Master thesis:
Detecting deviant behaviour in information systems by using outlier detection on logs

Author
Roelof Jan Mathijs de Groot
Program: Business Information Technology MSC
Institute: University of Twente
Student number: s1392999
Email: r.j.m.degroot@student.utwente.nl

Graduation committee
Maria E. Iacob

Department: Industrial engineering and Business information systems
Institute: University of Twente
Email: m.e.iacob@utwente.nl

Maya Daneva

Department: Service, cybersecurity and safety
Institute: University of Twente
Email: m.daneva@utwente.nl

Wout Slakhorst

Department: Nedap Healthcare
Team: Cupido
Email: wout.slakhorst@nedap.com

Joep Peeters

Department: Nedap Healthcare
Team: Privacy & security team:Shield
Email: joep.peeters@nedap.com

Abstract

Issues of IT-security remain prevalent in the world. While companies are starting to take information security more seriously, there is still a lot of potential left untapped that could support improvement of information security. This master thesis proposes a methodology that taps into a readily available source of data, namely system-logs. Logs can be generated by a system whenever an user performs an action. These logs might yield interesting discoveries when analysed with techniques from certain field such as data-mining and machine-learning. This thesis proposes a methodology that guides the creation of a system that analyses logs to discover deviant behaviour of users. This system will analyse the logs in two steps. First, clustering algorithms will be used to group users with similar behaviour together. The second step consists of using these groups of similar as input for outlier-detection algorithms. The outliers that are discovered can then be evaluated and used to improve information security. The methodology has been evaluated by using a prototype. This prototype has been used to analyse data supplied by Nedap Healthcare. This data was generated by caretakers while performing their jobs during the months Augusts, September and October of 2017. As such this data is representative of actual and relevant real-world circumstances. This means that problems that arise from using real-world data have been encountered during the creation of this methodology. The organizational relevance of the methodology should have been improved by using this data. The results of the prototype show that grouping users based on logs is feasible. However, the outliers detected by the outlier-detection algorithms were found the be of limited use to improve information security. It has been concluded that while the proposed methodology could be a step in the right detection, more research is needed in finding and enhancing the process to be truly useful in an organizational context. Presently the methodology shows a practical approach to analysing logs which can be enhanced easily.

Table of Contents

1 Introduction

In today's world one will have difficulty finding a company that is not dependent on information technology(IT). Communication, finance and management are all segments within an organization that are permeated by IT. However while IT has a place in everyday live, IT-security has not(Witty, Allan, Enck, & Wagner, 2003). Neglecting information security is very dangerous for companies as revealed information can result in loss of competitive advantage, customer trust or even result in law suits(Campbell, Gordon, Loeb, & Zhou, 2003). A good understanding of a company IT-systems is necessary to improve IT-security. However most companies do not build their own software products and rely on specialist companies to supply them with information systems. These software suppliers often use software as a service(Saas) as their business model. This means software is provided to the companies under license as a black-box, these products do not require their customers the understand the software inner workings. These software products often require users to log in and identify themselves to be able to use it. This is the first part of identity and access management(IAM), a very important part of information security for any system. The second part of IAM is access-management, this involves which user can access which part of the system. Access-management is often not done by the software suppliers themselves, but left to the companies. While this is an important and very difficult task, companies are still struggling with it(Gunter, Liebovitz, & Malin, 2011). Dhillon states that a majority of computer security breaches is done by internal employees(Dhillon, 2001). These breaches could be alleviated by proper access management. Security managers of companies that are responsible for access-management could be supported in their task with information that is automatically collected by software suppliers. Software companies receive a lot of data about their clients and often store this data as part of their services. Unfortunately this data is too large and unwieldy to be analysed manually. In order to make use of this data it must be transformed and analysed to ensure only interesting information is shown to the security manager. The field of data-mining already has answers to this problem by proposing techniques and algorithms to analyse large amounts of data. An autonomous system that empoys data-mining techniques could analyse the data for anomalies that are then represented to a security manager. These anomalies could then be investigated and used to improve information security.

It has been mentioned that the data will serve as input for such a system is already collected by most companies. Audit-trails of data-transactions is such an example, since most companies are required by law to collect such data. Finding an effective way to processes and store this data has been given priority over analysing it due to the large amount of data that is generated each day. Storage methods that are often used to store this data are of a type that makes it hard to analyse the data efficiently, an example of this is magnetic tape. In cases were companies are actively analysing the data, priority is given to creation of business intelligence that supports in getting a competitive advantage over competitors. When taking a security perspective audits are used after the fact. After an incident has happened, the data is used to gain more information on the impact of the incident. Using the audits to improve the internal security of the information system has been mentioned in theory, but few concrete ways of how companies are doing this are given. The internal security of an information system is important, however, especially when working with sensitive personal data such as in the healthcare industry. The protection of personal data should give incentive to companies to ask their software suppliers to create a system as this thesis proposes. Financial considerations are always a factor and standardizing and storing data could prove expensive. As a proof of concept less expansive data should be used.

A relatively novel approach would be to make use of an inherent characteristic of SAAS products.

Saas products are often build using an software-oriented architecture(SOA). A common principle of Saas is that of standard service contract, which entails that communications between services are standardized. This means that applications communicate to each other in a similar manner, they all speak the same language as it were. Since all inter-application communication is done in the same language, the logs of these messages will also be standardized. These logs could make for a good source of data, it should be noted however that only part of the actions a user makes is covered in these logs since only inter application-communication is logged. While this is a disadvantage the upset is that there is less of it which makes it easier to store. These logs still provide a lot of insight in actions made by users on the system, since every time a users requires information not present on his current application a message is logged. Another advantage is that the standard format of the logs makes the process of preparing the data to allow for analysing a lot easier.
Hence, using logs of all inter-application communication generated by Saas products allows for data that is easier to store and analyse because less preparation is required because of the standard format. This would reduce the need for software suppliers to make investments, which would make them more inclined to offer the system this thesis proposes as a service.
This thesis proposes a methodology that allows for the creation of a system that detects anomalies in data collected by companies. These anomalies could provide security managers with clues to improve information security. Logs of inter-application communication are used as input as a alternative to audit-trails. Including this business perspective makes this research interesting for businesses and researchers alike.

1.1 The problem statement

Companies could improve their information security with the logs their systems collect each day. Analysing these logs manually would be a difficult and time-consuming process since a lot of data is generated each day. An automated system should be made that analyses the data and provides only interesting occurrences in the data to be evaluated further. Techniques from fields such as data-mining and artificial intelligence could potentially be used to create such a system.
However when researching this topic it was found that there is limited knowledge available on how to approach the creation of such a system. A methodology that is a guide towards a practical implementation of a system that makes use of artificial intelligence to analyse logs should be readily available. This thesis proposes such a methodology. The methodology will focus on detecting deviant behaviour of users. These users are employees of companies that use the the system while performing their job. This methodology would be a step towards a more secure world that allow for companies to make use of the information available to them.

1.2 Research questions

To create the methodology proposed in the problem statement the following research question has been formulated.
How could one approach analysing system-logs to support information security using artificial-intelligence?
Several sub-questions are used to answer this question:

1. How can system-logs be used to support information security?
2. What are the data requirements that should be met by the logs?
3. How should an architecture that analyses logs be designed?
4. How does the methodology perform on data from the healthcare sector?

1.3 Research methodology

The structure of this be based on seven of the eight components that make up the anatomy of design theory described by (Jones & Gregor, 2007). These seven steps will be followed in a sequential

manner.

1. *Purpose and scope*
    - The goal of the system and the context it should be applied in should be clear. The scope and the boundaries should also be defined in conjunction. Chapter 1 has been dedicated to this component.
2. *Constructs*
    - The representation of the entities of interest to the theory. Chapter 2 will be dedicated to exploring the literature and subchapter 3.1 will elaborate on the terminology used during the thesis. Research question 1 can be mapped under this component.
3. *Principle of form and function*
    - A blueprint of the methodology is described in chapter 3. This is an abstract description of the architecture of the methodology.
4. *Artefact mutability*
    - The changes in state of the artefact that is anticipated by the theory. The answer to research question 2 describes this component. This is elaborated on chapter 3 and especially in subchapter 3.2.1.
5. *Expository instantiation*
    - A prototype has been developed that will be used to represent the methodology as an expository device and for the uses of testing. Chapter 4 is dedicated to the prototype.
6. *Testable proposition*
    - Truth statements will be made about the artefact. This will be done while evaluating the prototype in chapter 5. This component is represented as research question 4.
7. Justificatory knowledge
    - The underlying knowledge about the theory. Chapter 6 will conclude by answering what can be said about how the artefact functions using the information reported on in the previous chapter.

1.4 Scope

There are many different approaches to information security. To make the project feasible in the limited amount of time available the scope has to be defined.

- The focus will be put on data-mining approaches.
- Only access-logs received by the ESB of Nedap will be used.
- Only a maximum of one month of access-logs will be analysed.
    - While data from a year could potentially yield better results, this is not feasible within the limitations of the project.
- Solely free data-mining tools/libraries will be used.

1.5 Structure

The rest of this thesis will be structured as follows; A literature study will be presented in chapter 2, then a solution design will be given in chapter 3. Chapter 4 is dedicated to describing the prototype. The results generated by the prototype will be evaluated in Chapter 5. Chapter 6 will concluded by evaluating the methodology as a whole and discuss the limitations of the research. The chapter will finish by making recommendations for further research.

2 Literature

A common question in research is “What has already been written about this topic”. For the purpose of this thesis, the literature has been consulted on the topics of data mining and outlier detection. Subchapter 2.1 has been written as part of a separate course on the university named research topics. It forms the literary basis for choosing the approach for the methodology. In chapter 2.2 a short elaboration on Nedap can be found. This can be used to gain insight in the data that was used when testing the prototype.

2.1 Literature survey:Outlier-detection approaches

Data-mining is a field of research that receives more and more attention, one aspect of data-mining is outlier-detection. Outliers are sometimes regarded as negative disruptive instances that should be removed(Sreevidya & others, 2014). Others actively seek to find outliers due to their usefullness in certain fields, such as fraud and intrusion detection(Chandola, Banerjee, & Kumar, 2007). The detection of outliers or anomalies has a lot of potential in information security. Suppliers of software as a service(Saas) products are in group that could benefit from using data-mining and outlier-detection techniques. Saas-suppliers have access to a lot of data each day, since they can collect access-logs of their customers, some suppliers even collect audits for their customers as a service. These could provide a wealth of information, that could be used to improve the security of their products. This research aims to explore what approaches and techniques are available, to detect outliers in these logs. This is the first step towards tapping into these possibilities and developing a methodology to use access logs to support information security. Research in the field of outlier-detection has been consulted regarding this topic. This literature survey starts with conveying some concepts that will be used throughout the literature survey. Subsequently a overview will be given on the current outlier-detection approaches. This overview will be used to address which algorithms are used. With the information accumulated in the previous two paragraphs the approaches which are applicable for the analysing of access-logs will be selected. This selection will be evaluated and the approaches with the most potential will be selected for further testing in a prototype. This results in four knowledge questions.

- Knowledge questions:
  - What approaches are currently used in outlier-detection according to published literature?
  - What are the algorithms used in outlier-detection?
  - What approaches are applicable for analysing access-logs?
  - What algorithms are most applicable for our research-problem?

2.1.1 Research goal & questions

This literature survey serves to inform the creation process of a methodology that uses access-logs to improve information security. Systems that are used by employees of organizations will generate a large amount of access-logs everyday. Extracting or “mining” knowledge from large amount of data is called data-mining(Han & Kamber, 2010, p. 5). Using those patterns is referred to as outlier-detection. To develop a outlier-detection approach it is necessary to have an insight in what research has been done in this field. The goal of this research is to provide an overview of techniques currently used in outlier-detection. To accomplish this goal we pose two knowledge-questions:

1. What approaches are currently used in outlier-detection according to published literature?
2. What algorithms are currently used in outlier-detection approaches according to published literature?

2.1.2 Concepts

To talk in a sensible matter about this topic some concepts need to be explained. These concepts

will be used to talk about the papers found during the literature survey. First a broad definition of data-mining and machine-learning is given, since these fields have a lot of overlap they have different meanings in different field. For this paper these fields are defined as follows. Data-mining is about discovering patterns in data and machine-learning is about being able to create assumptions from the found patterns. One could say that a data-mining approach uses a machine learning algorithm. With this distinction made, the main topics of this paragraph can now be introduced, namely the types of data and supervision. These two concepts are closely linked and will be explained in the following paragraph.

2.1.2.1 Data and Supervision

When doing machine-learning one tries to create assumptions or rules about the data, it is beneficial for understanding the data if it can be subjected to some form of order. By doing this, predictions can be made about new instances of the data. How one does this depends on the type of data available and what one wants to predict. To predict something one needs data to make predictions about. Data consists out of variables, it is important that all instances of a variable adhere to a uniform format. A lot of preprocessing in data-mining is done to ensure this uniformity of data. For example: temperature could be noted either in degrees Celsius or in degree Fahrenheit not a combination of both. A very important question in data-mining is if one has data available that already contains the value that one wants to predict. Data that contains the value one wants to predict the data is called labelled. For example:
Say one wants to predict if people will work outside, to predict this data the variables outlook and windiness are available. An unlabelled data-set would look like Table 1:

Weather	**Windy**
Sunny	Yes
Rainy	No
Sunny	No

Table 1: Unlabelled data

There might be records available of when people worked outside in the past. These records are historical data that contain an extra variable, this variable would contain answer to the question if people actually worked outside that day. The predicted variable is now known, which means it can be exactly known in which weather situations people worked. A labelled data-set can be made that would look like Table 2:

Weather	**windy**	**Work outside**
Sunny	Yes	No
Rainy	No	No
Sunny	No	Yes

Table 2: Labelled data

Naturally labelled data can only be made if one knows what one wants to predict. When machine-learning is done using label data, this is called supervised machine-learning. In supervised machine-learning one trains a classifier with labelled data. Once the classifier has received enough training it can classify or predict new instances of the data. This new data is then naturally unlabelled. Machine-learning without labelled data is called unsupervised machine-learning. The distinction between supervised and unsupervised is very important. Having labelled data available that can be used as trainings data is very useful. Besides supervised an unsupervised learning, semi-supervised machine-learning also exists. In this case incomplete labelled data is used as input. In the previous

example we wanted to predict if people were going to work outside. One could only have historical data from when people actually worked outside and no data of the days nobody worked outside. This data could look like Table 3:

Weather	**windy**	**Work outside**
Sunny	Yes	Yes
Sunny	No	Yes
Sunny	No	Yes

Table 3: Incomplete labelled data

Since there are no instances of data when people didn't work outside this data is not really complete. Still one can use it for machine learning. This semi-supervised method would then create a norm and would predict everything that would diverge from the norm as not a work-outside day.

To summarize if one has labelled data one can make a training-set. Using a training-set one can train a classifier this is called supervised machine-learning. If there is no labelled data available one has to do unsupervised machine learning. In the case of labelled data that is incomplete one can still train a classifier. This is called semi-supervised learning.

2.1.3 Literature search approach

The literature search was conducted with the use of keywords and the database of Scopus and Google Scholar were used to find the literature. First a preliminary search was carried out were it was discovered that some areas of outlier-detection that received a lot of attention. These areas were fraud detection, specifically credit-card fraud and healthcare insurance fraud, system intrusion detection and healthcare. These will be searched specifically for literature and besides this a more general search in outlier-detection will be done.

2.1.3.1 Key-words

In the data-mining field outlier-detection and anomaly-detection have been used synonymously, the keywords will contain synonyms to facilitate a more comprehensive search. In Table 4 an overview is presented.

Data-mining			
Outlier detection	Anomaly detection	Deviant behaviour detection	
Approach	Method	Technique	
Survey	Literature study	Overview	Review
Pattern recognition	Behaviour patterns		
Logs	Access-logs	Event-logs	

Table 4: Overview of keywords and synonyms

These keywords were used mostly in combination with one or two others to make sure literature related to data-mining and outliers-detection was primarily encountered.

2.1.3.2 Fields

As mentioned earlier some areas of outlier-detection have received a lot of attention. Keywords related to these field were added to the keywords mentioned earlier in this chapter to search specifically in those fields.

Fraud		
Credit-card fraud		
Insurance fraud		
Healthcare		
System intrusion detection	IDS	Intrusion detection systems

Table 5: Overview of fields and synonyms

2.1.3.3 Inclusion & Exclusion

To only include papers that are relevant to the research. Some criteria have been made.

- Inclusion criteria
  - The paper is directly related to outlier-detection
- Exclusion criteria
  - The paper is not written in English
  - The paper is only about knowledge discovery via pattern discovery
  - The paper is not about outlier-detection.
  - The paper is only theoretical and does not apply the outlier-detection technique in a practical context.
  - Papers that focus on outlier-detection for the sake of removing them from the data set, for better pattern recognition.
  - The paper assumes infinite time and money is available and doesn't look at optimization.
  - The paper is not available in full text

2.1.4 Findings

This chapter lists the findings found in the literature. This is done in two step approach related to the knowledge questions mentioned at the start of this survey. In step one we use a decision tree to order all found literature on type of outlier-detection approach used. This tree uses questions about the approach used in the paper to classify it as one of eight approaches. The tree can be found in Illustration 1. The eight approaches the tree concludes with are as follows:

Type of approach	**Elaboration**
Supervised approach	Only one supervised algorithm is used. This algorithm is a classifier trained with labelled data.
Supervised stacking approach	Multiple supervised algorithms are used. These algorithms are classifiers trained with labelled data.
Supervised hybrid approach	Multiple algorithms are used on label data. Where at least one algorithm is unsupervised.
Semi-supervised approach	Only one supervised algorithm is used on incomplete-labelled data. This algorithm is a classifier trained with incomplete labelled data.
Semi-supervised stacking approach	Multiple supervised algorithms are used on an incomplete-labelled dataset. These algorithms are classifiers trained with incomplete-labelled data.
Semi-supervised hybrid approach	Supervised and unsupervised algorithms are combined on an unlabelled or incompletely labelled data-set.
Unsupervised stacking approach	Multiple unsupervised algorithms are used on an unlabelled

	dataset.
Unsupervised approach	An unsupervised algorithm is used on an unlabelled data-set.

Table 6: Overview of the approaches that result from using the decision tree. On the right a short explanation is given per approach.

During the literature survey the problem arose of lack of uniformity on classifying outlier-detection approaches. Research surveys group outlier-detection techniques in different ways. For this survey a simple and effective grouping was needed. The division presented in Illustration 1 is based on the several surveys on outlier-detection found in the literature. The most common type of division were based on type of supervision or type of algorithm. When grouping on type of supervision (Phua, Lee, Smith, & Gayler, 2012) and (Agrawal & Agrawal, 2015) recognize a fourth group, besides supervised, semi-supervised and unsupervised. This hybrid group refers to the combining of multiple techniques. An example of an hybrid approach is to combine unsupervised and supervised techniques. By first making use of unsupervised techniques like clustering, to label the data, the labelled data can then be analysed with supervised techniques. (Phua et al., 2012) uses the term stacking to describe approaches that use multiple similar algorithms on the same data-set to increase the accuracy of predictions. These terms have been incorporated in the decision tree. The paragraph 2.1.4.1 the findings will be presented per type of approach and in paragraph 2.1.4.2 all algorithms are presented divided in supervised and unsupervised.

2.1.4.1 What are the different approaches used for outlier-detection?

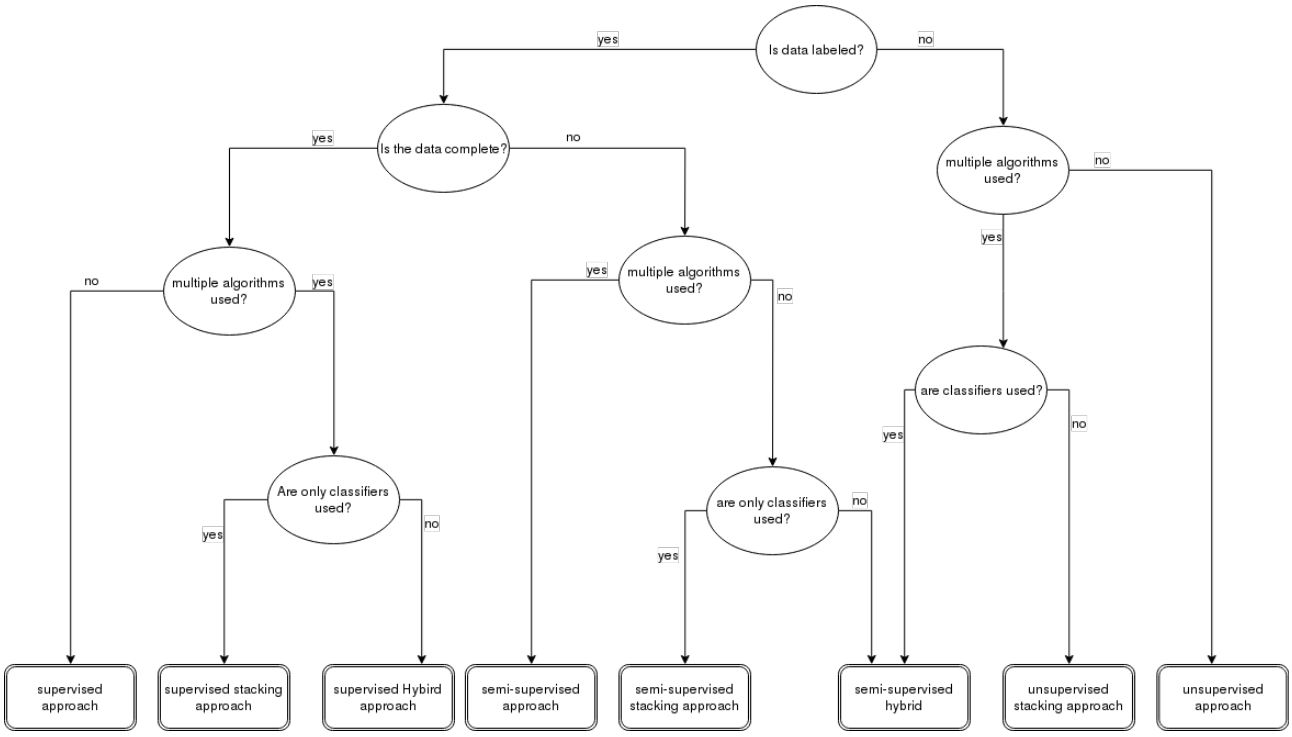

Illustration 1: The decision tree

The following areas have received a lot attention by outlier-detection researchers these are: Creditcard-fraud, insurance fraud, system intrusion and healthcare. Table 7 presents the type of approaches used per field.

Type of approach	**Author**
Credit card fraud	
Supervised approach	(Ngai, Hu, Wong, Chen, & Sun, 2011; Phua et al.,

	2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013; Wang, 2010; Yue, Wu, Wang, Li, & Chu, 2007)
Supervised stacking approach	(Ngai et al., 2011; Phua et al., 2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013; Wang, 2010; Yue et al., 2007)
Supervised hybrid approach	(Phua et al., 2012; Raj & Portia, 2011)
Semi-supervised approach	(Phua et al., 2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013)
Semi supervised stacking approach	(Phua et al., 2012)
Semi-supervised hybrid	(Phua et al., 2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013)
Unsupervised approach	(Kou, Lu, Sirwongwattana, & Huang, 2004; Ngai et al., 2011; Phua et al., 2012)
Healthcare fraud	
Supervised approach	(Aral, Guvenir, Sabuncuoglu, & Akar, 2012; Joudaki et al., 2015; Ngai et al., 2011)
Supervised stacking approach	(Aral et al., 2012)
Supervised hybrid approach	
Semi-supervised approach	(Aral et al., 2012)
Semi supervised stacking approach	(Aral et al., 2012)
Semi-supervised hybrid	(Joudaki et al., 2015)
Unsupervised approach	(Aral et al., 2012; Joudaki et al., 2015; Ngai et al., 2011)
Healthcare	
Supervised approach	(Farrell, Groshen, MacGibbon, & Tomberlin, 2010; Tomar & Agarwal, 2013) (Hauskrecht et al., 2016)
Supervised stacking approach	
Supervised hybrid approach	
Semi-supervised approach	(Farrell et al., 2010)
Semi supervised stacking approach	
Semi-supervised hybrid	
Unsupervised approach	(Bouarfa & Dankelman, 2012)
Intrusion detection	
Supervised approach	(Agrawal & Agrawal, 2015; S.-B. Cho & Park, 2003; Huang & Lee, 2003; Petrovskiy, 2003)
Supervised stacking approach	(Agrawal & Agrawal, 2015; Perdisci, Gu, & Lee, 2006; Petrovskiy, 2003)

Supervised hybrid approach	
Semi-supervised approach	(S. Cho & Cha, 2004)
Semi supervised stacking approach	
Semi-supervised hybrid	(Agrawal & Agrawal, 2015)
Unsupervised approach	(Agrawal & Agrawal, 2015; Petrovskiy, 2003; Zanero & Savaresi, 2004) (Casas, Mazel, & Owezarski, 2012; Leung & Leckie, 2005; Münz, Li, & Carle, 2007)
Outlier/anomaly detection	
Supervised approach	(Chandola et al., 2007; Chandola, Banerjee, & Kumar, 2009; Hodge & Austin, 2004; Kuna, García-Martinez, & Villatoro, 2014)
Supervised stacking approach	(Kuna et al., 2014)
Supervised hybrid approach	(Kuna et al., 2014)
Semi-supervised approach	(Chandola et al., 2007, 2009; Hodge & Austin, 2004; Kuna et al., 2014)
Semi supervised stacking approach	
Semi-supervised hybrid	(Hodge & Austin, 2004; Kuna et al., 2014) (Takeuchi & Yamanishi, 2006)
Unsupervised approach	(Akoglu, Tong, & Koutra, 2015; Chandola et al., 2007, 2009; Duan, Xu, Liu, & Lee, 2009; Hodge & Austin, 2004; Jain, Murty, & Flynn, 1999; Knorr, Ng, & Tucakov, 2000; Kuna et al., 2014; Lee, Yeh, & Wang, 2013; Sreevidya & others, 2014)

Table 7: Overview of type of approach and list of authors that use or mention them. The papers are by grouped by field of application and then ordered using the decision tree presented in Illustration 1.

2.1.4.2 What are the different algorithms used in outlier-detection?

With the overview made in paragraph 2.1.4.1 a broad understanding on what the approaches are used in the different fields. The next step is to give an overview of the algorithms used during every approach. At least one algorithm is essential to perform the actual machine learning. While there are many different approaches, this paper proposes that algorithms can be divided in two groups, these being supervised and unsupervised. This division by supervision was also used when grouping approaches, but then semi-supervised and hybrids were added. A semi-supervised approach is actually a supervised approach using incomplete data and hybrids are combinations of multiple algorithms, these can be supervised and unsupervised. So while approaches can have four types of supervision, algorithms only have two, namely supervised and unsupervised. In table 5 the different algorithms can be found.

Algorithm	**Specific algorithms**	**Authors**
Supervised		
Support vector machines		(Hauskrecht et al., 2016; Perdisci et al., 2006; Phua et al., 2012)

Neural networks	• RBF • Fuzzy • Auto-associative	(Hodge & Austin, 2004; Ngai et al., 2011; Phua et al., 2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013; Yue et al., 2007)
Bayesian networks	• Stage • Bayesian Belief Network	(Ngai et al., 2011; Phua et al., 2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013)
Naive bayes	• Adaboosted	(Ngai et al., 2011; Phua et al., 2012; Sharma & Panigrahi, 2013)
Decision trees	• Boolean logic • Atree • CART • c4.5	(Huang & Lee, 2003; Kuna et al., 2014; Ngai et al., 2011; Phua et al., 2012; Sharma & Panigrahi, 2013)
Rule-based	• Ripper • Slipper • Association rule • Prism	(Kuna et al., 2014; Ngai et al., 2011; Phua et al., 2012)
Case-based reasoning(CBR)		(Phua et al., 2012)
Regression	• Multinominal logistic model(MNL) • nested multinominal logistic model(NMNL) • model-averaged • Auto regression • hierarchical Bayes	(Petrovskiy, 2003; Phua et al., 2012; Sharma & Panigrahi, 2013; Yue et al., 2007)(Farrell et al., 2010; Takeuchi & Yamanishi, 2006)
Nearest Neighbour Method		(Ngai et al., 2011; Petrovskiy, 2003; Sharma & Panigrahi, 2013)
Hidden Markov Model		(S.-B. Cho & Park, 2003; Ngai et al., 2011; Raj & Portia, 2011)
Evolutionary algorithms	• Genetic algorithm • fuzzy genetic algorithm	(Ngai et al., 2011; Phua et al., 2012; Raj & Portia, 2011; Sharma & Panigrahi, 2013)
Dempster-shafer and bayesian learning		(Raj & Portia, 2011)
Bayesian estimation		(S. Cho & Cha, 2004)
unsupervised		
Link analysis		(Phua et al., 2012)
Graph mining		(Akoglu et al., 2015; Phua et al., 2012)

Neural networks	• SOM(self organizing maps) • hierarchical kohonnen net	(Hodge & Austin, 2004; Ngai et al., 2011; Phua et al., 2012; Sarasamma, Zhu, & Huff, 2005)
smart sifter		(Petrovskiy, 2003; Phua et al., 2012)
Statistical test		(Petrovskiy, 2003; Yue et al., 2007)
k-means		(Ngai et al., 2011; Phua et al., 2012)
BLAST-SSAHA	• BLAH_FDS	(Raj & Portia, 2011)
Clustering Distance based/Depth based	• COF • LOF • PCA • DBSCAN • k-means	(Casas et al., 2012; Duan et al., 2009; Knorr et al., 2000; Kuna et al., 2014; Lee et al., 2013; Leung & Leckie, 2005; Münz et al., 2007; Petrovskiy, 2003; Pokrajac, Reljin, Pejcic, & Lazarevic, 2008; Tang, Chen, Fu, & Cheung, 2007)
Dynamic programming	• Needleman-wunsch	(Bouarfa & Dankelman, 2012)

Table 8: List of algorithms used in the approaches found Table 7. Order by algorithm, alternative adaption of algorithm and authors, that mention them.

2.1.5 Discussion

A lot of interesting research has been done in the field of outlier-detection. Especially the field of credit-card fraud receives a lot of attention. The application of outlier-detection techniques in this field seems promising. The lack of an uniform way of classifying outlier-detection techniques was surprising however, especially when comparing multiple fields uniformity seems to be limited. This obfuscates the papers and limits the practical use of a paper in an organizational setting. It could be valuable to perform more research in developing a classification standard for outlier-detection techniques that is applicable in multiple fields.
Another surprising discovery was that a large amount of research focuses on supervised outlier-detection techniques. This requires labelled data and is more difficult to produce. Historical data can be used to make labelled data, but there are problems with this. Many companies do not have a lot of historical data easily accessible when they start to consider data-mining projects. When a lot of historical data is available it is very unlikely the data will be in a format that would require limited preprocessing. Besides this, a bigger problem lies in the nature of outliers. Outliers are just a deviation from the pattern. If one specifically searches for specific outliers any change in behaviour intentional or unintentional would hamper the detection system. Instead of using historical data, one could create labelled data. This is however often expensive and time-consuming. These considerations seem to make supervised outlier-detection techniques less usable in an organizational context. Therefore an unsupervised approach seems to be more interesting, but the lack of accuracy could be an issue. Hybrid approaches could solve this, since they combine unsupervised and supervised techniques. Especially in organizational context a focus on unsupervised and hybrid techniques seems fitting.
In an organizational context it is felt this paper would help with quickly sieving approaches so the outlier-detection approaches can be found that are usable in the organizations problem context. The decision tree shown in Illustration 1 should be helpful in this regard.

2.1.6 Limitations

Naturally there are some limitations that need to be discussed regarding this paper. Data-mining is a

broad and relatively new field of research, outlier-detection is a niche in this field. The lack of a standard framework for data-mining techniques creates a need for interpretation from the reader. Evaluating these interpretations is difficult. Since some fields have received more attention than others, approaches for outlier-detection are focused on the specific problems in these fields. This would mean that problems in other fields of outlier-detection would receive less attention, limited attention has been given to this problem.
Another limitation is the focus on outlier-detection approaches. These approaches are used in an organizational setting. This means that paper with a purely mathematical focus on finding the best outlier-detection algorithm might have been excluded.
Papers that where not translated in English have been excluded because of practical reasons. It is assumed that papers are available in English that have similar approaches to outlier-detection techniques.
Papers have been encountered that were not freely accessible when using the network of the Utwente. These papers have not been included in the survey.

2.1.7 Conclusion

This research introduces concepts to better understand data-mining and outlier-detection. A decision tree is presented that helps grouping different outlier-detection approaches. This tree is used to group the outlier-detection techniques found in the literature. The algorithms used by the approaches are presented in Table 8. The findings and the understanding gained during this survey will be used to support the creation of a methodology for analysing access-logs. This means a number of approaches and algorithms that fit with those approaches will be selected from the literature presented in this survey. These approaches will then be tested, to see if they perform adequately in the methodology. This survey will serve as the backbone for approach and algorithm selection in creating an outlier-detection methodology for analysing access-logs. The research questions that will be used to create the methodology, the first three questions can be linked to steps in the methodology and the fourth questions will address the validation of the methodology, the research questions are as follows:

1. How should access logs be preprocessed?
2. How can user groups be discovered in the data?
3. How can we find users that show deviant behaviour?
4. How can the methodology be validated?

These research questions will be answered in follow-up research and should prove sufficient to develop a methodology that uses outlier-detection techniques to detect deviant behaviour by data-mining access-logs.

2.2 Nedap

The data to evaluate the logs has been provided by Nedap Healthcare. This subchapter will discuss who Nedap is and why their logs are applicable to an organizational context. Nedap healthcare is software supplier for the Dutch healthcare market. They provide their Saas product named “Ons” to a multitude of companies active in healthcare. “Ons” is a large software packet that aspires to take care of every need a caretaker has regarding their IT-system. These needs are fulfilled by different applications that combined together make up “Ons”. For example the application “Agenda” supports all schedule activities and “Dossier” provides access to patient information and care-plans. Currently “Ons” makes use of an Enterprise service bus, ESB for short. The ESB handles all communication between applications. Every communication attempt is logged when it arrives at the ESB. The logs used in this thesis are logs that have been generated by users of “Ons” between the first of September and the last of November in the year 2017. This data only consist of inter-application communication, and are less expansive than the audit-trails. In compressed form these

log still took up 80 gigabyte in space.
It can be concluded that the logs Nedap provided can be seen as a natural representation of users active in the healthcare sector. This means that the data is very applicable to be used the evaluate the methodology represented in this thesis.

3 Solution design

This chapter will be dedicated to elaborating on how the solution will be designed. As a whole this chapter should be seen as a blueprint of the methodology. Before describing this blueprint it should be clear how the terms and context are defined, chapter 3.1 will ensure this. The information described in this chapter should ensure that it is understood in which situations the methodology is applicable. After the language this thesis uses is clarified, the methodology can be discussed. In chapter 3.2 a hypothesised end product of the methodology will be presented. An architecture-model will be used as a guideline to understand how the methodology is designed. The chapter will end with a summary of the solution design.

3.1 Dictionary

This subchapter will be dedicated to defining terms and definitions used throughout the paper. In Table 9 a list of definitions can be found and Illustration 2 shows the relation between some of these terms.

Term	**Definition**
Log	A list of requests to a system
Request	An attempt to add, change or receive information from the system. A request is considered instant and finite, as such it can be represented by single indication of time.
Resource	The specific type of information a request is influencing.
User	An entity that has identified itself to the system and can make requests.
Action	An user making a request.
Subject	The entity that the resource is about.
Behaviour	The culmination of all the requests made by an user to the system.
Deviant-behaviour	One or a couple of request made by an user that deviant from what should be normal for that user.
Entry	Data that was created using requests of one unique user.

Table 9: Definitions

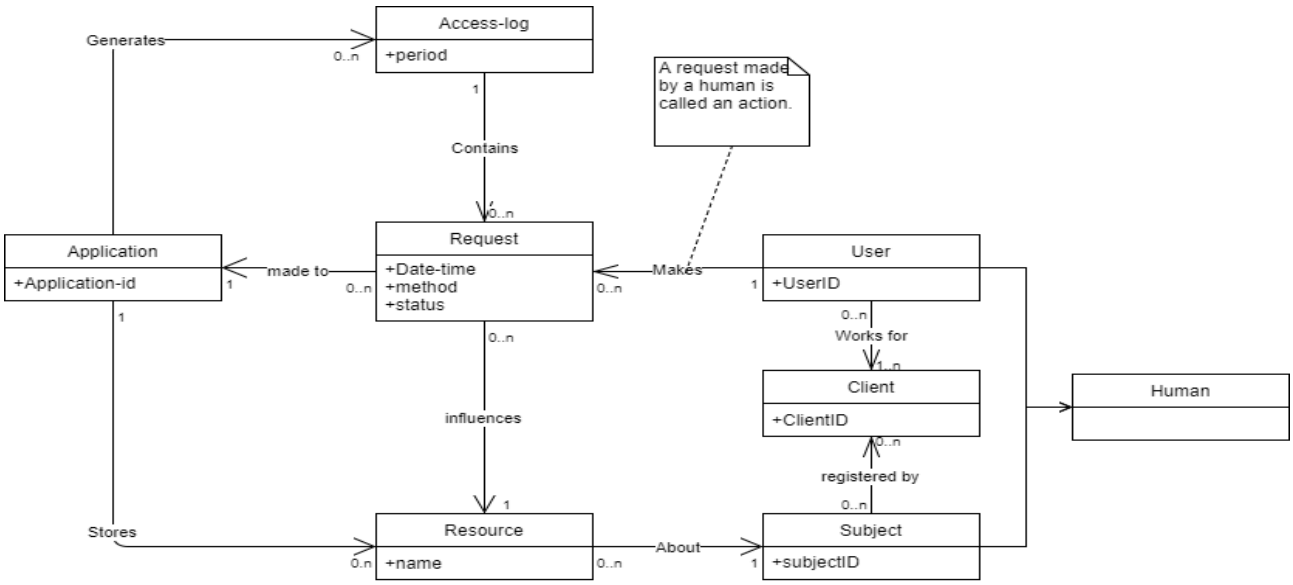

Illustration 2: Domain-model

Following the clarification the language this thesis uses, the solution design will be discussed.

3.2 Hypothesized Method

This section will be dedicated to describing the artefact that should result from the proposed methodology. As the literature in chapter 2.1 suggests the process which transforms data to outliers can be split in multiple parts. One part should focus on the grouping of users and and another part should be the detection of outliers. The reason for this is that deviant requests can be found by detecting behaviour that deviates from normal behaviour. This normal behaviour is found by grouping users together that broadly have similar behaviour. Therefore the grouping of users should be accomplished first. This grouping will be done with clustering algorithms and the outliers will be detected using outlier-detection algorithms. This suggests a hybrid approach as mentioned in the literature. The bpmn-model seen in Illustration 3 is the full representation of the method and its parts. The rest of this chapter will be dedicated to describing every part of the model. As can be seen the model is split up in groups: preparation, clustering and outlier detection. These will form the structure for the coming subchapters. Every subchapter will start with a short summary of the overarching process, then some details of the process will be discussed and the subchapter is concluded with a overview of the different subprocesses.

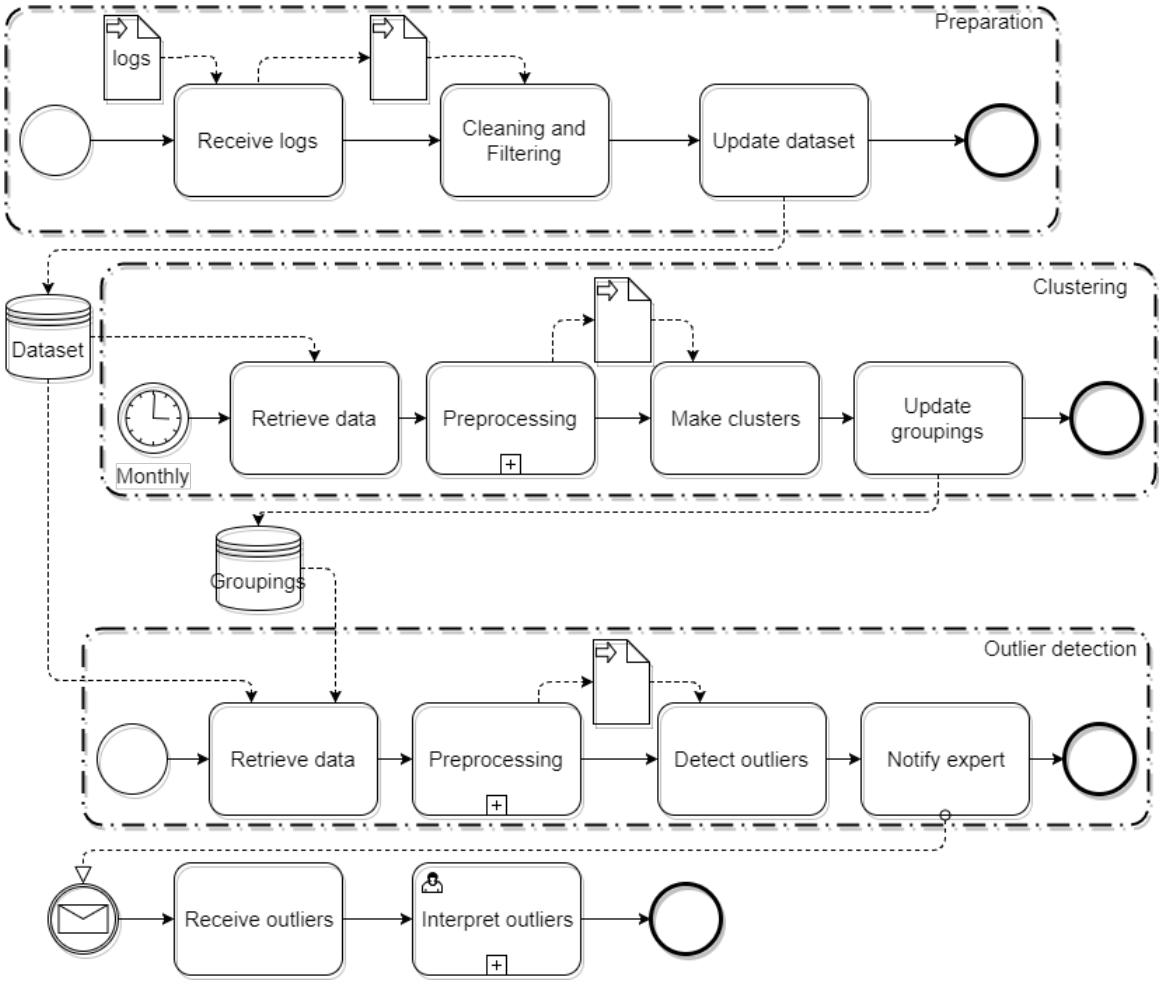

Illustration 3: Proposed-Method

3.2.1 Preparation

In any real-world scenario it would be reasonable to suspect that the logs should need some preparation before they can be analysed. These actions can be done completely separate from the rest of the processes. When the data is prepared appropriately the other processes will proceed

smoother, so it should be done well. The first task is to receive logs. These log are the source of information for the entire process. In Illustration 2 one can see the class-model that is most desirable to have as input. The method still works, however, on less complete input. It should be noted that when talking about incomplete input data, there is a big difference between the lack of a complete variable and a variable being logged inconsistently. It is assumed that when a variable is part of a request in the log. That variable is logged consistently for every request in a log. The core input is the request, but knowing only the fact that something was done on the system is not enough. There are two pieces of information that are critical for the methodology to work.
First, there should be an insight in which user filed the request. As such there should be a way to identify different users. As can be seen the most convenient method is to have a user-id. Other forms of identification are also valid, for example an session-id or IP-address. At this point one can see which users made requests to system.
The second piece of critical information is a way to differentiate between different request. This could be very simple information such as a timestamp, if the request was successful. The optimal solution would be to differentiate using resource. When only using resource the differentiate an issue could arise, which is best explained by an example. if a system is purely used to retrieve telephone numbers, knowing that the resource requested is a telephone will not be very useful. In this case knowing the subject, the person that the telephone-number belongs to, would allow for a lot more differentiation. If besides retrieving telephone-numbers users could also change the numbers in the database, then that could be a source of differentiation as well. In this case a subject would be valuable but not necessary. Hence both resource and subject are optional as long as there is some way to differentiate between different requests.
If a system consists of different applications that work closely together, knowing which application was the target of a request would allow for differentiating between variables. In this case an application id is useful but not required. Now it is evident what information is necessary the class-model has properly been discussed, unfortunately this does not cover everything that is required from the input. It would be beneficial that the data used to create user-groupings spans a period of time. It is assumed that user behaviour standardizes over time, this would decrease the effect irregularities might have on the clustering process.
Requirements have been developed keeping the above in mind. To use the logs each request in that log should full-fill the two critical requirements. A set of optional requirements have been added that would be convenient but are not necessary.
Critical requirements.

1. The request should be related to an user, for example the user-name is part of the log line.
2. One can differentiate between requests.

Optional requirements:

1. The request should be associated with a date and time.
2. Information should be available if the request was completed successfully or not.
3. If the logs consists of requests made by users of different companies, requests or should be associated to their respective company.
4. The resource that was requested or manipulated should be provided.
5. If a resource is about a subject, the log-line should be relatable to that subject.

It is highly likely that the requirements can not be ensured when supplying the logs as input, but after proper filtering the requirements can be fulfilled. Ensuring that requirements are met should be the first step of the cleaning and filtering process. During this step any request that does not full-fill the requirements should be removed from the dataset. The cleaning and filtering process purpose is to transform al the requests in a log to a standard format that can be added to the dataset. Every variable that is going to be analysed should have the same format across the entire dataset.
The end of the preparation phase should be an addition to the dataset. The dataset should consists of

requests without any missing variables, where each variable is easy to access. So different combinations can be used as input in the following phases.

Process name	**Input**	**Output**	**activity**	**Analysis technique**
Receive logs	System logs	List of parsed requests	All information to each unique request is parsed into an array of variables.	-
Cleaning & filtering	List of parsed request	Filtered list of requests	All requests that are not meeting the requirements are removed from the dataset.	-
Update requests	Filtered list of requests	Updated dataset	The requests are added to the dataset in an standard format.	-

Table 10: overview of the preparation processes

3.2.2 Clustering

The dataset will be used to group users of similar behaviour using clustering algorithms. As mentioned earlier when grouping users the dataset should be standardised, as such in the model the start-event only triggers monthly. This way when retrieving data from the dataset it will span a period of time and will be standardised. This data will serve as input in the preprocessing process. The goal of this process is to transform the data so that the clustering algorithm can analyse the data and create clusters. Since it's purpose is to create clusters of users, the data needs to be transformed to a format where every entry is one user. When inputted the data is a list of requests over a time period and many users will have made multiple requests. These requests made by the same user must be aggregated to concisely represent the user. This can be approached in multiple ways and in chapter 4.2 preprocessing methods used in the prototype are described in detail. Clustering algorithms have differences in the type of data they accept, so part of the preprocessing is dependent on what algorithm is used for clustering. Preprocessing also has an influence on the clustering however, import parts are feature selection and normalization. Feature selection refers to selecting specific variables from the dataset to use as input for the algorithms. It requires experimentation to know which features to select.
After preprocessing is done the cluster algorithm will make clusters of users. Every clusters represents a group of users with similar behaviour. The users will be linked to their respective groups and added to the group-dataset.

Process name	**Input**	**Output**	**activity**	**Analysis technique**
Retrieve data	Dataset, specific details	List of request	Retrieves from the dataset a certain group of requests according to the specified details.	-

Preprocessing	List of requests	Entries of users	Transforms the dataset to be used as input for the clustering algorithms	Vector normalize, Minmax scaler, standard scaler, Wout normalize, Binary processing, disimilarity matrix
Make clusters	Entries of users	Clusters	Analyses the users and assigns them to clusters	Kmeans, Agglomerative clustering
Update groupings	Clusters	Updated groupings dataset	Assigns every user to a group according to their clusters and updates the groupings dataset	-

Table 11: Overview of the clustering processes

3.2.3 Outlier-detection

The group-data created in the clustering phase will be used to select only users in the dataset with similar behaviour. This dataset will then be preprocessed, the same considerations apply as when clustering with one exception. User behaviour does not have to be aggregated into a single entry. A valid method is to do outlier-detection on all request in the group without taking the user into account. The preprocessing output is still dependent on the outlier-detection algorithm however. The outlier-detection algorithms output will be outliers, these have to be judged by an expert. When experimenting the outliers need to be checked for validity and when in practical use the outliers are send to an information officer to analyse and act upon. Since a human element is involved the outlier detection algorithm should not generated a lot of false positives so that the amount of information sent to the officer stays manageable. This methodology stops at the end of the outlier-detection process, the process of judging the outliers is very dependant on the type input. The outliers found should be analysed to find deviant behaviour. The outliers are the output that will support information security, by presenting them to information security officers.

Process name	**Input**	**Output**	**Activity**	**Analysis technique**
Retrieve data	Dataset, user-grouping, specific details	List of requests	Select the requests of users associated to the specified group according the the specified details	-
Preprocessing	List of requests	List of requests	Transforms the dataset to be used as input for the outlier-detection algorithm	Vector normalize, Minmax scaler, Standard scaler, Wout normalize, Binary processing
Detect outliers	List of requests	Outliers	Analyses the requests for	Isolation forest, Birch,

			outliers	Patternfinder
Notify experts	Outliers	List of Outliers	Send the interesting outliers to the experts this is the end of the methodology	-

Table 12: Overview of the outlier-detection processes

The methodology only works in a certain context, the terms used to describe this context is elaborated upon in chapter 3.1. A more detailed description of the exact data requirements is given in chapter 3.2.1. These requirements are described as part of the preparation phase. This is the first phase of the three phases of the solution design as described in Illustration 3. The preparation phase should ensure that the data is prepared in such manner that the clustering phase and the outlier-detection phase perform smoothly. The clustering phase links users together by using clustering algorithms. These user-groupings are then used during the outlier-detection phase as input for the outlier-detection algorithms to detect outliers. These outliers are requests that deviate from the normal request behaviour of similar users. The methodology described in this chapter is used as a guideline to make the prototype described in chapter 4.

4 Prototype

The hypothesized method will be tested with an expository instantiation. This prototype has been used to experiment with different approaches. The Archimate modelling language has been used to model the architecture of the prototype, this model can be found in Illustration 4(Iacob, Maria Eugenia et al., 2012). The model is split up in a business layer and an Application layer. Colour-coded yellow and blue respectively. The business layer shows the moment where the user has to interact with the process. The application layer represents the systems of the prototype. As can be seen the prototype is split up in three applications. Each application has a specific function that coincides with the phases as described in chapter 3.2. This chapter will dedicate a subchapter to each application. The subchapter will consist of a broad overview of the function of the application and then go into detail in explaining the application logic.

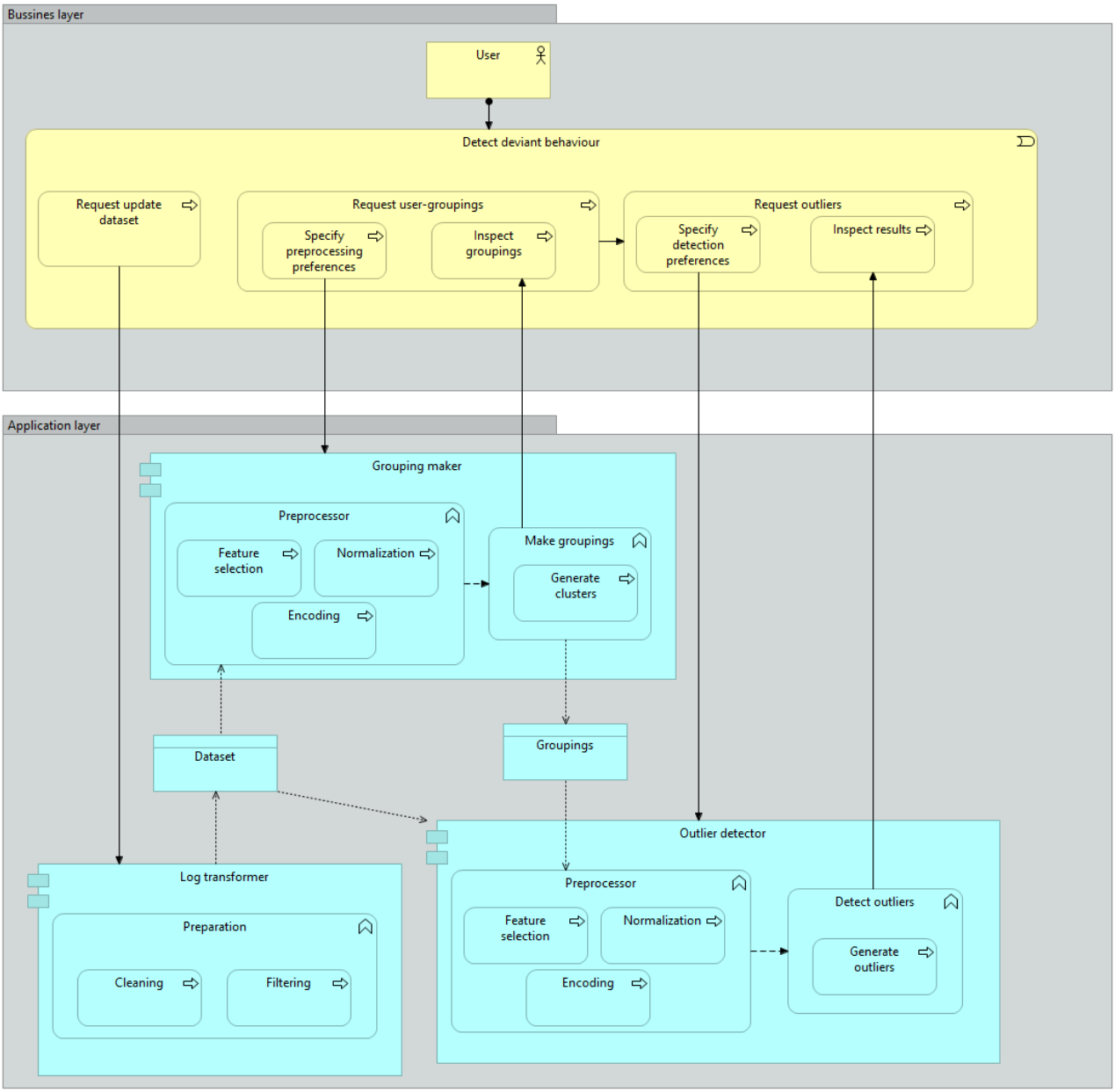

Illustration 4: The prototype architecture

4.1 Log-transformer

Preparation is the first part of the methodology and this application is responsible for that process. In a real world scenario it is quite uncommon to receive the dataset in the exact format that is needed. The preparation process purpose is to change the input to a clear and defined format that can be used as input for the other applications. Its output is a dataset as can be seen in Illustration 4. This dataset should consist of requests that are represented by different variables, that dataset should allow for the detailed selection of request and variables of request. So that different approaches can be experimented with. The preparation process primarily consist of two sub-process, namely that of cleaning and filtering. These sub-process are interwoven with each other. This subchapter should give a clear view of how the input that is used for the other application has come to be.

4.1.1 From logs to data

Access-logs are not uniform and the type of access-logs and the information stored in these access-logs are dependent on the supplier of the logs. Nedap provided the access-logs used in this research, this has been done using Nginx. In Ons(Nedaps Saas) Nginx serves as a reverse proxy server situated in front of the ESB. This access-logs consisted of lines and each line represents a request as defined in chapter 3.1. Therefore in this case a collection of log-lines is equal to a collection of requests. Nginx allows for some freedom when defining what a log-line consists of. After some considerations it was decided to let the log-lines have the format displayed in Table 13 :

Variable	**Notes**
Timestamp	Standardized time format(day/month/year:hour:minutes:seconds +timezone)
Request-method	The HTTP method the request used
Request-URL	The URL the request used
Status	The http result-code
Customer-identification-code(Cic)	Customer identification code(will be hashed)
Source-application	Application which the request originated form
Username	Username(will be hashed)
Quality of service	Type of request. This was used to filter background processes
Audit-UUID	An audit identification number

Table 13: Example log-line

An example log-line would look like this, The SubjectID, CustomerID and Username are replaced by dummy-data for obvious reasons:
“*02/Oct/2017:06:34:34 +0200|GET|/uc/clients/12345/main_address|200|Cu1234|Deployment|User1*|-|-”

Variable	Example
Timestamp	*02/Oct/2017:06:34:34 +0200*
Request-method	*GET*
Request-URL	*/uc/clients/12345/main_address*
Status	*200*
Customer-identification code(Cic)	*Cu1234*
Source-application(Src)	*Deployment*
Username	*User1*
Quality of service	-
Audit-UUID	-

Table 14: The example log-line split up according to variable

A collection of log-lines such as the example log-line seen here is the actual input given to the preparation process. The data will now be transformed to meet the specified requirements and added to the dataset.

4.1.2 Cleaning and filtering

The cleaning and filtering processes are tightly interwoven. During this process the critical requirements defined in chapter 3.2.1 will be ensured and when possible optional requirements will also be fulfilled. After the cleaning and filtering the output should be ready to be added to the dataset. The following subchapters will consist of examples on how this was achieved.

4.1.2.1 Splitting the URL

Request-URL is the variable that gives the most information about the kind of action that the user is performing on the system. In the example, Request-URL is "*/uc/clients/12345/main_address*". This variable actually consists of three pieces of information. These pieces of information allow for the optimal requirement described in chapter 3.2.1 to be met, this requirement is defined as: "If a resource is about a subject, the log-line should be relatable to that subject." Because of this the request-URL has been split in three parts. These parts are Path, SubjectID and Resource. Path represent a general indication of where the requested information is situated. SubjectID and Resource are defined in chapter 3.1. In Table 15 one can find an example of such URL is split.

Variable	example
Path	*/uc/clients*
Id	*12345*
Resource	*Main_address*

Table 15: Example of how request-URL is split up.

These variables are added to the dataset as separate variables instead of request-URL.

4.1.2.2 Selecting data

Not all log-lines are relevant for the dataset, since every access-attempt on the system is regarded as a request. Some background tasks result in requests not made by users. These among other things should be removed from that dataset. The input allowed for some stricter and more detailed requirements than defined in chapter 3. Only logs meeting the following requirements have been added to the dataset:

- The log-line must contain an user-name
- The log-line must contain an Customer-id
- The log-line must be related to an action relevant for caretakers.
    - This requirement was met by only accepting log-lines with paths(part1/part2):
        - part1: [uc, cc]
        - part2: [clients, dossier, employees, evs]

4.1.2.3 Cleaning data

This prototype receives as input data from a real-world company, as such it was not completely standardized. To ensure the data's uniformity some work had to be done. For example the format described earlier when splitting up the request_URL was not always possible. In some cases the variable Resource was not always present, because when a request was made for all the data about a client the request-URL could look like this: "*/uc/clients/12345"*. In Nedap's case this URL means that a overview is requested. These kind of exceptions to the rules had to be accommodated for in the prototype.
A special mention should be given to the following URL: "*/uc/clients/multiples?id=12345id=12346id=12347"*. This URL request information about multiple client_ids at once. Since this had to be represented somehow, it was decided to make an unique log-line per id. Resulting in the preceding example changing to this:

- "*/uc/clients/12345/multiples"*
- "*/uc/clients/12346/multiples"*
- "*/uc/clients/12347/multiples"*

This approach does increase the amount of actions an user makes on the system, but it was found to be the most convenient solution to the problem.
There were other cases of Request_URL not having the standard format. When it was possible to retrieve the information needed to fill the variables Path, SubjectID and Resource this was done so, otherwise all instances of that type of request were removed from the dataset. Everything described here is part of ensuring that the output of this process consisted of requests in an uniform format.

4.1.3 Dataset

According to Illustration 4 the output of the preparation process is added to the dataset. The dataset is then used as input for the other processes of the prototype. While the dataset is not part of the log-transformer application it seemed fitting to elaborate it in this subchapter. The dataset consists of requests and every requests has the following Format: "*Date*|*Method*|*Path*|*SubjectID*|*Resource*|*Status*|*CustomerID*|*Src*|*Username*". See Table 16 for more clarification. As can be seen the variables *QoS (Quality of Service)* and *AuditID* were removed. QoS served its purpose during the filtering of the data. *AuditID* would only be relevant if the research detected results that needed further evaluation, with the AuditID variable the specific action could be found within the audits. This variable is not relevant for data-mining however.

Variables	Notes
Timestamp	Standardized time format(day/month/year:hour:minutes:seconds +timezone)
Request-method	The HTTP method the request used
Path	The URL the request used.
SubjectID	Subject id(hashed)
Resource	The information about SubjectID requested by User
Status	The http result-code
Customer-identification code(Cic)	Customer identification number(hashed)
Source-application(Src)	Application which the request originated form
Username	Username(hashed)

Table 16: The format of a log-line in a data-set

After the cleaning and filtering of the data, assurances could be made about the dataset. Some occurred implicitly because of how the data was generated by Nedaps systems. Other assurances were ensured explicitly during the cleaning and filtering.

- Implicit Assurances
  - Every log-line contains a date.
  - Every log-line contains a request method.
  - Every log-line contains a path.
  - Every log-line contains a status.
  - Every log-line contains a source application.
- Explicit Assurances
  - Every log-line contains an username.
  - Every log-line contains an Customer identification code.
  - Every log-line contains an SubjectID.

The dataset is the collection of all the output generated by the log-transformer. It ensures above all that every request and all its variables have the same format.

4.1.3.1 Storage

This prototype stored the dataset in multiple csv-files. Every csv-file consisted of all the request made by user of one customer covering 24-hour period. The 24-hour period division occurred naturally since Nedap supplied a different log file for each day. The csv-files are split by costumers since Nedap serves a variety of different types of companies. Employees of different companies could have vastly different behavioural patterns. When experimenting the ability to select data from one company proved to be convenient.
It was expected that behaviour would standardize when looking at data over a longer period of time. It became clear that merging the dataset at the start of every experiment was time consuming. Because of this multiple csv-files of the same costumer were merged together into one. This merged file contains all the requests made by users of one customer in a certain period. The timespan of these periods could for example be a week, two weeks are a month. This proved to be a faster way of loading in data when experimenting with behaviour covering multiple days.

The function of the log-transformer is to transform the logs to standardized requests that can be

added to the dataset. Each process should help to ensure this. The end result of the log-transformer application is an update to the dataset. The dataset should contain requests that consist of the variables seen in Table 16. The dataset can then be used as input for the grouping-maker and outlier detector applications.

4.2 Preprocessing

The preprocessing process is part of both the grouping-maker and outlier detector applications. The details of the preprocessing process are greatly influenced by specifications made by the user and which algorithms is being used. Still there is a lot overlap between the preprocessing process of the grouping-maker application and that of the outlier-detector application. That is why a separate chapter is dedicated this process. The preprocessing process purpose is to ready the data to be used as input for the specified algorithm. Algorithms have different needs when accepting input and how they perform is very dependent on the data. The data need to be transformed in such a manner that the algorithms can properly analyse it, this can be done in vastly different ways. Illustration 4 describes three sup-processes within the preprocessing process, these are: feature selection, encoding and normalization. These will be elaborated on in the following sections.

4.2.1 Feature-selection

Feature-selection is the act of selecting the features to be used during the approach. The features that can be selected are the variables that can be selected in the dataset, for an overview see Table 16. Since the whole purpose of the master-thesis is to discover information about users. User is a variable that is always selected. This is not the case for the other variables, variables could be excluded because they are not descriptive enough, the algorithm doesn't accept the variables format or other reasons. The features that are selected are specified when activating the prototype. The output of this process is an extraction from the dataset. This extraction is a collection of requests over certain period with only the specified variables associated to each request. In essence this process is just the selection of variables to be used as input for further processes. Hence instead of application logic some interesting observations are discussed in the rest of this subchapter.

These observations will be discussed while referring to the variables as described in Table 16. The variable Date has seen limited use as input for datasets and was mostly used to give extra context to the results. This is mainly because most algorithms did not accept dates as an input. Since the dataset has been split per company as described in subchapter 4.1.3.1 it would serve no purpose to use it as input. Status and SubjectID are more interesting variables to discuss in this context. The status variable represents the http-response code of the request. It turned out that approximately 99% of all request had a response code of 200 (success). Thus it was not really a descriptive variable for making user-groupings. Using it as input when finding outliers is not really rational, since other security system are better equipped to detect such obvious outliers.
SubjectID proved to be problematic, as companies serve a lot of different subjects. The data expanded massively when encoding the data, the concept of encoding is explained in the next subchapter. While encoding is very useful it caused the size of the data to increase. For example if a thousand users request information about a different subject each, the encoded version of the would have thousand variables per user. Since an user serves more than one subject, the hardware proved to be insufficient to handle this in a short time-frame. For this reason only one approach was made that focused purely on using th*e SubjectID. Th*e next step in the preprocessing process consists of encoding this process was frequently used during the experiments. After encoding has been explained a quick return to feature selection is made to explain the effect of encoding on the feature selection process.

4.2.2 Encoding

A quick glance at Table 17 shows that the data is relatively uniform in type, only Timestamp and Status differ from the other variables. Status represents http status requests with a value of 3 digits.

Each number represent something different. For example 200 represents success and 400 represents a bad request, this means that it is still nominal data.

Variable	**Format**	**Type**
Timestamp	Date	Ordinal
Request-method	String	Nominal
Path	String	Nominal
SubjectID	String	Nominal
Resource	String	Nominal
Status	Integer	Nominal
Cic	String	Nominal
SRC	String	Nominal
Username	String	Nominal

Table 17: Data type overview

Nominal data cannot be used as input in the format represented here. To make it usable a method called one-hot encoding was used. The purpose of the encoding process is to transform the input in a binary table. This binary table is accepted as input by algorithms whereas nominal data is not. This is best explained with the following example. One-hot is a term used in digital circuits to describe a system where the legal combinations are only those where one value is high(hot) and al other values are low(cold). The idea is to present a variable by having a binary row with al possible options, the option the variable represent is given a value of 1(hot) and all other values are set as 0(cold). This is best explained by an example. In Table 18 you can see a table filled with two variables, Username and Path, from six hypothetical log-lines.

Log-line	**User**	**Path**
1	Alice	Uc/employees
2	Bob	Uc/clients
3	Claire	Uc/dossier
4	Alice	Uc/clients
5	Alice	Uc/clients
6	Claire	Uc/dossier

Table 18: Example dataset of six log-lines only consisting of user and path.

If this table is one-hot encoded the result would look like Table 19.

Log-line	User	Uc/employees	Uc/clients	Uc/Dossier
1	Alice	1	0	0
2	Bob	0	1	0
3	Claire	0	0	1
4	Alice	0	1	0
5	Alice	0	1	0
6	Claire	0	0	1

Table 19: Example of One-hot encoded data based on the data in Table 18

This format can be processed by an algorithm. One should note that encoding the data increases the size of the data a lot, because a new variable has been created for every value the original variable had in the dataset. As can be seen log-line one, four and five have the same user. To compare users all log-lines of the same user need to be merged. There are multiple approaches that can be used to have one set of variables per user, but first some implications regarding feature selection that resulted from encoding data will be discussed. The next sub-chapter will discuss the merging of some variables which became necessary because of the way encoding transforms the data.

4.2.3 Merging Method, Path and Resource

It became apparent during talks with experts from Nedap, that some variables have more descriptive power together with other variables than on its own. The variables Method, Path and Resource have been merged together with the variable MPR as result. The reason for this becomes apparent with an example in Table 20 you can see some reduced log-lines from a dataset.

Method	Path	SubjectID	Resource	SRC	Username
GET	*/uc/clients*	12345	main_address	Dossier	Alice
GET	*/uc/clients*	54321	main_address	Dossier	Bob
GET	/uc/employees	12345	main_address	Employee-portal	Claire
POST	*/uc/clients*	12345	main_address	Dossier	David

Table 20: Example log-lines in the format: Method|Path|SubjectID|Resource|SRC|Username

The log-lines can be given context quite easily. The variable method has two options in these log-lines GET and POST. In the case of our system GET is used when retrieving data and POST is used when changing data. So in this case Alice is requesting to see the address of a client with id 12345. Bob is requesting to see the address of a client with id 54321. Claire is requesting to see the address of a fellow employee with id 12345. David is requesting to change the address of a client with the id 12345. It can be summarised that Alice and Bob are doing the same type of action on a different client. Claire is doing a different action she is looking at the address of an employee, not a client. David is requesting changing the address of the same client Alice is requesting to see. While the subject is the same, the intent of the request is wildly different. An approach that was experimented with was to only look at the action the user was doing and disregard the SubjectID. This way people with the same job description but working with different clients should show similar behaviour. A more elaborate reasoning for this approach will be explained in the coming chapter. For the continuation of this subchapter the client id will be disregarded. In the preceding subchapter encoding was explained, that method has been applied on the example log-lines in Table 21.

GET	POST	uc/clients	Uc/Employees	Main_address	Dossier	Employee-portal	Username
1	0	1	0	1	1	0	Alice
1	0	1	0	1	1	0	Bob
1	0	0	1	1	0	1	Claire
0	1	1	0	1	1	0	David

Table 21: The example log-lines shown in Table 20 one-hot encoded.

The encoded log-lines have seven encoded variables and an username. If one counts the columns Alice has a 100% match with Bob, which makes sense since both were performing identical actions but on different clients. Claire has a two values difference with David, resulting in 71% match. Alice has four columns different compared with Claire, resulting in 42% match. The difference between the request of Alice and David is however less pronounced than this calculation suggest, since where Alice requested an address David changed an address. To make this difference more pronounced these three variable have been merged in a new variable MPR. In Table 22 one can see the new dataset and in Table 23 the data is encoded.

MPR	SubjectID	SRC	Username
GET|/uc/clients|main_address|	12345	Dossier	Alice
GET|/uc/clients|main_address|	54321	Dossier	Bob
GET|/uc/employees|main_address	12345	Employee-portal	Claire
POST|/uc/clients|main_address	12345	Dossier	David

Table 22: The dataset with MPR variable

GET|/uc/clients| main_address|	*GET|/uc/employees |main_address*	*POST|/uc/clients| main_address*	Dossier	Employee-portal	Username
1	0	0	1	0	Alice
1	0	0	1	0	Bob
0	1	0	0	1	Claire
0	0	1	1	0	David

Table 23: The one-hot encoded version of Table 22

As can be seen in Table 23 now there are only five variables. Alice and Bob still have a 100% match. Alice and Claire have no match at all, they were looking at completely different data so that makes sense. Alice and David have 40% match. They are still using the same application but filling a different kind of request. It was felt this could be a more honest representation of the dataset. Most experiments were tested with and without the MPR variable. Hereby, the discussion about the feature selection process and the encoding process has ended. Both proved useful in representing the data in such a way that it can be accepted by algorithms.

4.2.4 Normalization

Normalization is the last subprocess of preprocessing. This subprocess is divided into many approaches, which normalization approach to use is dependant on the type of algorithm that is used, the output of the preprocessing process and the specifications of the user. The output of this process should be a normalized format that can be used as input for algorithms that either detect groupings

or outliers. Several normalization approaches will be discussed in the following sub-chapters, but first an issue will be discussed that creates a necessity for the use of entries. After the feature selection and the encoding processes the output is still a collection of requests. In Table 24 one can find an example that is as input for the examples of every approach.

GET|/uc/clients| main_address|	***GET|/uc/employees| main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Username**
1	0	0	1	0	Alice
1	0	0	1	0	Alice
0	0	1	1	0	Alice
1	0	0	1	0	Bob
0	1	0	0	1	Bob
0	1	0	0	1	Bob
0	0	1	1	0	Claire
0	0	1	1	0	Claire

Table 24: Example dataset used for examples in every approach.

Multiple request in the collection will be associated to the same user. To make it easier to compare users all requests made by the same user have been condensed into one entry. This collection of entries are then used for the rest of the normalization process instead of requests. In Table 25 one can see the data from Table 24 in entry format. The encoded variables were normalized in groups. These groups were based on the original variable that was used to create the encoded variables. For example all variables that originated from SRC where normalized as group, this is represented by the colour scheme in Table 25.

GET|/uc/clients| main_address|	***GET|/uc/employees| main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Username**
2	0	1	3	0	Alice
1	2	0	1	2	Bob
0	0	2	2	0	Claire

Table 25: Light-grey is the original MPR variable and dark-grey is the original SRC variables.

4.2.4.1 Vector-unit-norm

The vector-unit-norm is a simple and commonly used approach. The are multiple norms that can be used, but during this research only the L2 unit-norm was used. The documentation on the library used can be found on the internet using the following reference: (‘sklearn.preprocessing.normalize — scikit-learn 0.19.1 documentation’, 2018). The result is shown in Table 26.

GET|/uc/clients| main_address|	***GET|/uc/employees| main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Username**
0.89	0	0.45	1	0	Alice
0.45	0.89	0	0.45	0.89	Bob
0	0	1	1	0	Claire

Table 26: The values normalized using the vector unit-norm.

4.2.4.2 Standardization

This approach is commonly used to standardize and is a very basic and standard standardization approach(‘sklearn.preprocessing.scale — scikit-learn 0.19.1 documentation’, 2018). This method consists of removing the mean and scaling the features to unit-variance. This results in Table 27.

GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Username**
1.22	-1.22	0.00	1	-1	Alice
0.00	1.22	-1.22	-1	1	Bob
-0.71	0.71	1.41	1	-1	Claire

Table 27: The values transformed using standard scaler

As can be seen the values of the feature fluctuate around the -1 and 1.

4.2.4.3 Min-max Scaler

The min-max scaler is commonly used as an alternative to the previous approach ‘sklearn.preprocessing.MinMaxScaler — scikit-learn 0.19.1 documentation’, 2018). The min-max scaler yield slightly different results using Table 25 as a starting point This approach puts more of a focus on the extremes of the variables, yielding in the results of Table 28.

4.11 The values transformed using min-max scaler					
GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Username**
1	0	0.5	1	0	Alice
0.5	1	0	0	1	Bob
0	0	1	1	0	Claire

Table 28: The values transformed using min-max scaler

As can be seen this result is a more simple presentation of the data.

4.2.4.4 Wout-normalize

One potential issue with preceding normalization methods was that some very descriptive actions that occur only a few times are overshadowed by other actions that occur a large amount of times. This alternative normalization method was approached to test this overshadowing of descriptive actions. By squaring every variable by 1/50 variables would transform to be closer to one, large values would reduce in size significantly while small values would barely change.

GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Username**
1.01	0	1	1.02	0	Alice
1	1.01	0	1	1.01	Bob
0	0	1.01	1.01	0	Claire

Table 29: Normalized via the Wout method

In the example can be seen that this result in very minimal difference. These difference would be larger in the real data-set because the example data, that is being used here consist of very small amount of requests, where in the real dataset. The occurrence of certain variable would be a lot larger.

4.2.4.5 Binary processing

Since simplicity seemed to generate interesting results, a very simple approach was taken. If a variable was requested by a user one or more times, the variable would be set to one. In any other case the value would remain zero. The resulting table for the examples case can be found in Table 30.

GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Username**
1	0	1	1	0	Alice
1	1	0	1	1	Bob
0	0	1	1	0	Claire

Table 30: The values-transformed via binary processing method

All the standard normalization approaches used by this research have now been discussed. The next subchapters will discuss some situational approaches that were taken during preprocessing.

4.2.4.6 Dissimilarity matrix

The dissimilarity matrix was used to simplify and reduce the size of the dataset. After the normalization process every user was represented by an entry consisting of multiple variables. These entries can then be used to calculate the root-mean-square-error (RMSE) between two users. The RMSE represents the dissimilarity between two users. A dissimilarity matrix is made by calculating the RMSE between every user in the input and then adding the values to a matrix that has all the different users on its axis. This dissimilarity matrix is a simpler representation that can be used as input for the grouping-maker application.
RMSE is frequently used to calculate differences between samples of a population. In this case it is used to calculate the difference between users. The specific calculations necessary to arrive at an RMSE is readily available on the internet. An example dissimilarity matrix using the values of Table 30 can be found in figure Table 31

Username	Alice	Bob	Claire
Alice	0	0.77	0.45
Bob	0.77	0	0.89
Claire	0.45	0.89	0

Table 31: The dissimilarity matrix created using RMSE.

As can be seen when comparing the same user the value is zero, since there is no difference in their behaviour. Compared to our original input the amount of variables in the dissimilarity matrix is reduced significantly. The dissimilarity matrix is a useful tool when comparing large amounts of users.

The preprocessing process can be approached in a lot of different ways and this thesis only elaborated on a single droplet of water in an ocean of possibilities. The end of the process must always result in output that can be used by the grouping-maker and outlier-detector applications.

4.3 Grouping-maker

The grouping-maker application is the next to be discussed. The grouping-maker application has a goal the generate a list of users associated to a certain group by using the information from the dataset. The output is a CSV-file that will be added to the groupings database. The user specifies

which algorithm is being used and specific parameters used by the algorithm. How the data is preprocessed is a combination of what is required for using the algorithm and specifications of the user. This application consist of the preprocessing process and the make groupings process. The preprocessing has been discussed at length in chapter 4.2. The Client_id approach is an alternative approach specific for this application and will be discussed in subchapter 4.3.1. Subchapter 4.3.2 specifies which algorithms have been experimented with during this thesis.

4.3.1 Preprocessing

The grouping-maker application always used entries as discussed in subchapter 4.2.4. The grouping-maker deals with large amount of data since it uses all the request made by users over a certain period. This creates the need for simplification to deal with the size of the data, because of this the use of a dissimilarity matrix is also standard. Subchapter 4.3.1.1 discusses a specific approach used only for making user groupings based on SubjectIDs.

4.3.1.1 Client_id approach

Healthcare workers have to take care of many different people and tend to work in teams. These teams should work in a same similar neighbourhood, this could mean they have a lot of client overlap. A hypothesis this thesis tested is the idea that users can be grouped purely on subject. As can be guessed during feature selection only the SubjectID variable was selected. However as it has become clear from earlier examples, the SubjectID does not always represent a client. In a request such as *"uc/employees/12345/main_address"* the subject is not a client but an employee of the company. Accordingly the dataset was filtered further to only include clients. Since this approach is based on creating groups of users that share subjects, all subjects that only appeared in requests by one user were removed from the dataset. After the feature selection has completed the normal encoding principle is applied. The dataset only consist of users and subjects so a simple table can be used to represent the data as shown in Table 32.

	12345	54321	56789	98765
Alice	1	0	0	1
Bob	0	1	1	0
Claire	1	0	1	0
David	1	1	0	0

Table 32: The amount of occurrences related to a subject per user.

The dataset seen in Table 32 can be used as input as is or be transformed to a dissimilarity matrix as explained in chapter 4.2.4.6.

The preprocessing process results in output that is a simple representation of users in the dataset. When a dissimilarity matrix is used the difference between users is represented in a matrix. This output can then be used as input for the make groupings process.

4.3.2 Make groupings

The make groupings process generates groupings of users than can be added to the groupings database. A specific algorithm is selected by the user and uses the input the generate clusters of users. Users within the same cluster are considered to be grouped together. The make groupings process outputs a simple user to group mapping. How the users are mapped depends on the input and the algorithm used to cluster the users. The following subchapters will discusses different cluster algorithms.

4.3.2.1 K-means

K-means is a very common algorithm in machine learning(‘sklearn.cluster.KMeans — scikit-learn 0.19.1 documentation’, 2018). It calculate distances between points and maps them on a n-dimensional field. Every axis is a variable, this way similar similar entries are mapped closely together in clusters. These clusters can be seen as groups during the user-grouping phase. During the outlier detection phase points that are not close to any clusters could hint at deviant behaviour. Therefore this is a simple versatile algorithm.
The input k-means requires is simple. It can only accept numerical data, since it wants to map variables among an numerical axis. The amount of variables makes k-means harder to grasp since every variable in an entry requires another dimension to map. Visualizing more then 3-axis makes the mapping very hard to understand. Because of this, if the preprocessing provides entries purely in an encoded format the result would get very hard to understand. Thus a limited amount of variables per entry would be beneficial. The dissimilarity matrix has thus mainly been used as input when using this algorithm.
K-means is a clustering algorithm thereby it outputs clusters. Since a dissimilarity matrix of users was supplied as input. The output is clusters of users. These clusters can be mapped as groupings.

4.3.3 Agglomerative clustering

Agglomerative cluster learning(ACL) is similar to k-means (‘sklearn.cluster.AgglomerativeClustering — scikit-learn 0.19.1 documentation’, 2018). It also uses distance to calculate clusters. The library that was used allowed for multiple ways to calculate distances, this paper chose to only allow the algorithm to use the euclidean way of calculating distance. ACL generates an hierarchy of clusters. The amount of clusters that the algorithm creates is specified by the user. The algorithm will make new clusters until the specified number and then starts merging clusters which are closest to each other until all entries have been added to a cluster.
ACL’s input is very close to that of k-means. When calculating euclidean distance numbers are necessary, so that limited potential options for variables. A dissimilarity matrix was mainly used as input.
ACL provides two results, a list of entries linked to cluster and a dendrogram. This dendrogram is a hierarchical tree that shows the distance between different clusters. The dendrogram can be used to reorder the axis of the dissimilarity matrix, since the axis are users, the users can be organized per cluster, by using colours to show similarity to each other a useful visual tool can be made to see if strong clusters have been found.

The Grouping-maker application uses some simple but powerful algorithms to generate groupings. The output is an update to the groupings database.

4.4 Outlier-detector

The outlier-detector combines data from the dataset and the groupings database to detect outliers within a certain group. These outliers can then be analysed by a human to see if deviant behaviour has occurred. The output of the application is the request that is detected as an outlier and the user that is associated to the request. Two outlier-detection algorithm were used to generate outliers, these had limited success so an alternative way to detect outliers has been developed, this approach starts in the preprocessing phase and will be discussed in subchapter 4.4.2.

4.4.1 preprocessing

The outlier detection application use the grouping database to only select users from the dataset that share the same group. The user specifies an extra step when using the outlier-detection application. Namely if deviant request or deviant users should be returned. When analysing requests the creation of entries is not necessary, the rest of the normalisation process proceeds as normal however. An approach that does not make use of a machine learning algorithm has also been explored. The preprocessing process is different using this approach, as such this approach will be elaborated upon now.

4.4.2 Pattern finding

Sometimes the best approach is the simplest approach. Instead of using a using machine learning, the more simpler approach of finding percentile differences could lead to fruitful results when detecting outliers. This approach uses groupings generated by other approaches or provided by the companies, to map requests. That mapping can be used to discover common behaviour and abnormal behaviour. This thesis explored four approaches on how this mapping could be done. These will be discussed in the proceeding subchapters. The example data-set seen in Table 33 will be used as a starting point. The example set is similar as Table 24 with the addition that groups have been added. Alice and Claire have been assigned to group one and Bob has been assigned to group two.

GET|/uc/clients| main_address|	***GET|/uc/employees| main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Username**	**Group**
1	0	0	1	0	Alice	1
1	0	0	1	0	Alice	1
0	0	1	1	0	Alice	1
1	0	0	1	0	Bob	2
0	1	0	0	1	Bob	2
0	1	0	0	1	Bob	2
0	0	1	1	0	Claire	1
0	0	1	1	0	Claire	1

Table 33: Expanded example dataset similar to Table 24 but with the group variable added

4.4.2.1 Sum-All

This approach sums all variables up per group, it does not take into account username. See Table 34.

GET|/uc/clients| main_address|	***GET|/uc/employees| main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Group**
2	0	3	5	0	1
1	1	0	0	2	2

Table 34: Example sum-all approach.

It can be seen that this table counts the amount of occurrences of each variable per group. This table is further refined by dividing every variable by the total amount of variables in the same group. See figure Table 25 for reference on how groups are selected. The reason for doing this is to attain insight in how common the variables are for a group. The results can be seen in Table 35.

GET|/uc/clients| main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Group**
40%	0	60%	100%	0%	1
33.33%	66.67%	0	33.33%	66.67%	2

Table 35: The final result of the sum-all approach.

Variables with a high percentage could be core variables for its associated group. A problem with this approach however is that if one users in a group files a large amount of the same requests, the results might get skewed, since other users are overshadowed. The following approach does not have this problem.

4.4.2.2 Bin-all

The Bin-all approach is completely binary. When using this approach the corresponding variables of requests made by one or more users are set to one. Values of variables that do not occur are set to zero.

GET|/uc/clients| main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Group**
1	0	1	1	0	1
1	1	0	0	1	2

Table 36: The example dataset transformed using the bin-all approach.

Using the same calculation as before to calculate the percentages. It yields the following results.

GET|/uc/clients| main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients| main_address***	**Dossier**	**Employee-portal**	**Group**
50%	0%	50%	100%	0	1
50%	50%	0	50%	50%	2

Table 37: The final result of the bin-all approach

This however does not take into account the number of users that make a certain request. When using this approach a request filed by only one user is just as significant as a request made by 100 different users.

4.4.2.3 Bin-user

This approach is a combination of the last two approaches. All request are combined in a binary manner per user and are then added up per group. Using Table 33 as input, the first step can be seen

in Table 38.

GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Username**	**Group**
1	0	1	1	0	Alice	1
1	1	0	1	1	Bob	2
0	0	1	1	0	Claire	1

Table 38: The first step in the bin-user approach

In the second step the variables will be summed together per group. In this manner it becomes clear how many users made a certain request per group.

GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Group**
1	0	2	2	0	1
1	1	0	1	1	2

Table 39: The variables aggregated per user.

These variables are divided by the total amount of request made in that category.This results in the following figure.

GET|/uc/clients|main_address|	***GET|/uc/employees|main_address***	***POST|/uc/clients|main_address***	**Dossier**	**Employee-portal**	**Group**
33.33%	0	66.67%	100%	0	1
50%	50%	0	50%	50%	2

Table 40: The final result of the bin-user approach

This last methods is advantageous because it reduces the influence of singular users and gives an insight in how many different users in a group make a type of request. This is useful information for indentifying patterns.

The pattern finding approach results in a table of information about all variables, analysing this information is a time consuming process and this must be done sample-wise.

4.4.3 Detect Outliers

The detect outliers process uses the output of the preprocessing process to generate outliers. The goal of all the preceding process is that the input of this process consist of data that belongs to users with similar behaviour. Any behaviour that deviates from the norm should be detected by the outlier-detection algorithm. As such detected outliers should warrant further investigation. The output of this process are outliers that can be linked to one or multiple requests and the user who made that request. The following subchapter will describe some the outlier detection algorithms that were used.

4.4.4 Isolation forest

The isolation forest is an algorithm that generates outliers by using a decision (‘sklearn.ensemble.IsolationForest — scikit-learn 0.19.1 documentation’, 2018). It also assigns an anomaly score to every outlier it finds. This score gives insight in whether the request is very deviant or barely outside the normal spectrum. This anomaly scores allows for the filtering of outliers by setting a minimal score specified by the users. Some potential false positives can be

filtered out of the collection of outliers that is send to the user for further investigation.

The input used for the algorithm is the output of the preprocessing process, which is a collection of users with similar job descriptions. The user can specify two formats for input. The first format is a dissimilarity matrix, this gives a more broad overview that allows for a comparison of users. The second format consists of all the encoded requests made by users in the group. This allows for detecting specific requests as outliers. Which user made the request is then not taken into account. This results in two types of outputs. The first format results in users who are deemed outliers, while the second format results in request which are deemed as outliers. In both cases however anomaly scores are returned. These scores can be used to filter the outliers and only return the truly interesting outliers.

4.4.5 Birch

Birch is also an hierarchical clustering method. its advantages lies in its memory efficiency (‘sklearn.cluster.Birch — scikit-learn 0.19.1 documentation’, 2018). Birch was mainly selected to use as an outlier detection algorithm to work with SubjectID data since that provided to be very demanding when used as input. There were a lot of subjects associated with each user. This resulted in a large amount of variables per user when encoding the data.
Normal encoded data was given as input and the dissimilarity matrix was not used. Since Birch is supposed to be memory efficient, it should be able handle large amounts of variables. The user can also specify some extra variables that determine how the algorithm functions. These variables are the branching factor and the threshold. The amount of clusters can be specified, but when this is not done the algorithm returns the amount of sub clusters it found.
These sub-clusters serve as interesting results for outlier-detection. Large sub-clusters represent behaviour that is relatively normal and small sub-clusters represent behaviour that is relatively abnormal. The request that make up the small sub-clusters can be analysed further to see if any request merit further investigation.

The outlier detector application consists of two processes the preprocessor process and the detect outlier process. The application uses an user-grouping and data from the dataset to analyse a collection of request of similar users. After the data has been preprocessed an outlier-detection algorithm is used to detect outliers. This prototype makes use of the Isolation-forest algorithm and the Birch algorithm. The pattern-finding approach is an alternative way of analysing the data that does not make use outlier-detection algorithms. A lot of options are available for finding outliers using the outlier-detector application.

With the outlier-detection application all major components of the prototype have been discussed. In the appendix a link to the source code and a description of which tools were used can be found. The output of the prototype has to be evaluated by experts. A security expert should judge if a request that has been deemed an outlier warrants further investigation. During this thesis many experiments where done using a variety of approaches and settings, the most interesting of these results have been presented to the experts of Nedap. The next chapter will discuss the results generated by the prototype.

5 Evaluation

The output of the prototype will be evaluated in this chapter. The output of both the grouping-maker application and the outlier-detection application will be discussed, since the second application uses input from the first application. This evaluation will be done by discussing the observations made during experimentation and presenting scenario's of outliers that were discussed with experts. Since this is a proof of concept, terms such as efficiency will not be discussed. The prototype could definitely be optimized and different hardware could be used that influences those factors greatly. This evaluation will discuss whether results where found that signify that this methodology can be used to find deviant requests in real-life data. Subchapter 5.1 will focus on evaluating the results generated by the user-grouping application. After this the results generated by the outlier-detection application and the prototype as whole will be discussed.

5.1 User-groupings

The purpose of the grouping-maker application is to generate user-groupings. Hence, it will be evaluated on its ability to generate sensible groupings. A lot of trial and error was involved in adjusting the settings of the preprocessing process and the algorithms, therefore only a few groupings will be discussed in this thesis. The application will be evaluated using scenarios about groupings that have been discussed with experts.
A preliminary check was done to determine if a grouping was worth discussing with experts. This check was done by comparing the grouping to a list provided by Nedap. This list linked every user to a role specified by an information officer as part of identity and access management. These roles have been used in lieu of a job description. A bar chart was made using the groupings and the roles to visualize the results. These visualizations could then sample-wise be discussed with experts.

5.1.1 Scenario 1

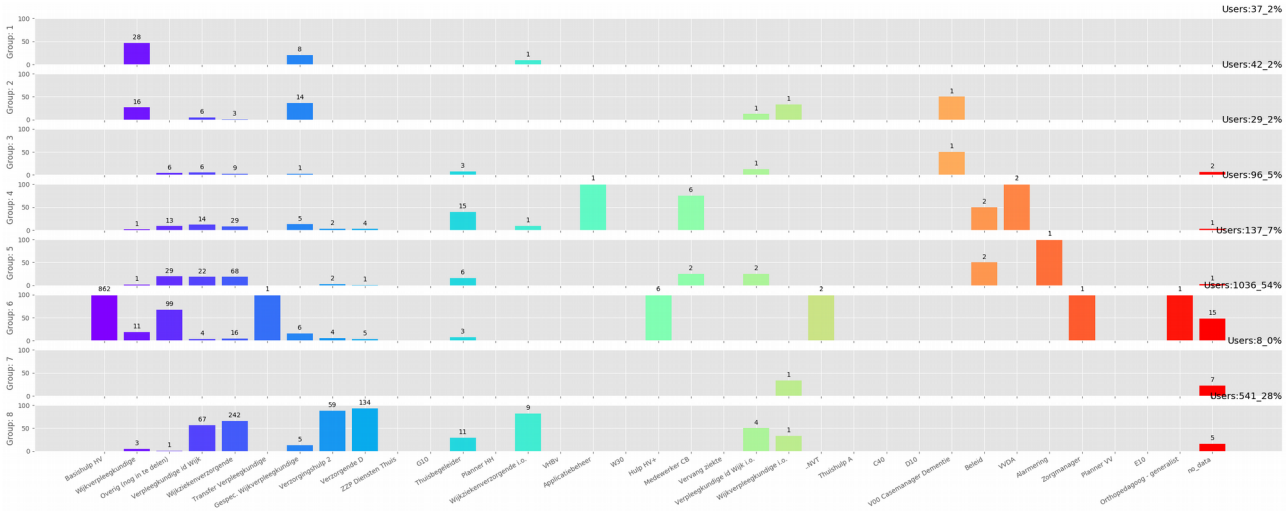

Illustration 5: Visualisation of a grouping generated by using the ACL algorithm.

Illustration 5 Shows a bar chart made using an groupings and the roles list. Users have been assigned to 8 clusters/groups. Each group is represented by a separate bar-chart. The left y-axis shows per bar-chart the percentage of the total users with a role in that group. The number on each bar shows the amount of users with the associated role in that group. For example group six contains 100% of users with the role "basishulp HV", which amounts to 862 users.
The right y-axis shows per bar-chart the amount of users in that group and after the underscore a percentage is given that represents that represent how much of the total amount of users have been allocated to each group.

As can be seen in Illustration 5 groups 6 and 8 consist of more then 80% of the users. This is partly because some roles are associated to more users then others, the role “basishulp HV” is a good example of this. However it also can be seen that groups have multiple roles assigned to them. The most optimal situation would be that one group consisted of users associated to one role only. This is not the case however and when increasing the amount of clusters this spread of roles did not decrease.
When looking at the role “wijkverpleegkundige” it can be seen that it is spread across multiple groups, this can also be seen when looking at some other roles. Some groups are clustered together successfully however and after discussing this grouping with experts some things became apparent. Users associated to different roles could behave very similar behaviours, this increases to difficulty for an algorithm to associated them to different clusters. Another more disparaging observation was however that some users with same role made very different requests to the system and had limited overlap.
This reveals however that the algorithm is able to distinguish between users based on their requests and assign users which are supposed to have similar behaviour to the same group. Real world data was being used however and some problems such as users with the same role having different behaviour appeared however. To solve these problems more research is required but the concept of grouping people based on requests has shown potential.

5.1.2 Scenario 2

Scenario 1 showed that users with different functions could be assigned to the same role and this caused difficulty for the system. This scenario however shows that users with the same role and the same function can still be assigned differently. In Illustration 6 shows a split in how the role “Basishulp HV” is being clustered, further investigation revealed an increase in request to the

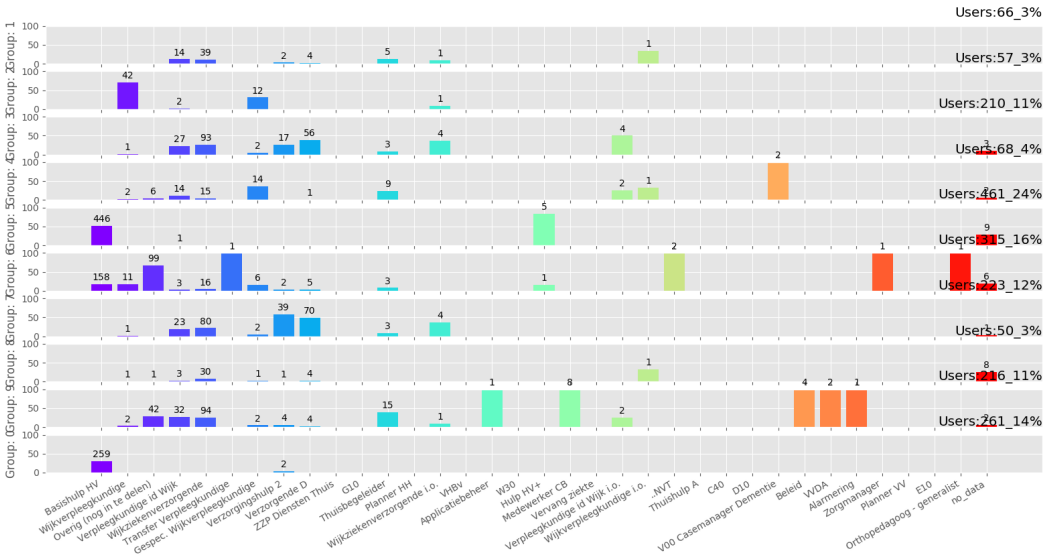

Illustration 6: Bar-chart of grouping made using K-means

work-hour administration system on Friday. It turned out that these employees all filled in their hours at Friday at half past seven in the evening and the rest if the user filled in their hours more spread over the week. The experts recognized this behaviour, since it had been noticed by other systems in the past. This behaviour occurred because a large amount of the employees watch a popular soap show on Friday evening and they file their work-hours to the system during the break of that show.
This shows that the system can pick up behaviour of users that has limited relevance but can

influence the quality of the grouping. On one hand this is good because it shows some sensitivity to changes in behaviour of users, but on the other hand this reveals another problem of working with real-life data.

As can be seen clustering on request behaviour garnered some success, but in chapter 4.3.1.1 an alternative approach was discussed. This grouping was based on the hypothesis that users could be clustered per geographical neighbourhood. This neighbourhood would be indentified by comparing the subjects users requested information about. If there is a large overlap of subjects between users that would mean these people worked in the same area. These groups based on neighbourhood could then be used for outlier-detection. Since if only one user looks at an user outside his area, this could signify he looked at information of a subject he had nothing do with. The Birch algorithm was used since the input generated by the id-approach contains a large amount of different clients. This method proved ineffective unfortunately. It became clear in discussions with experts that that the amount of user overlap was very limited and user could be associated to a large amount of unique subjects. Because of this no sensible groupings could be made and the approach was discarded.

The evaluation of the user-grouping application shows that the ability to group users based on request behaviour is possible. Real-world data creates some problems however and more research is required to evaluate the implications these problems have and how they should be fixed.

5.2 Outlier-detection

The outlier-detection algorithm is judged on its ability to identify outliers that point to deviant behaviour of users. Since this depends on the obligations and right of the users this has to be judged by experts. The Outlier-detection application was evaluated with two different types of groupings as input. The first type of groupings are the output generated by the grouping-maker application and the second type are groupings based on the role-list described in the previous chapter. This way the outlier-detection algorithm could be tested without the influence of the other applications in the prototype. Experts were consulted to evaluate the discovered outliers. It is dependent on the grouping supplied as input if an outlier constitutes deviant behaviour, since what is normal for one grouping is abnormal for another one. Since an outlier is a solitary instance of a request made by an user it can not be visualised in the same way the user-groupings cam. Scenario’s will still be used to elaborate on examples, but are not accompanied by a visual tool this time.

5.2.1 Scenario 3:

The isolation-forest algorithm was used in this scenario. The input the algorithm received was an encoded collection of request made users with the same role spanning a thirty day period. This resulted in a set of requests being judged as outliers. However when these requests were evaluated they were found to be normal behaviour. Various other settings also resulted in a lot of false positives and filtering on anomaly-scores yielded limited improvements.
After further evaluation it was concluded that there could be two reasons for these findings. The data is either so irregular that it is hard for the algorithm to define what is normal behaviour or there is no abnormal behaviour to find. Both these options have problematic implications regarding the thesis. The pattern-finding approach elaborated on in chapter 4.4.2 was used to look further in this problem.

5.2.2 Scenario 4:

The same data as in the previous example was provided as input for the pattern-finding approach. The output of this approach was used to create a table. This table consisted of uncommon requests made by members of a group. After manually evaluating the table the experts found a few interesting deviations. One such deviation was that in group that consisted of more then 100 users, 3

users made significantly different requests.
Further evaluation revealed that these users were caretakers that had completed additional training that enabled them to take care of patients the other users in their role weren't allowed to. This revealed that only three users made use of information that was made available to every user in the role.
This information could be used to improve information security by better defining the roles given to users. This weakens the assumption made in the conclusion of chapter 5.2.1, that there is no abnormal behaviour to be found. The misassignment of roles found here is exactly the kind of information that this thesis sets out to find.

It can be concluded that the pattern-finding technique shows that deviant outliers are present in the data. However the outlier-detection process left something to be desired since this information was not revealed by an algorithm but by a manual approach. The performance of the outlier-detection algorithms could likely stem from the irregularities found in the real-life data. To properly extract deviant behaviour from this kind of data using outlier-detection algorithms requires more research and experimentation.

6 Conclusion

The results of the prototype will now be used to make conclusion about the methodology. Referring back to the problem statement described in chapter 1.1. This thesis attempts to provide a methodology to analyse logs to support information security using artificial intelligence. The research questions described in chapter 1.2 are used to achieve this methodology. These research questions will discussed first, beginning with the sub-questions and using them to answer the main-question. After this the limitations of the research and recommendations for future work will be discussed.

6.1 How can system-logs be used to support information security?

To support information security information has to be extracted from the logs, doing this manually is not feasible because of the size of the logs. The focus was put on using data-mining techniques to automatically detect interesting information from the data. A literature review on outlier detection was done to discover a good approach. During the literature research a framework was created to categorize different outlier-detection approaches. After using that framework to classify different approaches it was concluded that a hybrid approach showed the most potential. A two-step method was developed using the findings of the survey as a foundation. First clustering techniques would be used to make user-groupings and then outlier-detection techniques would be used to detect outliers on these groupings. These outliers are then supplied to the experts. Chapter 3.2 demonstrates the method that has resulted from the research. This methodology shown in Illustration 3 is the answer that was arrived at for this research question.

6.2 What are the data-requirements that should be met by the logs?

The methodology only works on logs that meet certain requirements, these requirements have been developed and implemented on the logs provided by Nedap. The methodology also works with input that differs from the input that has been described in this thesis. Feature selection alludes to this, since every feature that has the option of not being selected is not critical for the analysing process. Any feature that fulfils that criteria would generate no problems if it was not present in the logs to begin with. To make the methodology as applicable as possible requirements that represent the very minimum of what is required for the methodology to work have been developed. The terminology described in chapter 3.1 is used here to differentiate to what is required of a log and what is required of every request in a log. The log is required to be reliable, it does not have to contain all the request that have been made to system, but if it contains one instance of a type of request it should contain every instance of that request. A log consist of requests and every request has the following critical requirements:

1. The request should be related to an user, for example the user-name is part of the log line.
2. One can differentiate between requests.

Meeting these requirements is enough for the methodology to work but its convenient when some optional requirements as described in detail in chapter 3.2.1 are met. The critical requirements might differ depending on organizational situation and what is expected from the result it produces, but these requirements can be used as the foundation for the methodology

6.3 How should an architecture that analyses logs be designed?

The architecture consist of three parts preparation, clustering, outlier-detection. Preparation is the task of transforming the data to an accessible format. After preparation the data should be accessible

and stored efficiently. The different variables of request should be easy to select and have a standardised format. Clustering as well as outlier-detection has been split up in a preprocessing task and an analysing task. The clustering part should result in user-groupings that are used as input during the preprocessing of the outlier detection part. Different algorithms have different requirements for accepting input and preprocessing is partly about fulfilling these requirements. Chapter 4 describes in detail what was used to design the prototype in this thesis, but there are other possibilities.

6.4 How does the methodology perform on data from the healthcare sector?

Chapter 5 is dedicated to evaluating the results that were generated using the prototype described in chapter 4. While not perfect the prototype was able to group users and provide sufficient groupings that could be used as input for outlier-detection. The results from the outlier detection were less conclusive. It became clear that the irregularities of real-life data create problems. An example is that the behaviour of users with the same role could still differ greatly from each other. The prototype showed that there is a possibility for this avenue of approach to work, but more fine-tuning is necessary to create more conclusive results.

6.5 How could one approach analysing system-logs to support information security using artificial-intelligence?

The method presented in chapter 3 and Illustration 3 prove to be interesting avenues of approach. The results of the prototype suggest that at least some patterns can be found following this methodology. The methodology presented in this paper is broadly applicable. As long as the requirements described in chapter 6.2 are fulfilled an input can be used. The focus on inter-application communication logs is an interesting avenue of approach, but this methodology is not limited to such input in anyway. As such this methodology has a good generalizability. Further research is however to refine the methodology. This thesis represent a step in the right direction to make this methodology useful in an organizational context. To encourage further research the limitations will now be discussed.

6.6 Limitations

While interesting discoveries have been made during the validation of the methodology, the validation still leaves something to be desired. It has been shown that some interesting results can be generated using this approach, but conclusive results have not been achieved. The limitations of this thesis could have caused this. The field of artificial intelligence has a lot to offer and is constantly improving. Not all options that the field had to offer have been explored sufficiently. Very interesting approaches and algorithms such as using a neural-net have not received sufficient attention. The complexity of the field and the variety of options required more time to explore than was available.
Another limitation proved to be the hardware. The data had to be simplified to be able to experiment with multiple algorithms since without these simplifications calculations would take days. A more appropriate setup would make different avenues of approach possible. While an interesting approach a big advantage of the dissimilarity matrix was the huge simplification of data.
Another limitation can be found in the dataset that was used to test the prototype. This data consisted purely out of inter-application communication. That means that activity that did not require communication with another application was not part of the dataset. A more inclusive logging apparatus like an audit-trail could generate different results.
It should also be noted that only three companies that use Ons have been analysed. It could be that

trough sheer bad luck, they were outliers that proofed to be very incompatible with this research and that companies with users that have a more homogeneous behaviour would generate different results.
Lastly it should also be noted that the role-list as described in chapter 5.1 that was used to evaluate the user-groupings might have influenced the evaluation. This role-list is made as part of identity and access management and is used to grant or restrict access to certain parts of the system. This means that a role is not truly representative of a job description. For example users with same role could have different job description. While evaluating the results there would have been limited ways of knowing this. As such the usage of a role-list in lieu of job descriptions might have influenced the evaluation.

6.7 Future-work

The limitations hint that a lot of work can still be done. There are a lot of algorithms available and this research only experimented with a small portion of them. Research that uses and judges different techniques should prove to be valuable. The Tenserflow library for examples offers interesting possibilities to use neural net and deep-learning techniques that could yield interesting results. It should also be mentioned that clients of different companies could be analysed together, this would make the result hard to evaluate since every company has specified different job description, the role-list also wouldn't work because of this different in specification. The employees that have similar job description and work at similar companies should roughly show the same behaviour. Analysing these users together could allow for the creation of global behaviour pattern for a certain type of job. This might prove to be an interesting avenue of approach for future work. Another options is to apply the methodology to a different dataset altogether. A more homogeneous dataset for example could yield different results. A dataset from companies outside the profession of care could also be used. As such there are a lot of options available for further research that could prove very interesting and the findings and considerations made in this thesis could be used as building-blocks for further work.

6.8 Recommendations for Nedap Healthcare

The evaluation was done using the data provided by Nedap Healthcare. The problems described during the evaluation are applicable on the logs generated by Ons. When Nedap Healthcare continues in exploring the possibilities their data has the offer. The source code of the prototype could be used as inspiration for their own application. If Nedap Healthcare decides to make an application using this methodology. The essence of the research can be summarised in 8 questions they should ask themselves. These questions are:

1. *“Can the company make use and act upon behavioural patterns and outliers?”*
2. *“Can I differentiate between users?”*
3. *“Can I differentiate between actions of the same users?”*
4. *“What should be done to prepare the data?”*
5. *“What techniques are applicable considering the data to make user-groupings?”*
6. *“What techniques are applicable considering the data to detect outliers?”*
7. *“What will be used to analyse the results?”*
8. *“Are the results sensible?”*

These questions are based on the results of the research questions. When each questions can be answered appropriately, some problems that occurred during this research should not be encountered.

7 Acknowledgements

I would like to thank my supervisors of the university of Twente for their support during the writing of this thesis. Maria Iacob's eye for quality and Maya Daneva's positive outlook and encouragement made this thesis a success.

I also would like to thank my supervisors of Nedap for their feedback and interesting ideas. I especially thank Joep Peeters for his willingness to always take time to discuss the current issue while creating the prototype. I thank Wout Slakhorst for his focus on keeping things practical and providing me with some valuable life lessons.

I would also like to thank Nedap Healthcare for providing me with the opportunity to make this research relevant in an organizational context. Their willingness to provide access to their logs and insight in Ons made this research a valuable experience.

I would like to thank Djurre de Jong for taking the time to explain and show me his work on membrane protein clustering. You provided me with some practical insights, that proved very useful in the beginning stages of creating the prototype.

Lastly I would like to thank Tim van Loo and Marc Groefsema from the university of Groningen. You provided me with new ideas, a fresh mindset and showed me the breathtaking possibilities in the field of artificial intelligence.

8 Literature

1. Agrawal, S., & Agrawal, J. (2015). Survey on Anomaly Detection using Data Mining Techniques. *Procedia Computer Science*, *60*, 708–713. https://doi.org/10.1016/j.procs.2015.08.220
2. Akoglu, L., Tong, H., & Koutra, D. (2015). Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, *29*(3), 626–688. https://doi.org/10.1007/s10618-014-0365-y
3. Aral, K. D., Guvenir, H. A., Sabuncuoglu, I., & Akar, A. R. (2012). A prescription fraud detection model. *Computer Methods and Programs in Biomedicine*, *106*(1), 37–46. https://doi.org/10.1016/j.cmpb.2011.09.003
4. Bouarfa, L., & Dankelman, J. (2012). Workflow mining and outlier detection from clinical activity logs. *Journal of Biomedical Informatics*, *45*(6), 1185–1190. https://doi.org/10.1016/j.jbi.2012.08.003
5. Campbell, K., Gordon, L. A., Loeb, M. P., & Zhou, L. (2003). The economic cost of publicly announced information security breaches: empirical evidence from the stock market. *Journal of Computer Security*, *11*(3), 431–448.
6. Casas, P., Mazel, J., & Owezarski, P. (2012). Unsupervised Network Intrusion Detection Systems: Detecting the Unknown without Knowledge. *Computer Communications*, *35*(7), 772–783. https://doi.org/10.1016/j.comcom.2012.01.016
7. Chandola, V., Banerjee, A., & Kumar, V. (2007). Outlier detection: A survey. *ACM Computing Surveys*. Retrieved from https://pdfs.semanticscholar.org/912b/0b7879ca99bf654a26bbb0d50d4b8e0ed6c0.pdf
8. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Comput. Surv.*, *41*(3), 15:1–15:58. https://doi.org/10.1145/1541880.1541882
9. Cho, S., & Cha, S. (2004). SAD: web session anomaly detection based on parameter estimation. *Computers & Security*, *23*(4), 312–319. https://doi.org/10.1016/j.cose.2004.01.006
10. Cho, S.-B., & Park, H.-J. (2003). Efficient anomaly detection by modeling privilege flows using hidden Markov model. *Computers & Security*, *22*(1), 45–55.

https://doi.org/10.1016/S0167-4048(03)00112-3

11. Dhillon, G. (2001). Violation of safeguards by trusted personnel and understanding related information security concerns. *Computers & Security*, *20*(2), 165–172.
12. Duan, L., Xu, L., Liu, Y., & Lee, J. (2009). Cluster-based outlier detection. *Annals of Operations Research*, *168*(1), 151–168. https://doi.org/10.1007/s10479-008-0371-9
13. Farrell, P. J., Groshen, S., MacGibbon, B., & Tomberlin, T. J. (2010). Outlier detection for a hierarchical Bayes model in a study of hospital variation in surgical procedures. *Statistical Methods in Medical Research*, *19*(6), 601–619. https://doi.org/10.1177/0962280209344926
14. Gunter, C., Liebovitz, D., & Malin, B. (2011). Experience-Based Access Management: A Life-Cycle Framework for Identity and Access Management Systems. *IEEE Security Privacy*, *9*(5), 48–55. https://doi.org/10.1109/MSP.2011.72
15. Han, J., & Kamber, M. (2010). *Data mining: concepts and techniques* (2. ed., [Nachdr.]). Amsterdam: Elsevier/Morgan Kaufmann.
16. Hauskrecht, M., Batal, I., Hong, C., Nguyen, Q., Cooper, G. F., Visweswaran, S., & Clermont, G. (2016). Outlier-based detection of unusual patient-management actions: An ICU study. *Journal of Biomedical Informatics*, *64*, 211–221. https://doi.org/10.1016/j.jbi.2016.10.002
17. Hodge, V., & Austin, J. (2004). A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, *22*(2), 85–126. https://doi.org/10.1023/B:AIRE.0000045502.10941.a9
18. Huang, Y., & Lee, W. (2003). A Cooperative Intrusion Detection System for Ad Hoc Networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks* (pp. 135–147). New York, NY, USA: ACM. https://doi.org/10.1145/986858.986877
19. Iacob, Maria Eugenia, Jonkers, H., Lankhorst, M.M., Proper, E., Quartel, Dick, & Industrial Engineering & Business Information Systems. (2012). *ArchiMate 2.0 Specification: The Open Group*. Van Haren Publishing. Retrieved from https://research.utwente.nl/en/publications/archimate-20-specification-the-open-group(51cb0851-e897-4c41-8a09-506b00add95c).html
20. Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing*

Surveys (CSUR), *31*(3), 264–323.

21. Jones, D., & Gregor, S. (2007). The Anatomy of a Design Theory. *Journal of the Association for Information Systems*, *8*(5), 1.
22. Joudaki, H., Rashidian, A., Minaei-Bidgoli, B., Mahmoodi, M., Geraili, B., Nasiri, M., & Arab, M. (2015). Using Data Mining to Detect Health Care Fraud and Abuse: A Review of Literature. *Global Journal of Health Science*, *7*(1), 194–202. https://doi.org/10.5539/gjhs.v7n1p194
23. Knorr, E. M., Ng, R. T., & Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The VLDB Journal*, *8*(3–4), 237–253. https://doi.org/10.1007/s007780050006
24. Kou, Y., Lu, C.-T., Sirwongwattana, S., & Huang, Y.-P. (2004). Survey of fraud detection techniques. In *IEEE International Conference on Networking, Sensing and Control, 2004* (Vol. 2, pp. 749–754 Vol.2). https://doi.org/10.1109/ICNSC.2004.1297040
25. Kuna, H. D., García-Martinez, R., & Villatoro, F. R. (2014). Outlier detection in audit logs for application systems. *Information Systems*, *44*, 22–33. https://doi.org/10.1016/j.is.2014.03.001
26. Lee, Y. J., Yeh, Y. R., & Wang, Y. C. F. (2013). Anomaly Detection via Online Oversampling Principal Component Analysis. *IEEE Transactions on Knowledge and Data Engineering*, *25*(7), 1460–1470. https://doi.org/10.1109/TKDE.2012.99
27. Leung, K., & Leckie, C. (2005). Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38* (pp. 333–342). Darlinghurst, Australia, Australia: Australian Computer Society, Inc. Retrieved from http://dl.acm.org/citation.cfm?id=1082161.1082198
28. Münz, G., Li, S., & Carle, G. (2007). Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*. Retrieved from https://pdfs.semanticscholar.org/634e/2f1a20755e7ab18e8e8094f48e140a32dacd.pdf
29. Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, *50*(3), 559–569. https://doi.org/10.1016/j.dss.2010.08.006

30. Perdisci, R., Gu, G., & Lee, W. (2006). Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. In *Sixth International Conference on Data Mining (ICDM'06)* (pp. 488–498). https://doi.org/10.1109/ICDM.2006.165
31. Petrovskiy, M. I. (2003). Outlier Detection Algorithms in Data Mining Systems. *Programming and Computer Software*, *29*(4), 228–237. https://doi.org/10.1023/A:1024974810270
32. Phua, C., Lee, V., Smith, K., & Gayler, R. (2012). A Comprehensive Survey of Data Mining-based Fraud Detection Research. *Computers in Human Behavior*, *28*(3), 1002–1013. https://doi.org/10.1016/j.chb.2012.01.002
33. Pokrajac, D., Reljin, N., Pejcic, N., & Lazarevic, A. (2008). Incremental Connectivity-Based Outlier Factor Algorithm. In *BCS Int. Acad. Conf.* (pp. 211–224). Retrieved from http://www.bcs.org/upload/pdf/ewic_vs08_s6paper4.pdf
34. Raj, S. B. E., & Portia, A. A. (2011). Analysis on credit card fraud detection methods. In *2011 International Conference on Computer, Communication and Electrical Technology (ICCCET)* (pp. 152–156). https://doi.org/10.1109/ICCCET.2011.5762457
35. Sarasamma, S. T., Zhu, Q. A., & Huff, J. (2005). Hierarchical Kohonenen net for anomaly detection in network security. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *35*(2), 302–312. https://doi.org/10.1109/TSMCB.2005.843274
36. Sharma, A., & Panigrahi, P. K. (2013). A review of financial accounting fraud detection based on data mining techniques. *ArXiv Preprint ArXiv:1309.3944*. Retrieved from https://arxiv.org/abs/1309.3944
37. sklearn.cluster.AgglomerativeClustering — scikit-learn 0.19.1 documentation. (2018). Retrieved 3 February 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering
38. sklearn.cluster.Birch — scikit-learn 0.19.1 documentation. (2018). Retrieved 3 February 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html#sklearn.cluster.Birch
39. sklearn.cluster.KMeans — scikit-learn 0.19.1 documentation. (2018). Retrieved 2 February

2018, from http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

40. sklearn.ensemble.IsolationForest — scikit-learn 0.19.1 documentation. (2018). Retrieved 3 February 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html
41. sklearn.preprocessing.MinMaxScaler — scikit-learn 0.19.1 documentation. (2018). Retrieved 2 February 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
42. sklearn.preprocessing.normalize — scikit-learn 0.19.1 documentation. (2018). Retrieved 30 January 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html
43. sklearn.preprocessing.scale — scikit-learn 0.19.1 documentation. (2018). Retrieved 2 February 2018, from http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html#sklearn.preprocessing.scale
44. Sreevidya, S. S., & others. (2014). A Survey on Outlier Detection Methods. *IJCSIT) International Journal of Computer Science and Information Technologies*, *5*(6). Retrieved from https://pdfs.semanticscholar.org/759d/21394435103ca62d23955f9bb99edda5e878.pdf
45. Takeuchi, J., & Yamanishi, K. (2006). A unifying framework for detecting outliers and change points from time series. *IEEE Transactions on Knowledge and Data Engineering*, *18*(4), 482–492. https://doi.org/10.1109/TKDE.2006.1599387
46. Tang, J., Chen, Z., Fu, A. W., & Cheung, D. W. (2007). Capabilities of outlier detection schemes in large datasets, framework and methodologies. *Knowledge and Information Systems*, *11*(1), 45–84. https://doi.org/10.1007/s10115-005-0233-6
47. Tomar, D., & Agarwal, S. (2013). A survey on Data Mining approaches for Healthcare. *International Journal of Bio-Science and Bio-Technology*, *5*(5), 241–266. https://doi.org/10.14257/ijbsbt.2013.5.5.25
48. Wang, S. (2010). A Comprehensive Survey of Data Mining-Based Accounting-Fraud Detection Research. In *2010 International Conference on Intelligent Computation Technology and Automation* (Vol. 1, pp. 50–53). https://doi.org/10.1109/ICICTA.2010.831

49. Witty, R. J., Allan, A., Enck, J., & Wagner, R. (2003). Identity and access management defined. *Research Study SPA-21-3430, Gartner*. Retrieved from http://www.bus.umich.edu/KresgePublic/Journals/Gartner/research/118200/118281/118281.pdf
50. Yue, D., Wu, X., Wang, Y., Li, Y., & Chu, C. H. (2007). A Review of Data Mining-Based Financial Fraud Detection Research. In *2007 International Conference on Wireless Communications, Networking and Mobile Computing* (pp. 5519–5522). https://doi.org/10.1109/WICOM.2007.1352
51. Zanero, S., & Savaresi, S. M. (2004). Unsupervised Learning Techniques for an Intrusion Detection System. In *Proceedings of the 2004 ACM Symposium on Applied Computing* (pp. 412–419). New York, NY, USA: ACM. https://doi.org/10.1145/967900.967988

9 Appendix

From a practical perspective some mention of what tools have been used during the creation of the prototype will now be made. While not very relevant for the thesis, this could be useful to anyone that wants to make use of the methodology to create a similar prototype. The source code can be found at: https://github.com/RJMdeGroot92/Hybrid_Outlier_Detection
During the preparation phase regular expressions were used to filter the logs. The data was stored between phases in gzipped .CSV files. The whole application was written in python. During the processes the pandas library was used to store the data. During preprocessing the Numpy and Scipy were used. The clustering or outlier-detection algorithms were retrieved from Scipy or Sklearn.

- Python was used as the programming language since it has a lot of libraries that support data-mining. Libraries I used were:
  - Pandas
  - Numpy
  - Scipy
  - Sklearn
- Regular expression were used in the Filtering & cleaning phase to handle exceptions in the log-lines.
- After filtering & cleaning the dataset was stored as CSV files.