 4

Evaluation and result

4.1 Overview

The short answers to our research questions, stated in section 1.2.2, are yes. However, an answer to such complicated question is seldom that elemental. Our research and tests shows that with great knowledge of the data set a couple of algorithms can be created and tuned to get good results. Nevertheless these algorithms are fine tuned just for our case and needs to be retuned for the next case. This creates a work load for each new type of data set further implying that our implementation needs future improvement to function in a completely general setting. Some suggestions to future modifications and additions are mentioned in 5.2 and in this chapter we only highlight the results of our findings.

In the following subsections we shortly summarize the results from each algorithm and finish the chapter with our comparison as well as presenting the results of using our rudimentary classification thresholds. Our implementation is based on some presets or predefined settings which are outlined in section 3.5.

As will be noted in the graphs below, depicting the results from each individual scoring algorithm for the level of SPN and type and for the metric Attempts, is that the highest scores peaks in the score distributions. The only exception is provided by the Normalization-algorithm.

From the score distribution of the Normalization-algorithm we can conclude that the algorithms rarely agreed on providing consensus scores, i.e., even if many of the individual algorithms found anomalies the committee could not, in many cases, agree. This would, as explained in section 3.2.5, force the score to be low.

It must be noted that our implementation was based in an unsupervised setting which means that an external party has to validate our results. Since we were unable to acquire a reference data set to compare our results with this chapter will not include an analysis with regards to true and false positives and negatives as described in section 2.1.4.

4.2 Visual inspection

In the following subsections we produce our results based on how each algorithm has scored all the input data. The scores are clustered in histograms to simplify visual inspection. The values we show are the scores for the level of SPN's and types and for the metric number of attempts.

4.2.1 Poisson

The Poisson algorithm creates very evenly distributed scores, except the end values. As seen in the Figure 4.1, both the lowest and the highest scores are overrepresented.

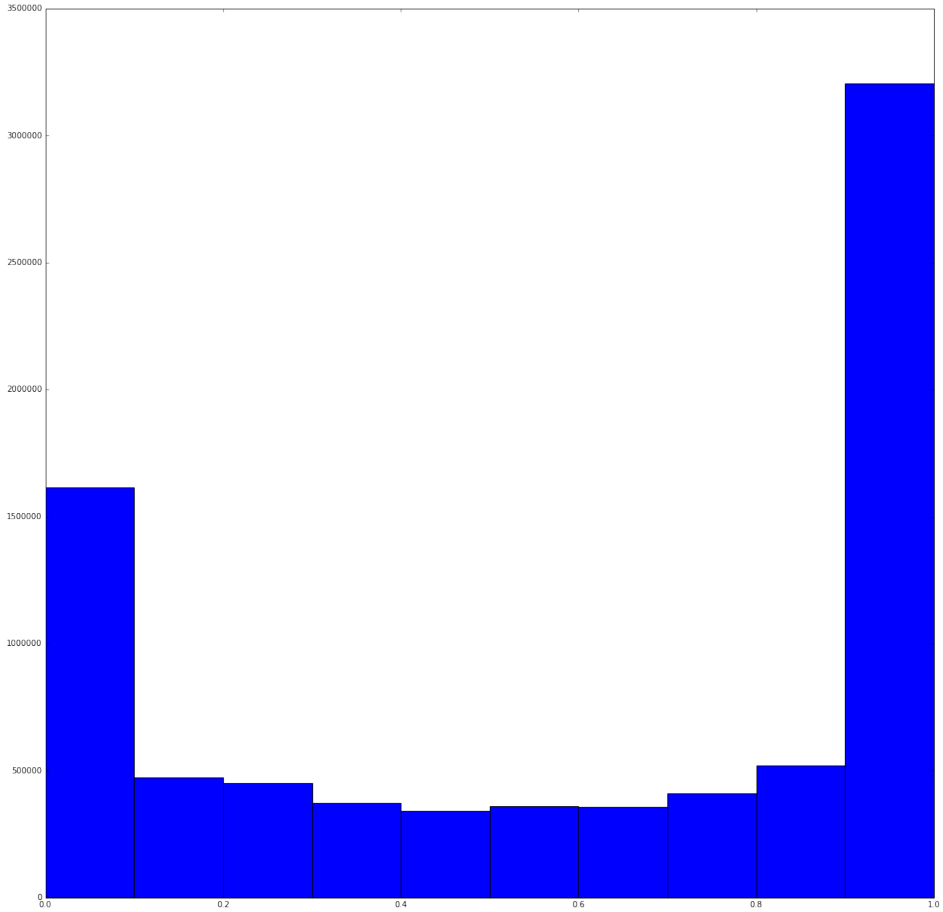


Figure 4.1: Poisson algorithm, distribution of scores

4.2.2 Bucket

As seen in Figure 4.2 the bucket algorithm gives divergent result. Most of the values are fairly high, and some are really low, this is probably because the algorithm compares ten buckets to create its score.

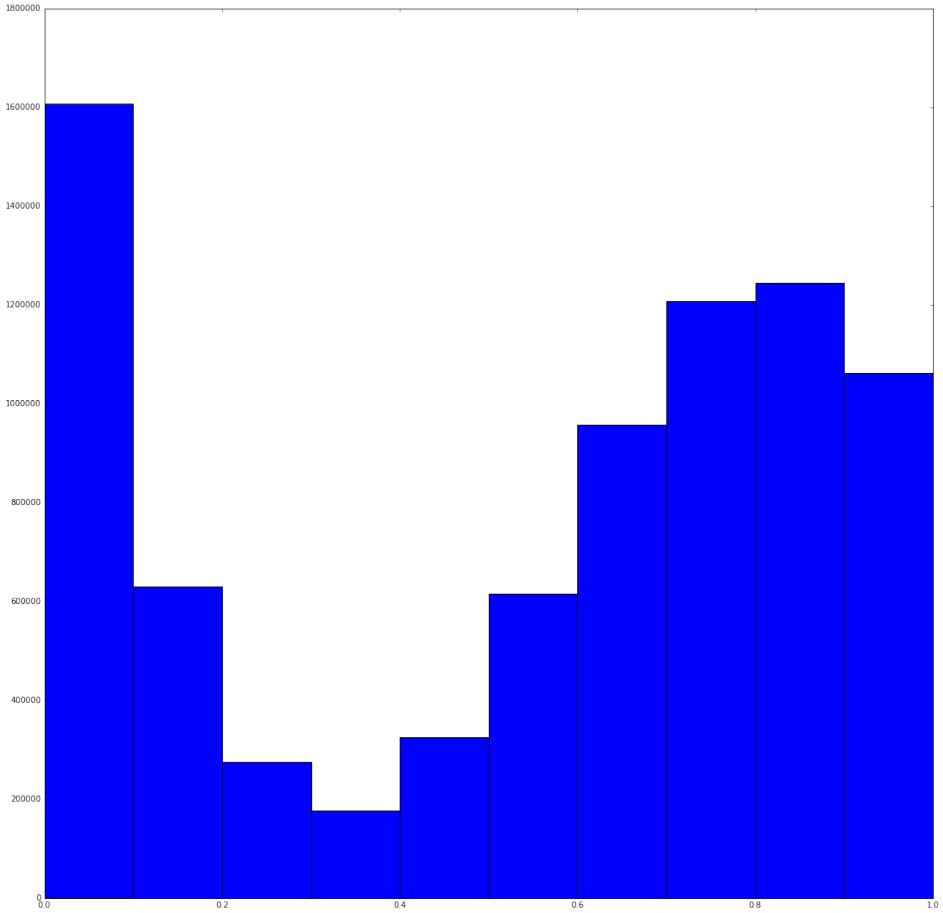


Figure 4.2: Bucket clustering algorithm, distribution of scores

4.2.3 Mean value

The mean value algorithm gives nearly a standard deviating result since the threshold in our algorithm is one standard deviation. However all result outside this standard deviation is set as 1.0. This distribution is easy to see in Figure 4.3.

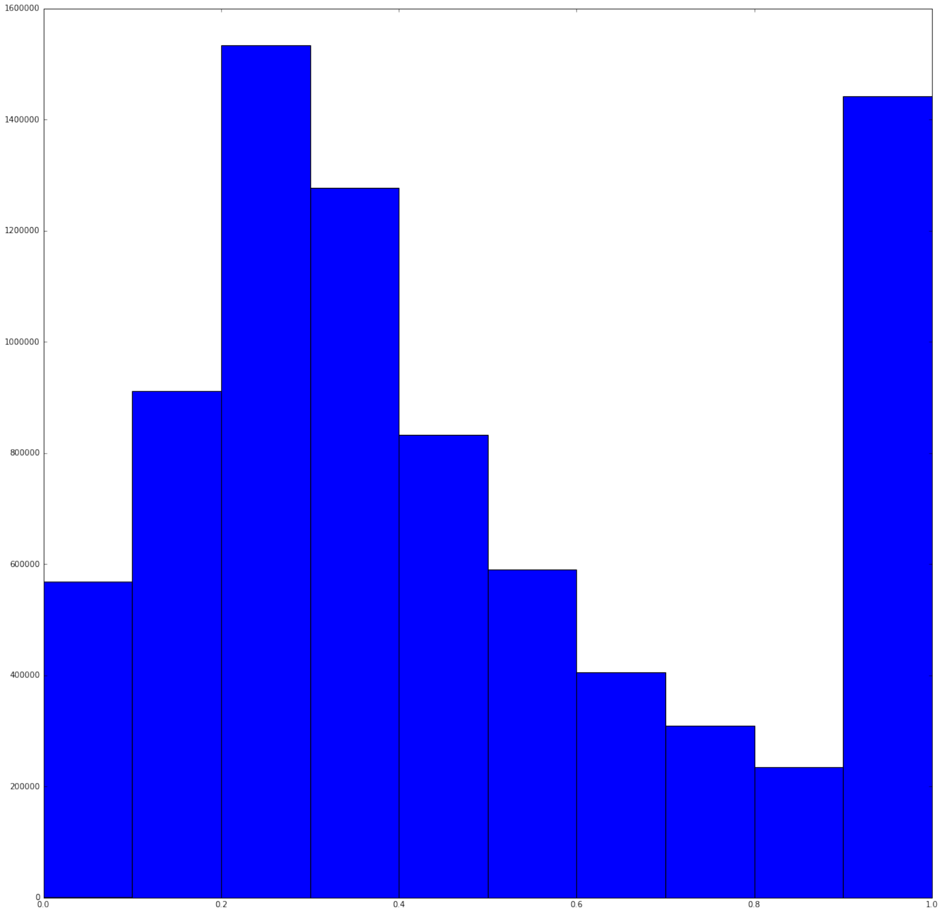


Figure 4.3: Mean algorithm, distribution of scores

4.2.4 Median value

The median value algorithm value changes more often than the median value algorithm. This is easy to see in the figure 4.4. The algorithm either gives a really low score or a really high score. One notable aspect of the Median-algorithm is that it shows three distinctive peaks and of them the extreme values are the most outstanding ones. This shows that in many cases the median value is to be expected. However, the high peak of 1.0-scores is an example of either the limitation mentioned in 3.3.5 or that the median value is insufficient in detecting outliers.

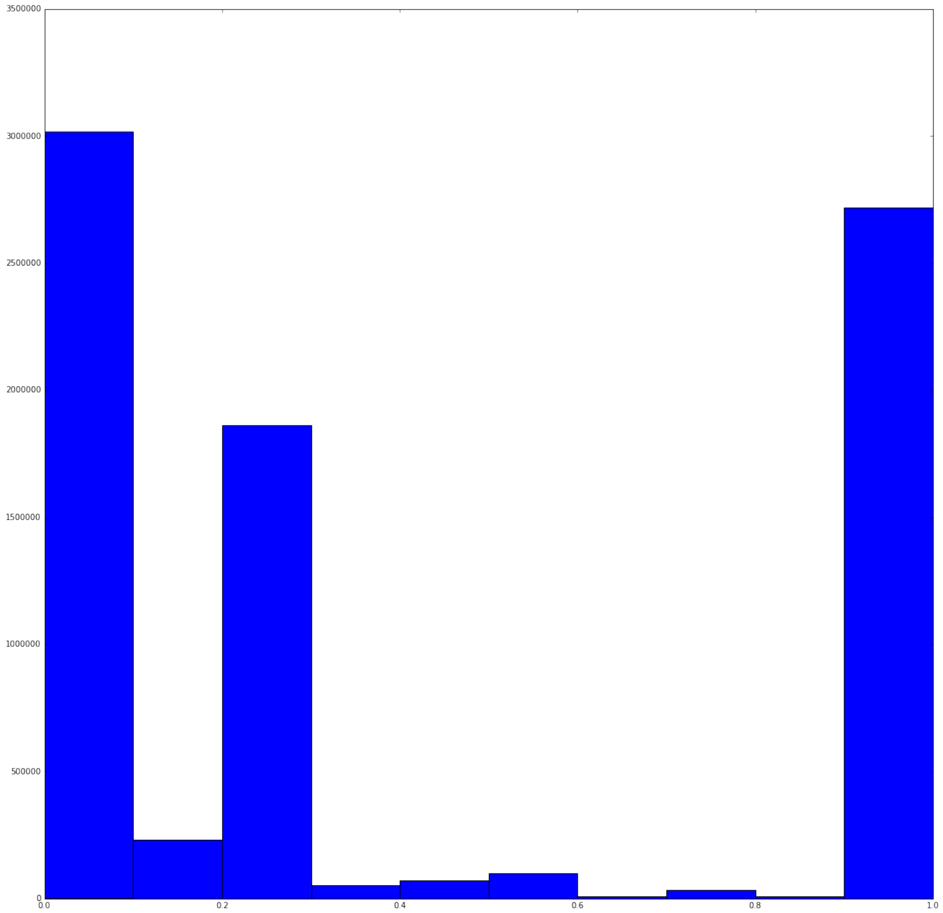


Figure 4.4: Median algorithm, distribution of scores

4.2.5 Normalization

If all the algorithms agrees about the score then the score is fixed to that value. However if they disagree with each other an algorithm normalizes the score to a fair value according to the equation described in 3.3.6. The result can be seen in Figure 4.5. A further analysis of the consensus scores are given in section 4.3

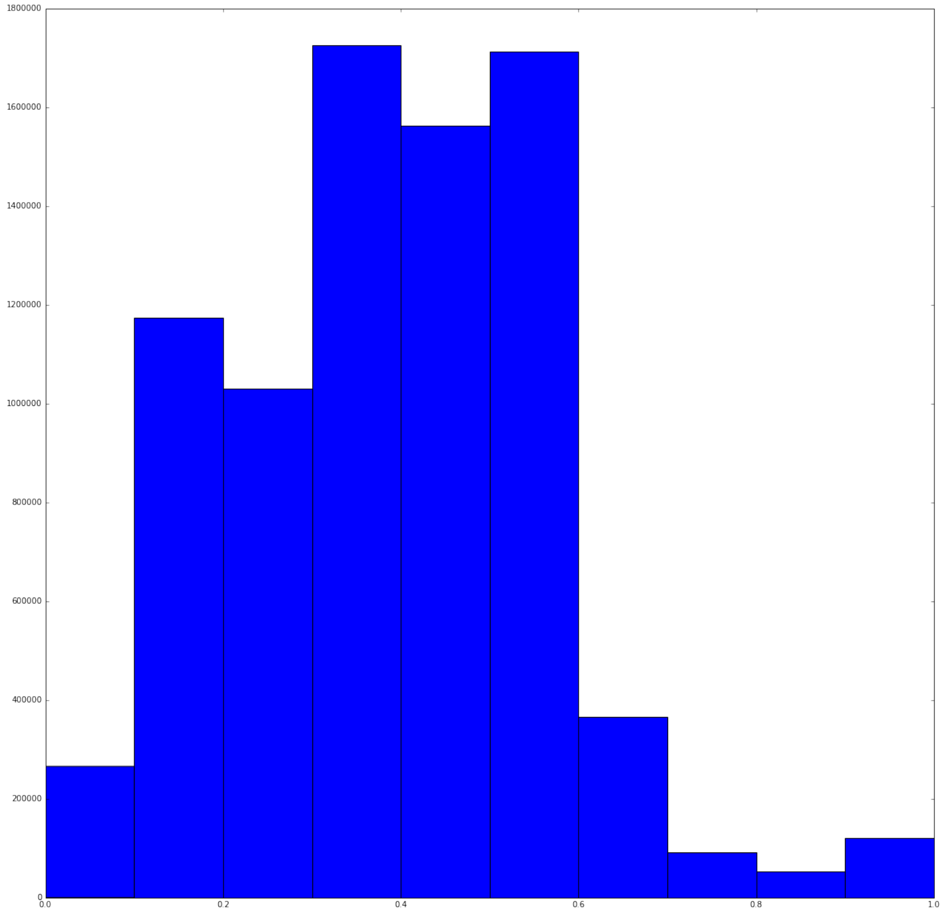


Figure 4.5: Normalization algorithm, distribution of scores

4.3 Consensus scores

In this section we present the ratio of consensus scores given for all input entries. When the consensus score is ≤ 0.1 a TN is assigned and for consensus scores ≥ 0.9 a TP is assigned. As can be seen in table 4.1 below there were relatively few entries that could be assigned a "definitive true" classification. The reason hat the term *definitive true* is in quote marks since its meaning is undefined in an unsupervised setting. This is why we choose call them "consensus scores" instead. As such the values used as thresholds are to be seen as guidelines or starting points which - when more knowledge about either the data or the correctness of the algorithms surfaces - could, and probably should, be changed for a more complete evaluation.

Please note that consensus scores are not by any means the only threshold for establishing true or false classifications. To further establish the true and false ratios a domain expert needs to analyze the data and the scores.

In the table below we show the ratio of true classifications for both metrics - attempts (A) and confirmations (C). Level 0 is the level of the keys matching Country and SPN (L0). Level 1 is the level of keys matching SPN and Type (L1). Please note that the results for L1 is not shown in the histograms above since the graphs would look somewhat similar. The percentage, within parentheses, presented in each cell is the approximate ratio of the classification occurrence for the entire input set of 8103481 entries. The scores that were not produced in consensus are marked as undetermined (U) in the table.

Table 4.1: Naive true classifications

	L0 - A	**L0 - C**	**L1 - A**	**L1 - C**
TP	91550 (1.13%)	83587 (1.03%)	395915 (4.89%)	374809 (4.63%)
TN	67637 (0.83%)	11941 (0.15%)	675312 (8.33%)	74722 (0.92%)
U	7944298 (98.04%)	8007957 (98.82%)	7032258 (86.78%)	7653954 (94.45%)

4.4 Comparison and Evaluations

4.4.1 Comparison with third party

The third party feeds all the data in to their system and produces collective anomaly detection based on previous data. In that way it is similar to our AD scheme, however the third party does not find anomalies based on seasonality over the week. They group the data only based on SPN, country and message type. This may be due to lack of knowledge of our data set.

As seen in Figure 4.6 the dark blue line is the actual value of each data point ordered by timestamp. The light blue area is the expected range the data points could adopt. If the data points adopts a higher value than the light blue area plus a threshold value during a significant amount of time, the system marks it anomalous. The anomaly is marked by a orange line. When an anomaly is found the system sends an alarm based on the operators specifications.

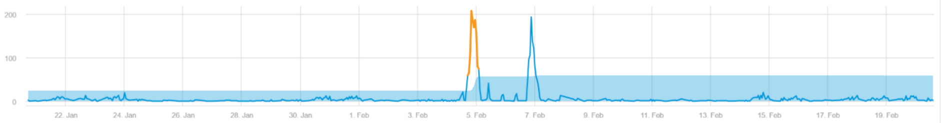


Figure 4.6: Visualization of the third party system (1)

With each incoming data point the expected value area is recalculated to get a new, better matching, expected value area. This recalculation can be seen in the right part of Figure 4.7 where the data points adopts very different values and the expected value area is matched to this differentiation.

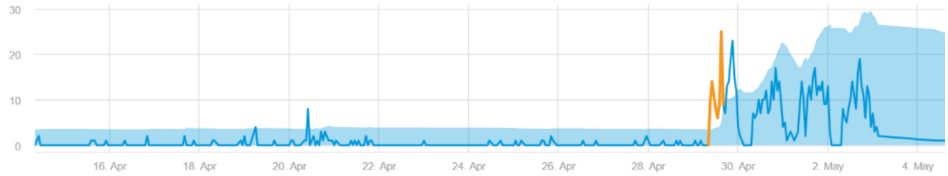


Figure 4.7: Visualization of the third party system (2)

4.4.2 Intrinsic score distribution

Table 4.2 summarizes the scores for each respective algorithm. The first column in the table represents certain intervals of scores. The other columns present the frequency of each score interval for each algorithm in our AD scheme. Thus each row shows the percentage of all output scores that each algorithm produced for a given interval. Conversely, each column outlines the score distribution for each algorithm but in a more aggregated way than as presented in section 4.2.

In table4.2 we outline 5 distinct intervals, where each interval represents an aggregated measure of certainty with regards to the true labeling of each algorithm. Please note that the last column, which outlines the result of our score committee (the normalization algorithm), is the official output and is of most interest for asserting trustworthiness of our scheme. Also, the percentages are rounded to one decimal (in one case two decimals).

The first interval, or row, exemplifies extremely high confidence in values being non-anomalous. As an example we can compare the Median algorithm, where 37.5% of all values were considered to be non-anomalous with high confidence. However, the score committee could only with high confidence label 3% as certain non-anomalies. The first and the last row of table 4.2 represent, what we have previously in this report named, the consensus score as described in section 4.3.

The middle rows of the table, row 2, 3 and 4, represent score distributions that are harder to confidently assign a truth label. Row 2 can be seen as representing scores with fair confidence, or somewhat unsure reliability, in being non-anomalies. The same distinction can be made for values falling in the interval given by row 4, but representing anomalies. Values that fall into row 3 are the unsure ones. The labels or classifications given to these values are ambivalent.

Table 4.2: Comparison between algorithms.

Score/Algorithm	**Poisson**	**Bucket**	**Mean**	**Median**	**Normalization**
$x \leq 0.1$	20%	20%	7%	37.2%	3%
$0.1 < x \leq 0.33$	13%	11.6%	35.5%	26.6%	33%
$0.33 < x < 0.67$	14.7%	21.2%	31.7%	2.7%	60%
$0.67 \geq x < 0.9$	12.8%	34%	8%	0.05%	2.5%
$x \geq 0.9$	39.5%	13.2%	17.8%	33.5%	1.5%

4.4.3 Evaluation

Comparing the "ambivalent distribution" for the score committee with the same for the other algorithms, in table 4.2, we can see that a great majority of scores are unreliable. One could argue that all scores below the value of 0.5 are given a negative classification, as non-anomalies, and the opposite for values above 0.5. The truthfulness of these classifications can however be debated considering the thresholds for establishing consensus scores, as detailed in section 4.3.

The threshold for establishing consensus scores could be moved such that values with scores less than or equal 0.33 and scores greater than 0.67 are subject to true classifications. However we refrain from moving the goalposts in this way as will be seen in section 4.3.

Given the current thresholds for consensus score we can estimate the absolute certain classification of our AD scheme, by adding the percentages from row 1 and 5 in table4.2, which is 4.5%. If the thresholds were changed as described in the previous paragraph the certain classification could be estimated to $(3 + 33 + 2.5 + 1.5)\% = 40\%$, where each value in the summation is taken from each cell except the 3rd in column 5 of table 4.2.

The reason for the score committee introducing ambivalence is based on the high variance in the score distribution for all algorithms pertaining any given value. Scores within the ambivalence range are a result of mean scores close to the endpoints (close to 0.0 or close to 1.0) together with a relatively high variance. Thus, one could interpret the 60% ambivalence, as presented in table 4.2, for the score committee output as based on 60% "disagreement" among the scheme's algorithms.

A high ratio of consensus scores and a low ratio of ambivalence scores would be optimal in the context of our evaluation. Our results show sub-optimal performance. This does however not necessarily lead to an automatic dismissal of the efficacy of our algorithm, as we note in section 2.1.4.

An unsupervised scheme has to be "put in action" where the amount of missed anomalies, as judged by another intelligent entity, must be measured in order to get a justified evaluation. The results of an unsupervised anomaly detection scheme have to be interpreted in order for it to make sense. Since the data used in our case study is real, not synthetically constructed, there is no verifiable truth to use as a benchmark [40]. In other word: The actual value of our AD scheme cannot be fully evaluated until all scores have been justified.

To conclude the evaluation of the algorithms and the final output from the score committee it is fair to say that the percentages shown in table 4.2 can easily be different. If another intelligent entity sets the threshold of the scores to be determined as certain anomalies or non-anomalies then a more thorough evaluation could be done.

Chapter 5

Discussion

5.1 Limitations

The framework of our solution is made to be adaptable to varying algorithms and data sets. However due to the scheme being general and easily adaptable the tuning of parameters needs to be done according to each new data set. Therefore our solution is fine tuned for CLX's data set and the progress described in 3.1 was guided by our preliminary results. The rest of this section outlines some of the more considerable limitations. This section can be seen as complementary to section 5.2 in that proposals for future improvements are implied.

5.1.1 Unintended learning

Once a entry in the input data has been scored it is added to the set of historical records. If this is done continuously for an extended period of time the old historical data will become redundant and the newer or more recent historical data more applicable in setting up AD-baselines. A malevolent user could fill, or train, the set of historical records so that extreme values are regarded as nominal or vice versa.

By pushing the median or the mean up or down, increasing the span of values existing or in any other way manipulating or engineering the inputted data, the security and trustworthiness of the system becomes exposed. But the AD framework in itself cannot prevent this. A user does not even have to be explicitly malevolent for this to happen.

A group of users that could produce surges, or bursts, of messages for an extended period of time producing an anomalous workload for an SPN. But once the AD framework is used to it, assuming that the surge lasts for such a long time that the traffic detected during the beginning of the surge is no longer marked as anomalous when detected by the framework, the equivalent drop of activity, or negative surge, might happen. This drop in traffic could lead to conditions that are very alike those before the surge. But again, the AD framework is expecting a high amount of traffic and will thus continue, until it expects it, to mark the traffic as anomalous.

The AD framework slowly evolves to accept a different set of values for its baselines. This is good in the cases where the user base naturally evolves a new behavior. But if surges happen, where the expected traffic flow peeks or drops, too often problems arise. This trend or cycle could repeat itself indefinitely and will result in one of two scenarios: Either the distribution of values for the matching historical records will be, somewhat, uni-modal which will mark more or less any value as "normal" or almost every single entry, matched against its historical records, will be marked as anomalous. Both cases destroy the credibility of the AD framework. To prevent this, a higher level of detail in discerning trends and patterns should be implemented. A suggestion to meet this end is presented in 5.2.2.

5.1.2 Monitoring meta flow

Our AD framework accepts any amount of entries inputted at any time. It calculates the scores for each entry independently of the amount of entries in a subset of the input, i.e., the entries that show up every 15 minutes. There is however one aspect that could mirror or shed light on how well the system is behaving and in a way be an additional metric used in AD. That is the total amount of entries for each XKC-combination each time an input-batch is fed to the AD-scheme. Our AD system does not count nor calculate the amount of entries into the system. This is surely an additional metric that could be investigated to reach even better scores in different data sets.

Additional metrics that could be used is presented in section 5.2.1. Another way to find anomalies in the number of entries during a specific time is to detect when there is no entry at all from a XKC-combination. This might not be called anomaly detection in the data since it is probably errors in the mechanisms or in the system. However in a system similar to this, a detection of such errors is as important as anomalies in the actual data.

5.1.3 Distinction for extreme outliers

Another limitation in our AD system is that our algorithms has no distinction for extreme outliers. I.e. if one of our algorithms gives the value 20 a score of 1.0, the values of 38 and 94 will also get a score of 1.0 as long as it is in the same XKC-combination. This means that these algorithms will categorize those values as equally anomalous. In some systems it could be relevant to grade the extreme outliers as well. I.e. if a patient comes to a hospital it is vital to decide how bad the injury is, but with a AD system like ours a fever and a car crash injury will get the same attention.

Through our research and knowledge of CLX's data set we decided that the extreme outliers are highly anomalous regardless of how extreme it is. Nevertheless grading of the extreme outliers could be highly relevant in future work.

5.1.4 Correlation of metrics

During development of our AD framework we assumed that the two metrics can be seen as independent of each other, but only in the context of our specific framework. The rationale behind this statement is the following: Even if the metrics, the number of attempts and confirmations, are completely correlated, the fact that we split each entry in univariate segments, as mentioned in sections 2.1.4 and 3.3, means that both metrics would be investigated parallel to each other. In the case where the correlation deviates, either one or both of the metric checks would have spotted this. The idea of dismissing the check for correlation is however still a limitation in our framework.

In section 5.2.1 we propose the idea of extrapolating additional metrics from the given data. In our case data there are some obvious candidates such as the ratio, co-variance and correlation of the metric values. These additional metrics could be used explicitly in the AD framework as baselines or they could be used in meta analysis, a subject touched upon in section 5.1.2. This is a viable future implementation because it could be seen in the case data that as the number of attempts increased, so did the confirmations, i.e., the confirmations and attempts "followed" each other if graphed.

5.1.5 Static thresholds

Some of our thresholds are static and fixed when launching the AD system. These thresholds are then used during the whole time the system is running regardless of seasonality or other variables. If a dynamic threshold determining the interval length, to use when comparing with historical data, would be tuned and changed depending on some values i.e. the seasonality of the data at a specific XKC-combination. Then the system could get even better results and be more flexible to different data sets, see section 5.2.2 for further explanation.

The interval of how old data the system should get from the historical data is static as well. With a detection of how long a trend have been represented, or other parameters regarding the data, a dynamic threshold of how much historical data that should be used in the calculation is possible. Such dynamic threshold could potentially give better results and in perhaps prevent unintended learning, since the system would know when to load a certain amount, or spans, of records. When fetching matching historical records using dynamic or changeable intervals, we are employing a more sophisticated way of discriminating entries, a subject which is further elaborated upon in section 5.2.2.

5.2 Future work

The above section detailed some of the limitations of our implementations as well as potential solutions to mitigate problems associated with those limitations. This section further tries to emphasize modifications or additions to improve performance.

5.2.1 Additional metrics

By defining the total volume for an entry as the sum of all metrics one creates an additional metric for the anomaly detection scheme. In our case data the volume can be seen as a third metric - implicitly derived from the current metrics. Furthermore, another additional metric can be defined as the ratio of the current metrics.

In our case data the ratio would be defined as the number of attempts divided by the number of confirmations for each entry. The two additional metrics are derived and are implicitly found from the original data. However they describe the data with a more generalized approach than the original metrics. For the total volume case, there is no distinction as to which metric contributed with what value to the sum. In the ratio case it is unknown what magnitude or absolute values that each respective metric value is that produce a specific ratio.

Either both or any of these two proposed additional metrics can be used in order to nuance the anomaly detection scheme. We assume that the new metrics would not add much clarity if applied as is on each entry or level separately in the data. However, if used on a meta-level to distinguish the change in volume and ratio for the entire system, i.e. the set of mechanism generating the entries, then the sum or ratio could describe if and how the data changes over time. The change in sums or ratios would indicate some sort of trend which in turn could be expected or unexpected. This information could then be used to either weigh the anomaly scores for each respective point or to indicate as to which entries of the reference data that should be discriminated against, as described in 5.2.2.

5.2.2 Dynamic discrimination

In our implementation we used a single condition for discriminating entries among the set of entries in the reference, or historical, data. The condition was that the only entries to be fetched were those where the difference in the time stamp between the current point and the reference points were within an interval. Furthermore, historical points aged more than 52 weeks before the current point were discarded immediately. However, our implementation can easily be modified to include more or different conditions. The examples described below can be seen as alternatives or modifications of the current implemented discrimination method and imply extended usage of clustering approaches [11], [40].

Trend and seasonality

Instead of only using entries that correspond to certain time stamp based on just the time that has passed one could instead divide the historical data into different catalogs. Each catalog would describe a certain behavior or trend of the data at a fixed or dynamical length of time. The discrimination process would be to see what trend or behavior that the current point is exhibiting, or being part of, and then pick the data points that have had a similar trend in the past, as reference values. This is conducive to using a clustering method for detecting collective anomalies

[58]. The discrimination would then not be to find previous entries with similar characteristics of the time stamp values but instead extract previous entries that are part of a similar pattern as the current entry.

One could use Seasonal Trend Decomposition (STL) which extracts a trend, seasonal and residual pattern of the time series data [59]. These patterns can be seen as additional metrics describing the data in a robust and versatile way [60]. Using STL on historical data one creates a forecast of what is to expected of the data in the future. STL can show if a value under scrutiny fits in with the expected patterns and how much. STL is a powerful tool which can be used not only in discerning which values or entries to discriminate against - it can in and of itself be used as an anomaly detection technique.

The methodology presented in the two aforementioned paragraphs can be employed in our implementation in two ways. Either by looking at the STL for a specific key-combination in an interval or it can be employed in a meta-level, a level which keeps track of how the volume or values, of the metrics, have evolved in recent time (the last x measurements). In other words, if the system or a specific key-value is showing a rapid or anomalous increase or decrease in data during the latest x measurements then one could discern that a burst is taking place. This burst could conform with the expected trend or it may not. In the case that it is not, the relative difference or ratio between the actual burst and the expected values could be used as a weighing factor in calculating a final anomaly score.

Tolerance for previous anomalies

As mentioned in 3.1 our approach starts of as a completely unlabeled anomaly detection scheme. However, after each iteration our scheme logs and labels each entry. In subsequent iterations it is most likely that some of the historical or reference data, that the algorithm will use in calculating anomaly scores contains labeled entries. In other words, our solution is converging towards a semi-supervised solution since all of the data that is being monitored, or is passing through our scheme, will have scores and the same data is appended to the historical records.

Please note that in the method presented in this report the scores produced by the scheme are not used as a selection criteria, meaning that we have created a basis for semi-supervised development which we have not utilized.

The frequency of labeled versus unlabeled data is in itself a value that can be tracked in order to see how fast the scheme is evolving. However, in the context of data discrimination one could use the score, the label, of those historical data entries that contain a label as a basis for discrimination.

As noted in section 3.3.2 outliers or anomalies in the reference data will give erroneous or misleading estimations and scores. One could easily implement a condition that, for example, makes sure that no entries that have an anomaly score $> x$, where x is either a constant or dynamically updated, will contribute to the set of values needed to calculate an anomaly score for a certain point. If that condition is implemented then we make sure that no anomalous data, as we have deemed it in the past, risks "contaminating" the calculations.

This is however not an advisable future step since the mechanism that generates the data could naturally be evolving in a way that produces values that will, or rather should, be deviating immensely (one might say anomalously) from historical values. If that is the case then all future values will receive a high scores since they will all be evaluated against an outdated baseline. In other words, this approach kills the potential for the data to naturally grow or decline while still maintaining a reliable anomaly detection. This increases the likelihood of false negatives. A way to circumvent this issue is by having the data generating mechanism evolve in a very slow fashion, a matter that is further discussed and elaborated upon in 5.1.1.

Another way to lower the risk of contamination while ignoring any potential labels is to operate under the assumption that all extreme values, low as high, are generally to be regarded as anomalous. The methodology is very alike to the one described in the paragraph above where entries with high scores are discriminated but instead of discriminating against high scores, the discrimination is based on extreme (actual metric) values. One could easily implement a discrimination scheme that disregards, from the relevant historical records, the $x\%$ most extreme values. This would mean that all entries whose metric value is on the ends of the spectrum of values will be ignored.

The spectrum ends are defined as the first and last $\frac{x}{2}\%$ entries - assuming that all values are sorted. This will, for the case of a natural evolving process, induce high scores, or false positives, for the initial entries but as the spectrum of values increases, the discrimination of the extreme values (which, for the "new" process are to be regarded as ordinary) lowers.

After a certain amount of time, given that previously extreme values are now common, the endpoints of the spectrum will contain a subset of the common values. As the super set of common values increases, the impact of the common values within the spectrum endpoint will diminish. In time, the endpoints will cover "actual" extremes.

Assuming that the anomaly detection scheme has access to not only the actual historical data but also its anomaly score one could easily modify our implementation to either exclude points or weigh the values for entries which have, in previous iterations, been marked as anomalous.

5.2.3 Distribution fit for the general case

As mentioned in 3.1 our solution was centered around the assumption that the mechanism generating the data values was a Poisson process. Accordingly, our solution can only reliably scale up or down to data that meets that assumption. In order to maintain the ability to apply our concept as a solution in applications where the data does not exhibit the same distribution, the setup of the anomaly detection scheme must be modified. Modifications should be made to accommodate for the possible processes generating the data under scrutiny. This can be done as a setup-step of the anomaly detection scheme where the distribution of the data is explicitly stated beforehand, either by manual or automatic inspection, so that when the scheme is executing it is aware of, or knows, which probability mass functions to use.

To further generalize the approach - with the risk-benefit of adding more complexity and computations for more precise scores - the distribution can be confirmed in real-time based on the distribution of the current, actual or requested reference data. This process can be done through sophisticated LOL clustering[11] or using density estimation[61].

5.3 Ethics

The data as it is presented in section 2.1.3 is anonymous such that it provides no insight in to what country, SPN or type that is actually being monitored. For legal reasons the exact information of the keys or identifiers are scrambled or masked - given "code words". This is to protected the integrity of CLX's customers. Furthermore, the case data contains no personal information, about any individual user of an application, i.e, type as per the description in section 2.1.3, or SPN. The only identifiable entities that are being tracked are the SPNs' and the types and no distinction is made between any user within the operational domain of the SPN. As such, the case data as it was provided for us cannot be used to track or monitor individual users; it only tracks the statistics, the traffic flow, of the SPN's that has willingly provided this data.

AD systems could be used to monitor individual users. As an example, our case data could very well be fitted with yet another key or identifier which represents the credentials of a single unique user. To protect the integrity of individual users we do not recommended adding personal identifiers to the case data. When monitoring a system on a macro level, the addition of a personal identifier might make monitoring harder or unnecessarily complex because the benefit margin would be too small. If a monitoring system were deployed to catch anomalies on the micro, or user based, level this additional identifier might become needed.

AD is widely employed in big or cloud data applications, as well as fraud and intrusion detection systems, where there exists a gray zone between what is personal information and meta-data [5], [14], [44], [49]. No matter if the intent of deploying an AD system is malicious or not, care must be taken when deducing what an anomaly is and if any individuals privacy is violated. An individual might not like being "marked as anomalous" or being monitored to begin with [63].

5.4 Conclusion

Unsupervised AD is a concept which is filled with ambiguities and context specific applications. Designing autonomous and reliable frameworks that minimize false classifications as well as maximizing the amount of true classifications is a hard task. Static thresholds or assumptions, such as those presented in this report, provide a starting ground or an initial phase of anomaly detection. Our implementation has focused on simple statistical and clustering approaches which are by no means sophisticated enough to stand on their own in order to produce meaningful results.

Our ensemble produces results which at first glance imply a sense of correctness. But since our case study was performed in an unsupervised setting there is, at the time of writing, no true justification for asserting whether our solution is successful. The methodologies and solutions presented in this report are by no means accomplished or adequate enough to fully establish a completely functional AD framework. For future researchers and developers who hope to create an AD framework or scheme that complies with the aims of us we suggest to investigate further in the domain of machine learning.

The purpose of this work was to monitor a system from a technical view point in order to minimize disturbances in traffic flow (as described in section 1.3.1) - not to profile the activities of either users or networks. Since our case study's domain of operation was on the level of *countries, networks and types* we had fewer restraints, or moral obligations, to provide a flawless classification mechanism. We mention this because, if our work is to be applied in the domain of health care, medicine or corresponding areas, caution must be taken as to how reliably our solution produces results in a domain where there is no room, or margin, for errors. A case of unintended learning, described in section 5.1.1, could further lessen the trustworthiness and safety of our implementation.

References

[1] *Number of Internet Users (2016) - Internet Live Stats, 2013* `http://www.internetlivestats.com/internet-users/` Accessed 13th Sep. 2016

[2] *World Internet Users Statistics - Internet World Stats, 2002* `http://www.internetworldstats.com/stats.htm` Accessed 13th Sep. 2016

[3] *Internet Society Global Internet Report 2015* `https://www.internetsociety.org/globalinternetreport/` Accessed 6th Nov. 2016

[4] Hilbert, Martin, and Priscila López, *The world's technological capacity to store, communicate, and compute information*, science 332.6025 (2011): 60-65

[5] Chandola, Varun, Arindam Banerjee, and Vipin Kumar, *Anomaly detection: A survey*, ACM computing surveys (CSUR) 41.3 (2009): 15

[6] Hodge, Victoria J., and Jim Austin. *A survey of outlier detection methodologies* Artificial intelligence review 22.2 (2004): 85-126.

[7] Sun, Jimeng, et al. *Neighborhood formation and anomaly detection in bipartite graphs* Fifth IEEE International Conference on Data Mining (ICDM'05). IEEE, 2005.

[8] Patcha, Animesh, and Jung-Min Park, *An overview of anomaly detection techniques: Existing solutions and latest technological trends*, Computer networks 51.12 (2007): 3448-3470

[9] *Definition of Anomaly by Merriam-Webster* `http://www.merriam-webster.com/dictionary/anomaly` Accessed 10th Oct. 2016

[10] Keogh, Eamonn, and Jessica Lin, *Clustering of time-series subsequences is meaningless: implications for previous and future research*, Knowledge and information systems 8.2 (2005): 154-177

[11] Liao, T. Warren. *Clustering of time series data—a survey* Pattern recognition 38.11 (2005): 1857-1874.

[12] Tsai, Chih-Fong, et al. *Intrusion detection by machine learning: A review* Expert Systems with Applications 36.10 (2009): 11994-12000.

[13] Bhattacharyya, Dhruba Kumar, and Jugal Kumar Kalita. *Network anomaly detection: A machine learning perspective* CRC Press, 2013.

[14] Pimentel, Marco AF, et al. *A review of novelty detection* Signal Processing 99 (2014): 215-249.

[15] Gaikwad, Prathamesh, et al. *Anomaly detection for scientific workflow applications on networked clouds.* High Performance Computing & Simulation (HPCS), 2016 International Conference on. IEEE, 2016.

[16] Bloedorn, Eric et al. *Data mining for network intrusion detection: How to get started*, Aug. 2001

[17] Fanaee-T, Hadi, and Joao Gama. *Event labeling combining ensemble detectors and background knowledge* Progress in Artificial Intelligence 2.2-3 (2014): 113-127.

[18] Kearns, Michael J., Robert E. Schapire, and Linda M. Sellie. *Toward efficient agnostic learning* Machine Learning 17.2-3 (1994): 115-141.

[19] Hyndman, Rob J et al. *Optimal combination forecasts for hierarchical time series*, Computational Statistics & Data Analysis 55.9 (2011): 2579-2589

[20] Salvador, Chan, Brodie *Learning States and Rules for Time Series Anomaly Detection*, 2004

[21] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. *Anomaly detection for discrete sequences: A survey.* IEEE Transactions on Knowledge and Data Engineering 24.5 (2012): 823-839.

[22] Rokach, Lior. *Ensemble-based classifiers.* Artificial Intelligence Review 33.1-2 (2010): 1-39.

[23] Moya-Gómez, Borja, et al. *Dynamic accessibility using Big Data: The role of the changing conditions of network congestion and destination attractiveness* arXiv preprint arXiv:1610.06450 (2016).

[24] Gaikwad, Prathamesh, et al *Anomaly detection for scientific workflow applications on networked clouds* High Performance Computing & Simulation (HPCS), 2016 International Conference on. IEEE, 2016.

[25] Pencina, Michael J, Ralph B D'Agostino, and Ramachandran S Vasan *Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond*, Statistics in medicine 27.2 (2008): 157-172

[26] Leape, Lucian L. *A systems analysis approach to medical error* Journal of evaluation in clinical practice 3.3 (1997): 213-222.

[27] Department of Electrical and Information Technology at Lund University, Faculty of Engineering `http://www.eit.lth.se/` Accessed 14th Dec. 2016

[28] About CLX Communications, 2016, CLX Communications `https://www.clxcommunications.com/about/` Accessed 20th Oct. 2016

[29] Zhu, Xiaojin. *Semi-supervised learning literature survey* (2005).

[30] Chan, Philip K., and Matthew V. Mahoney. *Modeling multiple time series for anomaly detection* 5th IEEE International Conference on Data Mining (ICDM'05). IEEE, 2005.

[31] Song, Xiuyao, et al. *Conditional anomaly detection* IEEE Transactions on Knowledge and Data Engineering 19.5 (2007): 631-645.

[32] The IEEE and The Open Group *POSIX time* The Open Group Base Specifications Issue 7 IEEE Std 1003.1-2008, 2016 Edition `http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html` Accessed 20th Nov. 2016

[33] Blum, Avrim, and Tom Mitchell. *Combining labeled and unlabeled data with co-training* Proceedings of the eleventh annual conference on Computational learning theory. ACM, 1998.

[34] Sekar, R et al. *Specification-based anomaly detection: a new approach for detecting network intrusions*, Proceedings of the 9th ACM conference on Computer and communications security 18 Nov. 2002: 265-274

[35] Robinson, Jason et al *SHARD: a framework for sequential, hierarchical anomaly ranking and detection*, Pacific-Asia Conference on Knowledge Discovery and Data Mining 29 May. 2012: 243-255

[36] Hong, Chi-Yao et al. *Tiresias: Online anomaly detection for hierarchical operational network data*, Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on 18 Jun. 2012: 173-182

[37] Breunig, Kriegel, Ng, Sander *LOF: Identifying Density-Based Local Outliers*, 2000

[38] Powers, David Martin. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation* (2011).

[39] Sokolova, Marina, Nathalie Japkowicz, and Stan Szpakowicz *Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation*, Australasian Joint Conference on Artificial Intelligence 4 Dec. 2006: 1015-1021

[40] Berk, Richard, and John M. MacDonald. *Overdispersion and Poisson regression* Journal of Quantitative Criminology 24.3 (2008): 269-284.

[41] *Machine Learning* taught by Andrew Ng at Stanford University `https://www.coursera.org/learn/machine-learning`, 2 Nov

[42] Leys, Christophe, et al. *Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median.* Journal of Experimental Social Psychology 49.4 (2013): 764-766.

[43] Rousseeuw, Peter J., and Christophe Croux. *Alternatives to the median absolute deviation.* Journal of the American Statistical association 88.424 (1993): 1273-1283.

[44] Sabahi, Farzad, and Ali Movaghar, *Intrusion detection: A survey*, 2008 Third International Conference on Systems and Networks Communications 26 Oct. 2008: 23-26

[45] Ciocarlie, Gabriela F et al. *Detecting anomalies in cellular networks using an ensemble method*, Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013) 14 Oct. 2013: 171-174

[46] Ciocarlie, Fisher, Wendy, Tracy Camp, and Valeria Krzhizhanovskaya *Crack Detection in Earth Dam and Levee Passive Seismic Data Using Support Vector Machines*, Procedia Computer Science 80 (2016): 577-586

[47] Estan, Cristian, Stefan Savage, and George Varghese *Automatically inferring patterns of resource consumption in network traffic*, Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications 25 Aug. 2003: 137-148

[48] Zhang, Yin et al. *Online identification of hierarchical heavy hitters: algorithms, evaluation, and applications*, Proceedings of the 4th ACM SIGCOMM conference on Internet measurement 25 Oct. 2004: 101-114

[49] Vasilios A.Sirisand Fotiini Papagalou *Application of anomaly detection algortihms for detecting SYN flooding attacks*, 2006

[50] Portnoy Leonid *Intrusion detection with unlabeled data using clustering*, 2001

[51] Jason R. Chen *Making Subsequence Time Series Clustering Meaningful*, 2005

[52] Zhou, Yan, and Sally Goldman. *Democratic co-learning* Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on. IEEE, 2004.

[53] Georgiou, Harris V., and Michael E. Mavroforakis. *A game-theoretic framework for classifier ensembles using weighted majority voting with local accuracy estimates* arXiv preprint arXiv:1302.0540 (2013).

[54] Jupyter Notebook `http://jupyter.org/` Accessed 20th Sep. 2016

[55] Hunter, J. D. *Matplotlib: A 2D graphics environment* Computing In Science & Engineering 9.3 2007

[56] Anodot `http://www.anodot.com/product/`, Accessed 10th Oct. 2016

[57] Simhon, Ben, et al. *System and Method for Transforming Observed Metrics into Detected and Scored Anomalies.* U.S. Patent No. 20,160,147,583. 26 May 2016.

[58] *Anomaly Detection using K-Means Clustering* `https://anomaly.io/anomaly-detection-clustering/` Accessed 17th Oct. 2016

[59] Cleveland, Robert B., et al. *A seasonal-trend decomposition procedure based on loess.* Journal of Official Statistics 6.1 (1990): 3-73.

[60] Rob J Hyndman and George Athanasopoulos *Forecasting: principles and practice* May 2012 `https://www.otexts.org/fpp/6/1` Accessed 17th Oct. 2016

[61] Silverman, Bernard W. *Density estimation for statistics and data analysis* Vol. 26. CRC press, 1986.

[62] Surowiecki, James. *The wisdom of crowds* Anchor, 2005.

[63] Sicari, Sabrina, et al. *Security, privacy and trust in Internet of Things: The road ahead* Computer Networks 76 (2015): 146-164.

Printed by Tryckeriet i E-huset, Lund 2017

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2017-567

http://www.eit.lth.se