Using machine learning for detecting anomaly accesses to EHRs: a comparative analysis and case study

By
Bart Belleman

In partial fulfillment
Of the requirements for the degree
Master of Science in Artificial Intelligence

Utrecht University

Graduate School of Natural Sciences
University of Utrecht
The Netherlands
October 2020

Using machine learning for detecting anomaly accesses to EHRs: a comparative analysis and case study

By
Bart Belleman

Abstract

The Dutch Data Protection Authority (AP) demands an intelligent and proactive way to find internal data breaches where patient records are deliberately and maliciously accessed by an employee outside the employee's job requirement.

This thesis compares a set of unsupervised machine learning approaches in their ability to find anomalous accesses, outliers, in Electronic Health Record (EHR) logging data.

Multiple machine learning methods, based on literature, are discussed first. Then, from those machine learning methods are the Local Outlier Factor, One Class Classification (OCC), Isolation Forest (IF), and Self-Organising Map selected to be trained on two control datasets.

Based on the performance on the two control datasets are the OCC and IF model applied to EHR logging data. Use cases are created based on the results from the OCC and IF model. Those use cases are discussed with several employees from the hospital.

The validation of the results is difficult due to the need for excessive contextual information, domain knowledge, and foremost a lack of a hospital-wide policy or guideline. We have proven with the control datasets that the outlier detection models can detect internal data-breaches, but without a policy or guideline, we cannot use these outlier detection models effectively in practice. Nevertheless, we found already interesting cases that would never have been found without these outlier detection models and in that way closer to compliance with the demands of the AP.

Contents

1 Introduction

Personal health records are used by medical experts to provide the right care for a patient. A personal health record contains all the information relevant to a specific patient, such as current and historical health, tests outcome, and referrals. Whereas this record used to be paper-based, there is a worldwide shift to the electronic version, the Electronic Health Record (EHR). The shift from paper- to electronic records has led to better communication with patients and as well among practitioners, the ability to remotely access the patient information, streamlining of work-flow processes, and more [60]. Unfortunately, this shift creates challenges too. The main challenge is to ensure patient privacy and confidentiality, within this information sharing focused environment of the EHR, by protecting the patient's data against data breaches.

Data breaches occur when patient data is inappropriately accessed. Breaches can be from the outside, such as hackers, but from the inside too [4, 16]. The most frequent breach from the inside is theft followed by inappropriate access to patient data [19].

Inappropriate accesses are data breaches from the inside that occur when hospital staff who deliberately and maliciously access patient records outside of their job requirements [19]. A hospital employee is looking at patient records not because it is needed for them to do their job, but out of personal interest. Most of the inappropriate accesses are staff reading family members' charts, neighbors' charts, co-workers' profiles, deceased patients' charts, sensitive/VIP patients' charts [7, 59, 4]. This thesis is focusing on detecting the inappropriate accesses from the inside.

Existing approaches to detect inappropriate accesses are to use access control schemes. Access control schemes help determine how resources, i.e. records, in a system are made available to users. Access control can be rule-based, task-based, or team-based [15]. It is difficult to implement a valid access control policy in hospitals due to several reasons [7, 59], such as dynamic workflows and unpredictable care patterns.

Because of these reasons and the need for quick access to data, hospitals often implement a "break the glass" (BTG) procedure. This procedure allows staff to access data they normally are not permitted to. The BTG procedure is mostly used for appropriate behavior such as peer to peer consultations, vacation/break coverage, accidentally accessing the wrong patient, outside hospital access [59, 7, 4].

A common pitfall for the BTG procedure is that the broad-based privileges can be misused for inappropriate accesses [7, 59, 4]. Therefore, a log is kept with who, when, and sometimes why a staff member uses the BTG procedure. This results in a log file containing both records from appropriate and inappropriate use of this procedure. This log file can be used to detect inappropriate assesses to patient records. However, in some hospitals, the security mechanisms, such as the BTG procedure, are used up to 54% of the time to access a record [4]. A security mechanism used that much creates a large log file. This large log file makes it difficult to separate the inappropriate uses from the appropriate ones. Thus, the log file of the BTG procedure can not be used (alone) for inappropriate access detection.

Currently, inappropriate access detection is done manually by sampling logging

data or in retrospect when there is a reason to believe an inappropriate access has occurred. Limitation of this manually auditing are [7]:

- The volume of the log record. A month of logging data contains around 9.000.000 rows. The chance of finding an inappropriate access by sampling is next to nil.
- Log record only contains information about the access itself, there is no situational or related information or knowledge regarding the access itself.
- Manually auditing is labor-intensive.

A recent event in a Dutch hospital, where a Dutch reality star was hospitalized and dozens of staff members inappropriately viewed her record, has led to an investigation from the Dutch Data Protection Authority (Autoriteit Persoongegevens AP) [56]. The authority concluded that the current measure of sampling is not sufficient to protect the patients' privacy. Hospitals need to implement a proactive system to detect internal data breaches.

Logging data can be used to proactively detect internal data breaches by searching for accesses that stand out from normal activity, also called anomalies or outliers. Logging data has been used in a healthcare setting to update access control schemes, but not for outlier detection [5]. Outlier detection can be done through machine learning and is already used in the medical field for different purposes [3, 58]. There are different kind of machine learning models. It depends on the data which machine learning model is required. Logging data is unlabeled and, therefore, asks for unsupervised learning models. Unsupervised models are already used to detect data breaches using logging data in a network or database setting [25, 35, 6][40, 18]. Chen et al. have successfully used unsupervised machine learning methods to detect anomalous users in an EHR, and Boddy et al. show even the potential of using logging data to detect inappropriate accesses to patient records [7].

1.1 Research question

Because of the promising results of machine learning-based outlier detection in different disciplines, this thesis aims to explore a set of machine learning algorithms and compare their performance on anomalous access detection. This thesis tries to answer the research question: *given a set of unlabeled logging accesses to the electronic patient records of a hospital, which machine learning model performs the best to detect anomaly accesses to the patient records in terms of the trade-off between being as precise as possible and detecting all anomalies.*

We divide this research question into two sub-questions. Where each question is answered with a different dataset. The two sub-questions are:

- *How accurate are machine learning models in detecting anomalies in terms of the trade-off between being as precise as possible and detecting all anomalies.*
- *To what extend can the machine learning models detect anomalous accesses when applied to real EHR logging data.*

A control dataset is used to answer the first sub-question. The real EHR logging data is used to answer the second sub-question. The same machine learning models

are used on the both dataset. However the control dataset is labeled, whereas the real data is completely unlabeled. The labeled control dataset is only used to evaluate the performance of the machine learning models.

1.2 Structure thesis

An overview of the structure of the thesis can be found in Figure 1. This introduction is followed by section two, the background, and section three, the research method. After the method section, the different datasets are explained in section four. Then there is a clear separation between the control datasets (labeled) in section five and the case study (unlabeled real-life EHR logging data) in section six. Both section five and section six are organised in the same way. We start by explaining how we evaluate the control dataset, show the results, and end both sections with a discussion. The gained knowledge from the control datasets serves as input for the case study. We end this thesis with section seven, the conclusion.

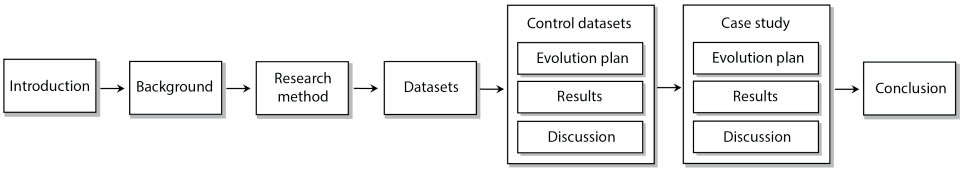

Figure 1: Structure thesis.

2 Background

In this section, we first discuss the reasons why access control schemes are difficult to implement in a hospital setting and the use of logging data to make a more fitting representation of the real workflow by updating the access control schemes in section 2.1. Then, we examine different purposes of outlier detection within the medical field, followed by an explanation of why unsupervised machine learning models are suitable for outlier detection in section 2.2. We give some examples of unsupervised machine learning models used to find network intrusions in section 2.3. Then, we list some reasons why the evaluation of unsupervised machine learning models is difficult in section 2.4. We end with a summary in section 2.5

2.1 Access control and logging data

It is difficult to implement a valid access control scheme in a hospital setting due to several reasons, [7, 59], those reasons include:

- The unpredictable and dynamic care patterns, including scheduled and unscheduled inpatient, outpatient and emergency department visits.
- The varied workflows, with providers requiring access in unexpected areas. The workflows vary to adjust to demand fluctuation or disastrous situations.
- A mobile workforce, with access required at unexpected locations and times.
- The collaborative nature of clinical work and teaching environments.
- A large number of users with varied job titles and roles.
- The user's job titles are not directly relating to a list of patients whose records it would be appropriate to access.

Bhatti et al. used logging data from the EHR to find a representation of those complicated workflows and update the access control schemes accordingly [5]. An initial ideal workflow is made beforehand and a model is used to find the gap between the initial and real workflow. This model uses logging data for policy refinement by updating the access control schemes over time. The model searches for reoccurring reasons for activation of the BTG procedure, for example a nurse needs to access referral data for a registration purpose, but the policy allows the use of such data only for treatment purposes. With high enough coverage, this reason becomes a new privacy policy. The access control scheme is updated and it is not necessary to use the BTG procedure for this particular reason in the future [5].

The advantage of using logging data is that the data fits the clinical workflow and does not require the workflow to fit the data [5]. This means that by using logging data the access control scheme will gradually follow the clinical workflow. Without logging data, it may be possible that hospital staff needs to change their workflow to follow the access scheme. Updating privacy policies will eventually lower the need for the BTG procedure, but will not proactively find inappropriate accesses. An alternative approach is to use the logging data to find outliers or anomalies within the data, i.e. the possible inappropriate accesses to patient records.

2.2 Outlier detection: supervised vs unsupervised machine learning

Outlier detection is already used for different purposes within the medical field. For instance, EHR data is used to detect outliers in the treatment of patients. If there is an outlier in the treatment of a large group of patients compared to the medical guideline, this outlier may suggest the need to update the guideline [3]. Another example is detecting healthcare fraud and abuse by observing outliers in clinical pathways drawn from EHR data [58]. While outlier detection models are already used in a hospital setting, outlier detection is not used for access control using logging data.

While anomalous access detection is relatively new in the medical field, anomalous access detection is already used for intrusion detection within network or database environments. For intrusion detection both supervised and unsupervised machine learning models are used [22, 13]. Supervised learning models are trained using labeled data, meaning that for the input the desired output is already known. However, the problem for logging data is labeling. Labeling is a time-consuming task for privacy officers. Each label needs a lengthy, in depth analysis. Furthermore, access violations are rare, making the dataset highly unbalanced. The last problem is the sample selection bias, privacy officers tend to provide labels for conspicuously inappropriate accesses, making it unrepresentative for the future [25, 36].

Unsupervised learning models do not require labeled datasets, making it more suitable for anomalous access detection than supervised learning models. Unsupervised learning models make use of the inherent structure of data to determine outliers. For example grouping similar data instances into clusters, while labeling the data points outside the clusters as outlier.

2.3 Unsupervised intrusion detection

Unsupervised outlier detection models make two assumptions about the data. The first assumption states that the number of appropriate accesses vastly outnumber the inappropriate ones. As the exact percentage of outliers varies per dataset, it is common for the outlier class to be around 5% of the whole dataset [11, 34]. The second assumption states that the inappropriate accesses are different from the appropriate ones. So following these two assumptions, the anomalous data points are rare and different from the other points. The outliers detection models use these characteristics to isolate the anomalous data points from the normal data points [18].

Unsupervised learning models trained on logging data are already used for intrusion detection. Clustering is used to show the usefulness of the outlier detection-based methodology when the access pattern of users deviates from the overall distribution of the normal access pattern [25]. Portnoy et al. used unsupervised clustering and were able to detect a large number of intrusions while keeping the false positives low [40]. The combination of neural nets with clustering is considered a great promise too [6]. The results from other unsupervised machine learning models, such as isolation forest, support vector machine, and k-nearest neighbors (kNN), are also seen as encouraging for intrusion detection [35, 18].

While using unsupervised learning on logging data for outlier detection is wildly

used for intrusion detection, there is still little known about detecting inappropriate accesses to patient records. However, Chen et al. have proposed an unsupervised community-based anomaly detection model to find anomalous users in collaborative information systems, like an EHR [15]. The proposed unsupervised model is a blend of kNN models and principal component analysis (PCA). This PCA model has the highest performance at detecting simulated insider threats compared to the kNN, PCA, and high volume model [15].

2.4 Difficulties in performance evaluation

However beneficial and easy unlabeled data is during the training phase, the unlabeled data creates difficulties for the evaluation in the long run. Without labeling the dataset it is difficult to measure the performance. Normally, performance measures such as accuracy, precision, recall, etc. can say something about how well the model can detect outliers. A solution is to use a benchmark set, an already labeled and publicly available dataset, that is close to the unlabeled dataset. This benchmark can serve as an indication of how the models perform on the unlabeled dataset and compare this performance to other models from other papers.

Unfortunately, there is a shortage of appropriate public benchmark datasets [11, 54]. The primary reason for this shortage of publicly available data arises from data's sensitive nature [54]. For example network traffic from a company can reveal highly confidential information such as personal communication, business secrets, or user's network access patterns [54]. Furthermore, the same holds for hospitals, EHR data is considered highly confidential. Without a benchmark dataset, it is difficult to compare different outlier detection models. So, Campos et al. have discussed the suitability of multiple publicly available datasets to serve as outlier detection benchmark dataset for anomaly detection [11].

2.5 Summary

We have discussed how logging data has been used to update access control schemes in a hospital, but not for outlier detection [5]. Outlier detection is a medical setting that is already used for other purposes than access control [3, 58]. Therefore, in this thesis, we use outlier detection to find anomalous accesses to patient records, because these outlier detection models are already successfully used to detect data breaches using logging data in a network or database environment [25, 35, 6][40, 18]. For the evaluation of the outlier detection models, we use a benchmark dataset proposed by Campos et al.

3 Datasets

In this section, we discuss the different datasets used, their characteristics, and the data preprocessing steps done for each dataset.

As mentioned before, we are using multiple datasets, two labeled control datasets, and a private unlabeled hospital dataset. An overview can be found in Table 1. The control datasets are used both for (i) an indication of the performance for the unlabeled set and (ii) to compare the performance to literature. To satisfy both reasons the control datasets that are chosen have to be comparable to the hospital dataset (i) and also be used in other papers (ii). At the end of section two of this thesis, it is addressed why finding a control dataset is difficult. Ideally, we find a dataset with logging data from a hospital, that is also used in other papers, but unfortunately we do not have access to such datasets. Therefore, we used other logging data. Network logging data is broadly used and in essence comparable to the hospital logging data. The two kinds of logging data, EHR and network data, are both based on the interaction of a person with a system.

Finding such a network dataset that is also used as a benchmark set is difficult. An effort is made to propose some datasets that are used by multiple papers to serve as a benchmark set for outlier detection models [11]. Our first control dataset is chosen from this proposed list of datasets. The KDDcup99 is the most often used dataset for outlier detection models and therefore the closest to a benchmark set. Our second control dataset is used to generalise and validate the results from the KDDcup99 and does not necessarily need to fullfill the task of a benchmark. The second dataset is the ISCXIDS2012 set and discussed after the KDDcup99 in this section. We end this section by describing the hospital data. For all three datasets, we explain the characteristics of this dataset and also the manipulations done before training.

Table 1: Overview datasets used.

Dataset	**Records**	**Features**	**Outliers (%)**
Control dataset 1 KDDcup99	4.848.431	41	80.1
Control dataset 1 10 percent KDDcup99	494.021	41	80.8
Control dataset 2 ISCXIDS2012	275.528	19	8.00
Case study EHR logging data single log	7.970.083	18	N/A

3.1 KDDcup99

The most used dataset for bench-marking outlier detection models is the KDDcup99 [11]. The KDDcup99 is the dataset used for The Third International Knowledge Discovery and Data Mining Tools Competition [27]. The KDDcup99 is derived from a DARPA dataset. The DARPA dataset contains multiple weeks of network activity from a simulated Air Force network from 1998 [54].

The papers mentioned by Campos et al. [11] are using the KDDcup99 dataset

to propose a unification of outlier scores from various outlier detection models [30]. Nguyen et al. used the KDDcup99 to introduce a feature extraction method for outlier detection [38] or to test the combination of multiple outlier detection models [37]. Schubert et al. mention a new framework for the evaluation of outlier detection models and uses the KDDcup99 to test different models on [43].

During the background section of this thesis, we discuss multiple papers that use datasets to train their models on. Of those papers, three papers (Bivens et al. [6], Portney et al. [40], and Eskin et al [18]) use a KDDcup99 dataset as benchmark. The other papers are using a private dataset. Three of those (Menon et al. [36], Kamra et al. [25], and Chen et al. [15]) are using a publicly available dataset beside their private dataset. With a publicly available dataset is it possible for others to recreate the research. Therefore, we choose a dataset such as the KDDcup99 for this paper.

While the DARPA dataset from 1999 is no longer adequate for modern intrusion detection, the DARPA dataset can still be used to compare outlier detection models. The DARPA dataset is not only old, but the way the data is generated, is outdated too. However, the purpose of the control dataset is to compare models to each other within this thesis and with literature, not to find the latest form of intrusions or attacks in a network. Therefore, we have chosen the DARPA based dataset as a control dataset. From the different kinds of DARPA based datasets we are using the original KDDcup99 and the ten percent version of this set.

The KDDcup99 dataset consists of approximately five million instances. The instances are collected from raw network data. The network data is collected while recreating a U.S. Air Force Local-Area Network (LAN) environment. Multiple attacks were simulated, while running the LAN as if the LAN is a true Air Force environment. Seven week of raw data is processed into around five million connection records [27]. Next to the whole set of five million records, there is also a ten percent version, with only around half a million records.

Each record is a connection. The connection is a sequence of Transmission Control Protocol (TCP) packets starting and ending at some well-defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. The given features are the basic features of an individual TCP, that are obtained using some domain knowledge, or are computed using a two-second time window. There are 41 features in total [27]. Each connection/record is labeled as either normal or as an attack, with exactly one specific attack type, with a total of 24 attack types [27].

3.2 ISCXIDS2012

The Intrusion detection evaluation dataset (ISCXIDS2012) is created to replace outdated and static datasets, such as the KDDcup99. Real network traces are analysed to create profiles for agents that generate real network traffic [48]. Various multi-stage attacks are carried out to create outliers in the dataset.

The complete dataset consists of multiple days of network activity. There are in total 7 days of network data available, each day with different attacks plus normal activity or only normal activity. The dataset size varies between different days varies, from 3.95GB to 23.4GB. We have chosen the least amount of data for time saving and because this dataset functions as second control dataset. The dataset we

have chosen is the dataset of Sunday, 13/06/2012. This day consists of infiltrations from the inside and normal network activity.

The dataset consists of 275.528 rows with 19 features, excluding the label. Each row represents a network package payload. The features contain information about this payload. There are 255.170 normal payloads, the inliers, and 20.358 payloads from attacks on this network, the outliers.

3.3 Case study

The case study includes EHR logging data from a general hospital located in the Netherlands. This hospital treats around 9000 day-care patients, 60.000 out-patients, and 11.000 new admissions a year [23]. The hospital logging data date back to October 2019, the moment the hospital switched to a new EHR system. We explain next the structure of the logging database: the tables we use, and the relationship between the tables. Then, we discuss the features we use and the preprocessing steps needed before training.

3.3.1 Tables

A schematic representation of the database used can be seen in Figure 2. This database conaints the EHR logging data. The three tables are discussed next.

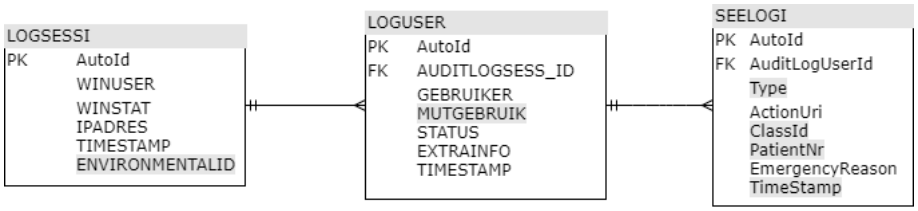

Figure 2: Database tables and relationships. Private Key (PK) and Foreign Key (FK). Used columns are highlighted grey

The LOGSESSI table holds all the log sessions. A log session is created after an EHR instance is created, i.e. opening the EHR on a device, and the session is ended when this instance is closed again. The columns that can be used from the LOGSESSI table are:

- AutoId: private key.
- WINUSER: refers to the login group from the EHR.
- WINSTAT: refers to a windows station that is logged in.
- IPADRES: refers to the IP address from the windows station.
- TIMESTAMP: refers to date and time of the moment a log session is started.
- ENVIRONMENTID: refers to the ID of the environment used.

The LOGUSER holds the information about a single user of the EHR. A new user ID (LOGUSER.AutoId) is made when a user logs into the EHR. In theory it is possible for multiple users to use the same log session, user one logs off and user two logs in without closing the EHR instance. However, in practise we see a one to one relationship, where the number of logsession are equal to the user IDs (LOGUSER.AutoId).The columns that can be used from LOGUSER are:

- AutoId: private key.
- GEBRUIKER: refers to EHR logging group.
- MUTGEBRUIK: refers to the EHR login user. This column is used to trace back the user.
- STATUS: status of the log user.
- EXTRAINFO: additional information about the log user.
- AUDITLOGSESS_ID: refers to a unique ID log session (foreign key).
- TIMESTAMP: refers to the date and time of the moment a log user is created.

The SEELOGI table holds the logs from the actions of a user within the EHR. The columns that can be used from AUDITLOG_SEELOGI are:

- AutoId: Private key.
- AuditLogUserId: foreign key.
- TYPE: states if the BTG procedure is used or not.
- ACTIONURI: refers to the type of action.
- CLASSID: refers to a class in the EHR.
- PATIENTNR: refers to the patient number.
- EMERGENCYREASON: refers to the reason, in free text, why the emergency procedure is used.
- TIMESTAMP: refers to data and time of the moment a log is written to the SEELOGI table.

3.3.2 Cardinality database

The relationships between the tables can be seen in Figure 2. As mentioned before, an one-to-many relationship may exist between the LOGSESSI table and the LOGUSER table, but in practice we see an one-to-one relationship. The cardinality between LOGUSER and SEELOGI is an one-to-many, where one user can perform multiple actions.

3.3.3 Feature selection

We do not use all the features from the three tables but make a selection based on experience. We exclude features that we think are abundant. Another reason to exclude a feature is that in practice the columns are not filled to a extend so that this feature can be used. From the first table, LOGSESSI, we use ENVIRONMENTID column to select a particular environment. The columns we use from the second table, LOGUSER, is MUTGEBRUIK. From the third table, we use the TYPE, CLASSID, PATIENTNR, and TIMESTAMP columns. The selected columns are highlighted in grey in Figure 2.

3.3.4 Data selection

We have access to logging data dating back to October 2019. We use the ENVIRONMENTID feature to differentiate between the design, test, acceptance and production environments of the EHR. From those environments, we only select the logs within production environment. A single month contains on average eight million log records. To keep it comprehensible and computationally feasible we select a single month, in particular from 18-07-2020 to 18-08-2020. Note that this month is both within the summer holiday and during the SARS-CoV-2 pandemic. The number of logs is usually lower during a holiday than during regular months. Also, we see by the number of logs that the normally regular months, during the pandemic, are close to the number of logs during holidays.

3.4 Data preprocessing

In this section, we discuss the three data preprocessing steps taken on the datasets. The first part of this section is focused on explaining the three preprocessing steps: downsizing, normalization and data transformation. In second, we explain how we perform each preprocessing step on the different datasets. We end this section with an overview of all datasets after the preprocessings steps.

3.4.1 Downsizing

One of the two assumptions about outliers in logging data, states that inliers vastly outnumber the outliers. If the ratio inlier/outlier in the control dataset is not correct, the ratio needs to be corrected by downsizing or the number of outliers present or the number of inliers present to match the right ratio.

For the control dataset, we set the ratio inlier/outlier to 99 to 1% as default in line with Eskin et al. and Portnoy et al. [18, 40]. We consider the ratio 99%/1% as standard, but as an addition, we examine the influence of the number of outliers on the performance, by also increasing the percentage of outliers.

3.4.2 Normalization

Attributes of a dataset can have different measurement units, with different scales. This difference in scales plays a role, especially with nearest neighbour based models. Some attributes may influence the nearest neighbour structure more than others, biasing the outcome. Campos et al. show that outlier detection models trained

on normalized data give higher performance rates [11]. There are multiple ways to scale the dataset, the four common ways are [26]:

- Minimum (Min) and maximum (Max) normalization.
- Mean and standard deviation (SD) normalization.
- Median and the interquartile range (IQR) normalization.
- Median and median absolute deviation (MAD) normalization.

For most models, including the LOF, on most datasets, the Min-Max and median-IQR outperforms the mean-SD and median-MAD methods [26]. Between the Min-Max and median-IQR normalization methods, the Min-Max method performs better than the median-IQR on most datasets [26].

3.4.3 Data transformation

In this section, we discuss the data transformation from categorical variables to numeric variables. Some tree-like models are able to handle categorical variables, but most models can not. There is no consensus about the best method to convert categorical variables into a numeric variable. The focus of this study is not to choose the best method, but we are discussing three methods proposed by literature [11, 26].

The first method is 1-of-n encoding, or one-hot encoding, where a categorical variable with n possible values is mapped into n binary attributes for which the value represents the presence (1) or absence (0) of the categorical variable [11, 26]. The downside of this method is the dimensions added to the dataset. An extra column is added for each possible value per categorical variable. The dimensionality can be a contributing factor to the performance of some models and increasing the dimensionality can have a negative influence [11]. Plus, the more dimensions the longer it takes to train a model.

The second method is the Inverse Data Frequency (IDF) method [11, 26]. IDF maps the rare values to higher numbers and common values to lower numbers. The more a category appears, the closer this category is mapped to other frequently occurring categories. A rare occurring category can be a sign of an outlier and therefore mapped to a higher number, further away from the more occurring categories. IDF is calculated by $IDF(x) = ln(N/n_x)$ where N is the total number of observations in the dataset and n_x is the number of times the categorical variable takes the value x [26]. However, problems can occur with categorical values with the same frequency. If the frequency of two categorical values is the same, those values are represented by the same number, but ideally those values should be relatively far apart.

The last method is to remove the categorical variables. Removing the categorical variables is possible when the variance of this variable is low enough, otherwise removing the variable can influence the performance negatively.

3.4.4 KDDcup99

Downsizing: The outliers are over-represented in the KDDcup99. There are even two kinds of attacks that have more instances than there are inliers in the dataset.

Thus, the amount of attacks needs to be downsized. The dataset is filtered such that the set consisted of 1% outliers and 99% inliers. These percentages are in line with other papers [18, 40]. The deleted attack instances are chosen randomly while keeping the same ratio between the different kinds of attacks.

Normalization/Transformation: There are three categorical features present in the KDDcup99: service, protocol_type, and flag. Together containing 45 unique categories. This amount of unique categories is an example of the downside of a one-hot encoder. If we are using a one-hot encoder instead of the IDF method, it results in 83 features $(41 - 3 + 45)$, 42 more than with the other methods. The categorical features are converted into numeric features through the later selected method (Appendix subsection A.1). Then all features are scaled using the Min/Max method.

3.4.5 ISCXIDS2012

Downsizing: The inlier/outlier ratio is originally 92%/8%. Thus, the number of outliers is downsized to fit the 99%/1% ratio. The deletion is done randomly.

Normalization/Transformation: There are 15 categorical variables. Four of them consists of over 50% of empty rows. Suggested by Campos et al. is to exclude features with more than 10% missing values [11]. However, excluding those features have a significant negative effect on the performance, a TNR/FNR AUC score of 0.20 lower, implying that the absence of this value for those feature holds information too. The categorical features are converted into numeric features through the later selected method (Appendix subsection A.1). Then all features are scaled using the Min/Max method.

3.4.6 Case study

The data preprocessing steps for the case study differs from the control datasets. Instead of downsizing, we perform feature engineering. Next to the data preprocessing steps we derive a different dataset from the logging data of the case study.

Feature engineering: We expect a difference between logs made during different parts of the day. Therefore, we use the TimeStamp feature. The time from Timestamp is measured to the millisecond, creating $1,15741e-8$ possible values (number of milliseconds each 24 hours). This scale unnecessary precise for the purpose of the study, so we convert the time into part-of-days. The time is converted into 'morning' (between 5am and 12pm), 'afternoon' (between 12pm and 18pm), 'evening' (between 18pm and 23pm), and 'night'.

Normalization/Transformation: All features are categorical features at this point. We convert these features into numeric variables through the selected method in Appendix subsection A.1. Then, we scale the numeric feature between zero and one with the Min/Max method.

Extra case study dataset: As the EHR also keeps track of sessions, we can use those sessions to find anomaly accesses to patient records too. Each session contains hundreds to thousands logs, so we have to summarize each session. We summarize each session by taking the number of users (unique MUTGEBRUIK), number of patients (unique PATIENTNR), part of day each session starts (derived from TIMESTAMP), use of BTG procudre (TYPE), and the sections of EHR where each record is accessed (CLASSID). Each session may contain different TYPE and CLASSID values, so to maintain the same dimensionality for each row (a row represents a single log session), we have to use the one-hot encoding. So, there are three columns for the three different TYPE values and the number of CLASSID columns is not fixed and depends on the unique CLASSID values used in the dataset. The one-hot encoding significantly increases the number of features, from 5 to 119 features, and thus the run-time. However, because each session represents at least a couple hundred log rows, so expect the decrease in the number of rows, from almost 8.000.000 to only 20.184 rows, to out weight the drawback of the extra dimensions.

3.4.7 Overview datasets

Table 2 shows the number of records, number of features and the percentage of outliers present for each dataset after the preprocessing steps taken.

Table 2: Overview datasets after data preprocessing.

Dataset	**Records**	**Features**	**Outliers (%)**
Control dataset 1 KDDcup99	982.508	41	1
Control dataset 1 10 percent KDDcup99	98.251	41	1
Control dataset 2 ISCXIDS2012	257.722	19	1
Case study EHR logging data single log	7.970.083	5	N/A
Case study EHR logging data log sessions	20.184	112	N/A

4 Research method

Figure 3 shows the steps needed from the plain dataset to the eventual results. The first two steps, datasets and data preprocessing, are discussed in the previous section. This section discusses the next three steps. First, we discuss the selection of the outlier detection models, followed by the selection of the packages to build those models. We end this section with how we decide on the parameters for those packages.

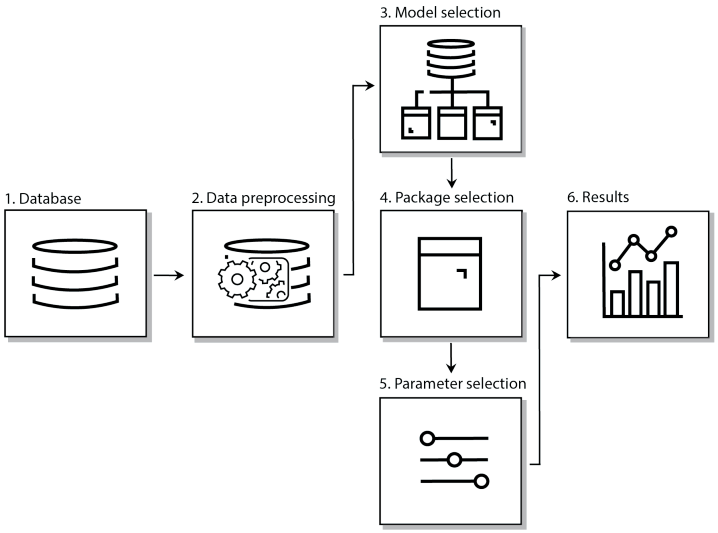

Figure 3: Flowchart from data to results.

4.1 Model selection

In this section, we discuss a set of machine learning models in their ability to detect anomalous access to patient records. Outlier detection methods are generally categorized in (i) density-based models, (ii) distance-based models, (iii) parametric-based models, (iv) graph-based models, and (v) neural net-based models.

A selection of models is made based on literature. The selected models are models that are commonly used. Next is discussed if the model is included for further evaluation. This discussion is based on the following criteria:

- **Data type:** whether the data is labeled or unlabeled. The EHR logging data is unlabeled, so a model needs to be able to handle unlabeled data.
- **The computational costs:** It has to be feasible to train the model on a large dataset within a reasonable time on the hardware used by the hospital.
- **Prior knowledge:** Some models need prior knowledge. It has to be possible to extract this knowledge from the data, based on a prediction from a privacy officer, other experts, or based on literature.

The ability to handle unlabeled data and prior knowledge needed are the most important criteria. If a model needs labeled data or knowledge that can not be provided, the model is excluded without considering the other criteria.

In the following section, we discuss the models used for intrusion detection as well as common unsupervised learning models. For each model, we provide a general description, the overall applicability based on the assessment criteria listed above, and its addition to the set of models. An overview of the assessment can be found in Table 3.

Table 3: Overview assessment outlier detection models.

Model	**Model type**	**Data type**	**Prior knowledge**	**Complexity**	**Included**
Local outlier factor	Density	Unlabeled	No	Polynomial	Yes
Collaborative filtering model	Distance	Labeled	N/A	N/A	No
k-Means clustering	Distance	Unlabeled	Number of clusters	N/A	No
One class classification	Parametric	Unlabeled	Percentage outliers	Depends on kernel function	Yes
Isolation forest	Graph based	Unlabeled	No	Linear	Yes
Hidden Markov model	Graph based	Unlabeled	No	Polynomial	Yes

4.1.1 Density based models

Local Outlier Factor (LOF): a LOF model determines whether or not a point lays in a spare region of the feature space by calculating the sum of distances to the k-nearest neighbors (kNN) of this point. A point in a dense region, an appropriate access, has a lower score compared to a point in a sparse region because the distance to the other points is smaller. The LOF model does not require labeled data or prior knowledge about the data. But, the LOF model can be computationally expensive. The model needs to compute the kNN score for each point [35, 18]. Despite the computational costs LOF model is included in this thesis.

4.1.2 Distance based models

Collaborative filtering model: the goal of a collaborative filtering model is to predict a label for the interaction of a pair of entities based on historical data. One entity of this pair has to contain a label. Because of the scarcity of labels within logging data, this model can only be used as an addition on a model that labels the data first [7, 36]. This model can be used as an improvement for detecting anomalous accesses, which can be interesting for further research. Therefore, the evaluation of collaborative filtering models is excluded in this thesis.

***k*-means clustering:** k-means clustering divides the dataset into multiple clusters, making each cluster as compact as possible [1]. The problem with k-means clustering lays in the knowledge needed beforehand. This prior knowledge is the prediction of the number of clusters, k. This aspect makes k-mean clustering less suitable for access control than other clustering models and thus excluded from further evaluation in this thesis.

4.1.3 Parametric based models

One-class classification Support Vector Machines (OCC): OCC models are unsupervised learning models that try to classify one class of objects and distinguish them from the other objects [35, 18, 47]. Conventional multi-class classifiers use data from at least two classes and the decision boundary is supported by the presence of data objects from each class. The data from the classes supposed to be more or less equally represented, otherwise the classifiers do not work that well. In contrast, the decision boundary from the OCC can be determined using only positive data, the one class. Defining the classification boundaries is more difficult for an OCC than for the conventional classifiers because the OCC has to accept as many objects as possible from the positive class while minimizing the chance of accepting outliers. A method to solve this problem is using a Support Vector Machine (SVM). An OCCSVM constructs a hyper-sphere around the positive class, that surrounds almost all positives points with a minimum radius. An outlier is classified as such, as the point falls outside the hyper-sphere [28]. The OCC model does need an estimation of the percentage of outliers within the data to set a threshold. This prior knowledge can be estimated. The complexity depends on the kernel function used. Hence, the OCC model included for further evaluation in this thesis.

4.1.4 Graph based models

Isolation Forest (IF): IFs consist of isolation trees. These trees are used to isolate instances by random partitioning. A tree is constructed by selecting a random feature and a random split value until either a height limit is reached, only one data point is left or all data have the same value. This partitioning is repeated recursively until all instances are isolated. Outliers are few and different making the outliers more susceptible to isolation than normal points, so the outliers can be isolated closer to the root, i.e. a short path length. Whereas normal points generally require more partitions to be isolated, i.e. a larger path length. This path length is used to determine the anomaly score. The IF algorithm has a linear time complexity with a low constant and low memory requirement. IF does not rely on prior knowledge and therefore included for further evaluation in this thesis [35, 33].

Hidden Markov models (HMM): HMM explores the relationship between consecutive segments of data [33, 12]. Within the logging data, this can mean the order of accessing the patient records. The likelihood of the next access can be calculated and then be classified. The complexity of the HMM is polynomial [55] and does not require labeled data. There is also no prior knowledge needed for the HMM. For those reasons, is the HMM included for evaluation in this thesis.

4.1.5 Neural net based models

Self-organizing maps (SOM): A SOM is an artificial neural network. The model clusters and projects a high dimensional input space into a two-dimensional output space. A SOM learns both the distribution and topology, so the topological properties of the data are preserved [52, 29]. Making it possible to visualize high dimensional data in a normal graph. A SOM can be used to determine new x en y coordinates in a two-dimensional space for each data-point. These coordinates can serve as input for other classification models or just for visualization of the dataset. The SOM is not a standalone classifier but can be used as input for other models. The SOM is included mostly for visualization and data prepossessing.

4.2 Package selection

We use packages to build the selected outlier detection models. We use packages because the focus of this paper lays in finding the best model for logging control, not to find any significant difference in different implementations.

In this section, we discuss the different packages that can be used to build a model in Python 3.7. The selection of packages is based on the availability and comprehensiveness of documentation, because we want to try to explain any significant differences in performance, if found. After the selection is made looking at the documentation, the performance of the package is compared. An overview of the packages and parameters each one needs is shown at the end of section 3.3 in Table 4.

4.2.1 LOF

There are multiple LOF packages available. Some stand-alone GitHub packages with little to no documentation and two well-documented and commonly used packages. We only test the well documented packages because there is not much known about the stand-alone GitHub packages such as the performance, i.e. one packages is only tested on whole numbers, not decimals [17].

The first package is a Scikit-learn package. Scikit-learn designs open-source, commercially usable machine learning models based on literature [39]. The package containing the implementation of the LOF model is called sklearn.neighbors.LocalOutlierFactor [50].

The second package is from PyOD. PyOD is a comprehensive and scalable Python toolkit for detecting outlying objects in multivariate data [61]. The PyOD package contains different outlier detection models. PyOD relies on Scikit-learn, so we expect not much of a difference between the two packages.

4.2.2 OCC

Three packages are found to build an OCC model in Python. The first package is from LibSVM (Library for Support Vector Machines). LibSVM is Integrated software for support vector classification [14]. The second package is the sklearn.svm.OneClassSVM from Scikit-learn [51]. Notable is that the implementation of the OCC model from the Scikit-learn packages is based on the LIBSVM. The

third package is PyOD. As for the LOF model, PyOD is based on the Scikit-learn package, so the foundation of all three packages are the same.

4.2.3 IF

Three promising packages are found to build an isolation forest. The first one is from Scikit-learn and is called sklearn.ensemble.IsolationForest [49]. The second package is from PyOD and the third package is from H2O.ai. H2O is an open-source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform that allows you to build machine learning models on big data and provides easy productionalization of those models in an enterprise environment [57]. H2O already compared their isolation forest algorithm to the Scikit-learn algorithm and the results are similar [2].

4.2.4 HMM

Two well-documented packages are found to build an HMM. The first package is from Scikit-learn [21]. The second package found is the pomegranate package [24]. The pomegranate package is an open-source machine learning package for probabilistic modeling in Python [42].

4.2.5 SOM

Seven SOM packages are already compared to each other based on the availability of documentation, simplicity of installation, and more [41]. Of the seven packages, four meet the criteria of useful documentation. All four are considered easy to install and are suitable for unsupervised clustering. The first package is a package proposed by the authors in the same paper [41]. The second package is the MiniSom package. MiniSom is a minimalistic Numpy based implementation of the SOM [44]. The third package is the PyMVPA. The PyMVPA is a specialized Python framework for machine learning-based data analysis proposed in another paper [20]. The last package is from NeuPY. NeuPy is a python library for prototyping and building neural networks. [46].

4.3 Parameter selection

In this section, the parameters for each model are explained. An overview of the packages and parameters can be found in Table 4. The parameters mentioned are the most important parameters of each model, the other parameters that are not mentioned deemed less important and are left on the default setting.

Table 4: Overview packages and parameters used. * means optional.

Model	**Packages**	**Parameters**
Local Outlier Factor	Scikit-learn PyOD	n_neighbors contamination*
One Class Classification	Scikit-learn LibSVM PyOD	kernel nu gamma
Isolation Forest	Scikit-learn PyOD H2O	contamination*
Hidden Markov Model	Scikit-learn Pomegranate	N/A
Self-Organizing Map	MiniSom NeuPy Susi PyMVPA	grid size learning rate epochs

4.3.1 LOF

The LOF model measures the local deviation of the density of a sample with respect to its neighbors. This locality is given by the k-nearest neighbors, their distance is used to estimate the local density. The most important parameter of the LOF model is the 'n_neighbors', the number of neighbors used to determine the density, i.e. the number of closest points to calculate the average distance. For most datasets, the number of neighbors should be between 10 and 20 [10]. We train the LOF model on a different number of neighbors to determine the influence of this parameter on the performance.

4.3.2 OCC

The most important parameter for the OCC model is the kernel. The kernel is a mathematical function, that takes the data as input and transforms the input, relative to that kernel function, into output. The different kernels functions are linear, polynomial, Radial Basis Function (RBF), and sigmoid. If the complexity of the function is known, a linear or polynomial function performs best, because this function is faster. Without prior knowledge, a RBF performs better compared to the linear and polynomial function [53]. A RBF is still able to classify input correctly even when the dimension space is not linearly separable by using a higher dimensional feature space. The sigmoid function is a logistic function, which is related to neural networks. The sigmoid function can behave like a RBF within a certain range of the sigmoid function's parameters and the performances of the sigmoid function is similar or worse than the RBF function [8, 31]. For this reason, a RBF function is used to train the model.

Another parameter is the kernel coefficient, gamma. Gamma is set on 1 divided by the number of features times the variance of the data point. The last important parameter is the 'nu'. 'nu' is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors [51]. 'nu' can be seen as a prediction of outliers within the data set.

4.3.3 IF

The 'n_estimators' parameter, the number of isolation trees in the isolation forest, is set to 100. The path lengths usually convert before a 100 [32]. The 'max_sample' parameter is the number of samples drawn from the data. This parameter is set to 'auto'. For datasets greater than 256 samples the 'max_sample' is set automatically on 256. A sub-sample size of 256 is proven to provide enough details to perform anomaly detection [32]. The contamination parameter is the proportion of outliers in the dataset. This is used to define a threshold on the anomaly score of the samples. The contamination parameter is set on 'auto', this way the threshold is determined automatically as stated in the paper [32].

4.3.4 HMM

The implementation of an HMM is different from the other models. An HMM requires linear data. The date is taken to order the different states. The states are the user and a patient file and the hidden states are appropriate or inappropriate access. After grouping the data, it results in 9810 states from which 9705 states are unique. There are too many unique states. To train the model correctly it is necessary that a state is visited multiple times from different states. With that many unique states, it is difficult to make predictions about the possibility of a future state. Even generalising the state, for example, taking the department or function instead of an user, results in too many unique states. The more the state is generalised the more difficult it is to find real outliers. For that reason, the HMM is excluded from the final set of learning models for doubting its feasibility. However, the HMM is still experimented with as an addition to the other models. By experimenting with different interpretations of the states and hidden states, such as taking the patient records as hidden states and calculating per state the probability a patient record is accessed.

4.3.5 SOM

The parameters changed from default are grid size, learning rate, and epochs. The grid size is the number of neurons for the two-dimensional space, each neuron corresponds with an x- and y-coordinate. The epoch parameter sets the number of passes, wherein each pass the weights are updated. The learning rate controls the changes made to the weights of the model. A small learning rate requires more epochs than a larger learning rate. While a too large learning rate causes the model to converge too fast to a sub-optimal solution.

The grid size is set to $5 * \sqrt{N}$, where N is the dataset size [45]. The learning rate is set to 0.01 and the number of epochs is 10.000. The combination of 10.000 epochs and a small learning rate of 0.01 is considered sufficient to converge to an optimal solution.

5 Control datasets

The disjunction between the labeled control datasets and the unlabeled hospital data becomes more evident during the next two section. This section focuses exclusively on the control datasets. Because the evaluation of the control datasets is different from the case study, we start with the evaluation plan. Then, we show the results of the two control datasets and we end with discussing the results.

5.1 Evaluation plan

The evaluation of the control datasets is different from the evaluation of the case study. The control datasets serve as an indication of how the models perform when trained on the unlabeled hospital data and as a benchmark set. A benchmark dataset can be used to compare the results with other papers. For the controlled dataset, the labels are known, we may use the performance measures that are also used to evaluate the supervised classification algorithms.

5.1.1 Performance measures

The calculation of all measures are based on the comparison between the labels and the prediction from the model. A confusion matrix can be made as an overview for this comparison, see Table 5. Outliers are classified as negative and inliers are classified as positive. The True Negative (TN) values from the cross-table are the real outliers that are predicted by the models as outliers. The True Positive (TP) values are the real inliers predicted as inliers. The False Negatives (FN) are the data points predicted by the model as outliers but are labeled as inliers. Whereas the False Positives (FP) are the data points predicted as inliers but are outliers.

Table 5: Confusion matrix: Outliers are classified as negative, Inliers are classified as positive. True Negative (TN), False Positive (FP), False Negative (FN) and True Positive (TP).

Labeled		Predicted	
	Outlier	TN	FP
	Inlier	FN	TP
		Outlier	Inlier

The accuracy is the average of the True Positives (TP) and the True Negative (TN), i.e. the average of all the right classified inliers (positives) and the right classified outliers (negatives).

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

For most models, the accuracy measure can be a decent representation of the performance of the model. However, because of the highly imbalanced ratio of inliers and outliers, classifying all data points as inlier will still result in an accuracy high up in the 90 percent. Therefore, we use measures such as the True Negative Rate (TNR), False Negative Rate (FNR) and, the Negative Predictive Value (NPV).

The TNR shows the capability to classify the right data point as outliers. The TNR is the true negative divided by the total data points classified as negative..

The TNR can be seen as the percentage of outliers found by the outlier detection model, for example, a TNR of 0.75 means that 75% of all outliers are classified as such.

$$TNR = \frac{TN}{TN + FP}$$

The FNR is the false negatives, so the inliers that are classified as outlier, divided by the total amount of inliers. FN / (FN + TP). The FNR can be seen as the percentage of inliers classified wrong. For example, a FNR of 0.10 means that ten percent of all inliers are incorrectly classified as outliers.

$$FNR = \frac{FN}{FN + TP}$$

The NPV is true positives divided by the number of data points classified as outliers: TP / (TN + FN). The NPV tells something about the percentage of points that are classified as an outlier are actually an outlier. For example, a score of 0.80 means that of all points classified as outliers, 80% is correctly classified as outliers.

$$NPV = \frac{TN}{TN + FN}$$

Previous measures all depend on a fixed threshold. Most outlier detection models score each point.The lower the score, the higher the chance this point being an outlier according to the model. The threshold determines the size of the top lowest-scoring point that should be classified as an outlier. For example, a threshold set on 1% on a dataset of 10.000 entries, will result in 100 outliers, the 100 lowest scoring points. Changing the threshold will change the outcome of a performance measures. It is possible to use this change in outcome per threshold to create a curve for a certain performance measure. This curve is made by gradually increasing the threshold, from zero to one or 0% to 100%, and calculate the outcome of this performance measure each time, resulting in a value of this performance measure per threshold. This eliminates the dependency of the manually set threshold.

When two performance measures are plotted against each other per threshold, a Receiver Operating Characteristics (ROC) curve is created. It is possible to create different ROC curves by plotting different performance measures against each other. Each ROC curve tells something different about the performance of a model. For example how much the model is capable of distinguishing between classes when plotting the TNR against the FNR. It is even possible to use the ROC curve to score and compare models by calculating the area under this curve, the Area Under the Curve (AUC) score. When plotting the TNR against the FNR, the AUC score represents the degree of separability, i.e. an AUC score of 0.75 means there is a 75 percent chance this model will classify a data point correctly. The ROC-AUC score is common approach to evaluate outlier detection models [11, 36, 15, 40, 18].

We mainly focus on two different ROC-AUC scores to evaluate the outlier detection models. The two ROC-AUC scores are derived from two graphs. The two graphs are the TNR plotted against the FNR, from now on referred to as TNR/FNR, and the NPV plotted against the TNR, referred to as NPV/TNR. The two graphs, TNR/FNR and NPV/TNR, can be explained with the help of Figure 4.

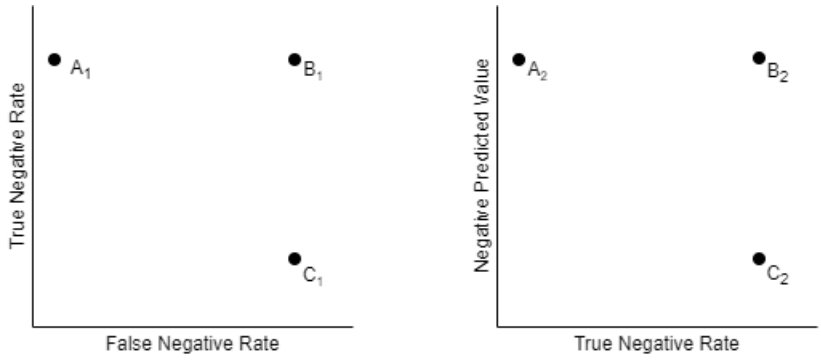

Figure 4: Example of the True Negative Rate plotted against the False Negative Rate (left) and Negative Predictive Value plotted against the True Negative Rate (right).

TNR/FNR curve represents the degree of separability between the two classes, the outliers, and the inliers. A high TNR and low FNR (point A_1 of Figure 4) means a model can detect most of the outliers, while keeping inliers, falsely classified as outlier, relative to the TP, low. Point A_1 results over time in point B_1. When in the beginning most outliers are found, the number of FNs increases automatically because when increasing the threshold can not result in finding more outliers because there are none. Point C_1 has a low TNR and high FNR meaning the model has not found many real outliers and the points that the model classified as outliers are mostly inliers. With unbalanced classes, as for outlier detection problems, are the FNs almost always relatively low compare to the TP, because the number of positive points (inliers) are 99 times greater than the negative points (outliers). A better representation of the performance is the NPV/TNR curve.

Because the NPV is calculated by dividing the TNs by the TNs and FNs, the NPV is not biased by the great number of positive points. Point B_2 of Figure 4 is harder to achieve than point A_1, or in a later stage point B_1. To get point B_2 the model needs to not only find almost all outliers but also keep the FNs to a bare minimum. A FN has more influence on the NPV than on the FNR, because the FNs are now relative to the smaller number of TNs instead of the greater number of TPs. Another example is point A_2, point A_2 can only happen in the beginning, at a small threshold. Only a few outliers are found, but almost all points classified as outlier are correct. Increasing the threshold from point A_2, can for example result in point B_2 when the FNs are kept minimal until all outliers are found or result in point C_2 if the number of FNs increases.

5.2 Results

An overview of the used packages and parameters can be found in Table 6. The motivation behind the chosen categorical value handling method, the selected packages, and parameter settings can be found in Appendix A.

The control datasets are downsized and normalized, as mentioned before. Because of this downsizing, the dataset varies a bit every time. To consider this variation we take the average of the TNR/FNV and NPV/TNR AUC score with a 95% Confidence Interval (CI) over ten runs. Next to this 95% CI we show the performance of each model after changing the ratio of outliers, the contamination. The contamination is set to 0.5%, 1%, 5%, 10%, 20%, 50%. Please note, a contamination of 50% means in this case, 50% of the total of normal instances, not 50% of the whole dataset. So, in relation to the whole dataset the contaminations are respectively 0.05%, 1%, 4.8%, 9%, 17% and 33%. As last a SOM is trained on the whole dataset. The LOF, OCC, and LOF models are trained both on the output of the SOM and the regular dataset, where the SOM is also trained on, to show the difference in performance.

Table 6: Overview selected packages and parameters. Local Outlier Factor (LOF), One Class Classification (OCC), and Isolation Forest (IF).

Model	**Package**	**Parameter**	**Parameter value**
LOF	Scikit-Learn	kNN	between 7.5% and 15% of the dataset
IF	Scikit-Learn	N/A	
		kernel	RBF
OCC	LibSVM	nu	0.01
		gamma	auto
SOM	Susi	grid size	39
		epochs	10.000

5.2.1 KDDcup99

LOF: The optimal kNN parameter setting for the LOF models is 7.5% of the KDDcup99 dataset, see Appendix subsubsection A.3.1. 7.5% of the whole KDDcup99, after downsizing to fit the right inlier/outlier ratio, is 75.000 neighbours. However, keeping track of the distance of 75.000 for almost a million records is computationally expensive, too expensive for the used hardware. Therefore, we run a subset of 10.000 points instead of the original 1.000.000 to show the potential of the LOF model. Instead of running the model ten times, as for the other models, we run the LOF model fifty times on different subset of 10.000 rows with a contamination of one percent. We run the LOF fifty times instead of ten because the subset is only one percent of the original dataset. The smaller a subset the higher the chance a subset does not accurately represent the whole set. When a subset does not represent the whole set, the performance of the used model can be different. The 95% CI already takes these differences into account, but needs more runs to minimize the range of the 95% CI. We can do fifty runs, instead of ten runs as for the OCC and IF model, because taking a subset of only 10.000 points decreased the run time

compared to the OCC and IF model.

The average TNR/FNR AUC score over fifty runs is 0.927 (95% CI 0.925 to 0.929), an example of a plot can be seen in Figure 26 of Appendix subsection B.1.

Next, the average NPV/TNR AUC score is 0.145 (95% CI 0.144 to 0.146). The NPV plotted against the TNR and the NPV and TNR separate per threshold can be found respectively in Figure 27 and Figure 5 of Appendix subsection B.1.

Looking at the separate TNR and NPV (Figure 5 we see five different stages. The first stage of around 80 data points, 1% of 10.000, are almost all FNs, where the most TNs are expected. The second stage up until around 200 points are all TNs, illustrated by the steep increase of the TNR line. As a result, the NPV increases too. The third stage is characterised by a flat TNR line. This flat line means almost no TNs are found, consequently the NPV line drops. The next little increase, stage four, means an increase in TNs, until stage five where around 90% of the outliers is classified as such.

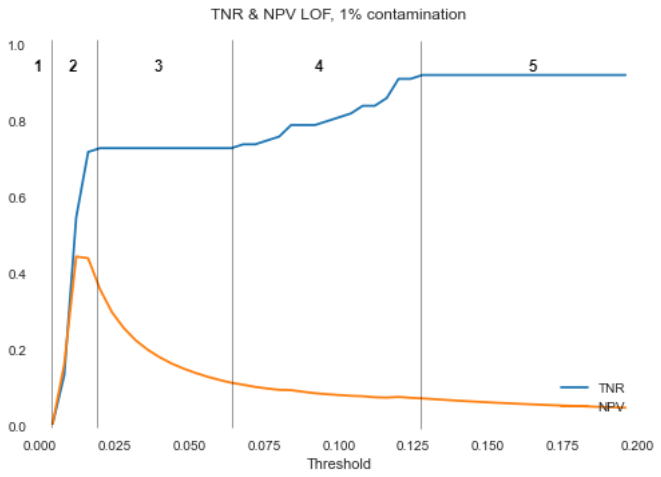

Figure 5: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value Local Outlier Factor (LOF) model divided into different stages, Dataset used: KDDcup99

Next, the LOF model is trained on a contamination of 0.5%, 1% , 5% 10%, 20%, and 50%. Because the number of points increases, the kNN parameter is kept on 7.5% of the number of points per subset. The results can be found in Figure 6.

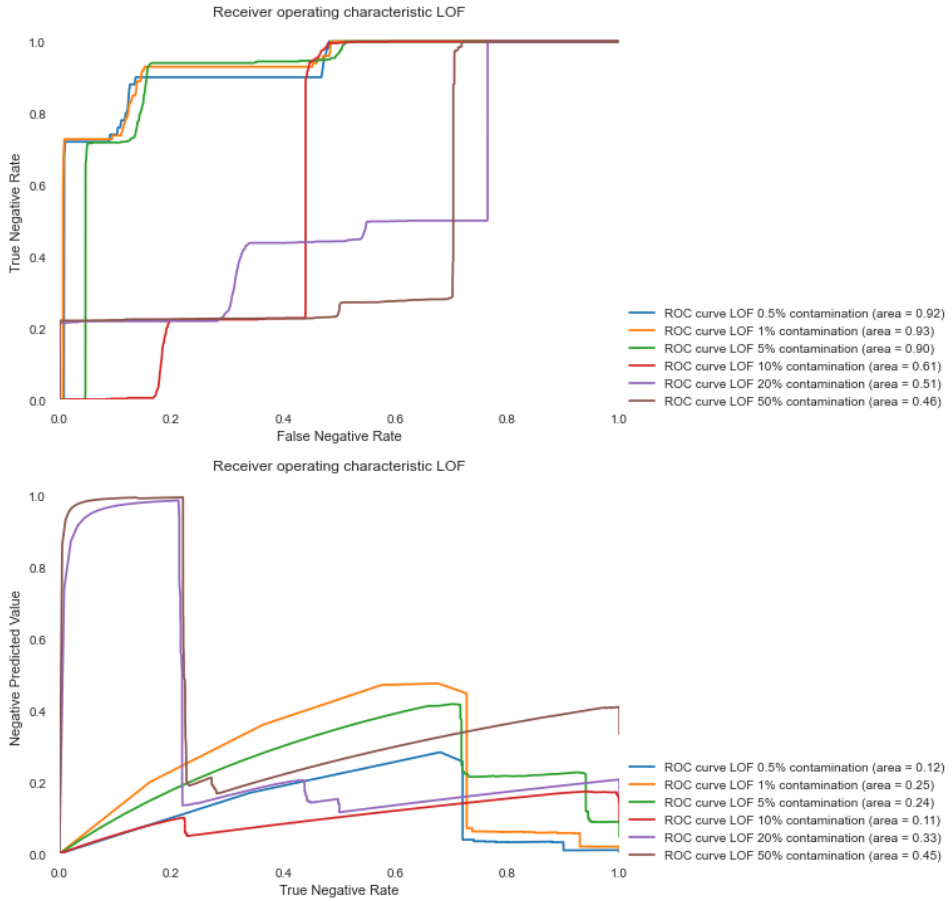

Figure 6: Local Outlier Factor (LOF) model trained on different contaminations with Area Under the Curve score. Dataset used: KDDcup99.

Figure 6 shows an overall decline of the TNR/FNR AUC score, when the contamination increases, while the NPV/TNR AUC score increases, except for a contamination of 0.5% and ten percent.

OCC: The OCC model is trained ten times on the KDDcup99 with a contamination of one percent. The average TNR/FNR AUC score is 0.991 (95% CI 0.991 to 0.991). An example of a TNR/FNR plot can be found in Figure 31 of Appendix subsection C.1.

Then, the average NPV/TNR AUC score is calculated: 0.49 (95% CI 0.489 to 0.491). As for the TNR/FNR curve, the NPV/TNR curve all look the same, an example can be seen in Figure 32 of Appendix subsection C.1. Besides the NPV/TNR curve the NPV and TNR plotted separately in one graph for different thresholds too, see Figure 33 of Appendix subsection C.1.

Last, the performance of the OCC model trained on contaminations is shown in Figure 7.

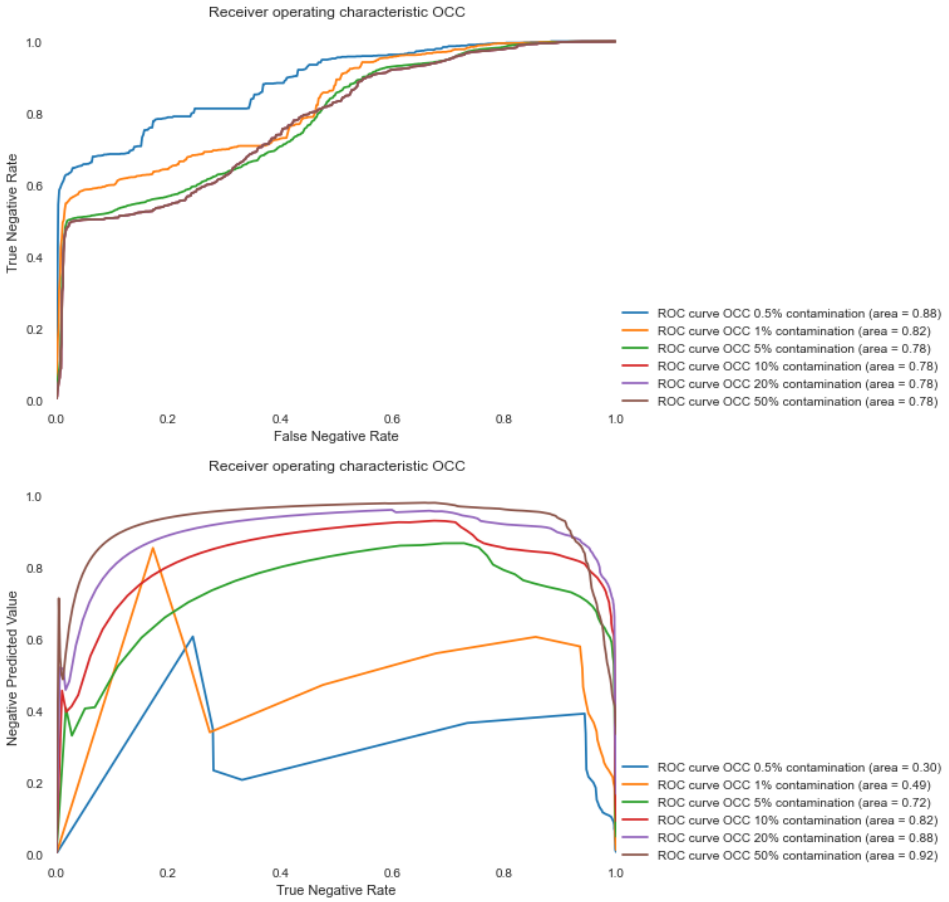

Figure 7: One Class Classification (OCC) model trained on different contaminations with Area Under the Curve score. Dataset used: KDDcup99.

Figure 7 shows at first a decline of the TNR/FNR AUC score, with the increase of the contamination, but then the TNR/FNR AUC score stabilizes at 0.78. On the contrary, the NPV/TNR score keeps improving.

IF: First is the IF model trained ten times on the KDDcup99 with a contamination of one percent. The average TNR/FNR AUC score over these ten runs is 0.990 (95% CI 0.986 to 0.994), a single plot can be seen in Figure 37 of Appendix subsection D.1.

Secondly, the NPV/TNR AUC score is calculated, the average score is 0.497 (95% CI 0.395 to 0.599). Looking at the 95% CI we see a greater range of the AUC scores than for the OCC model. The range of the NPV/TNR AUC score of the IF model is a little over 0.3 while the range of the same score for the OCC model is only 0.003. This greater difference can be seen in the TNR/NPV plots. Figure 8 shows two extremes.

Both plots start the same, an increase with a peek at a TNR of 0.2. Then both the plots decrease a little, meaning for a period there a more FNs than TNs. After this decrease, the NPV of the better scoring plot stays stable for the remaining

thresholds, while the NPV of the other plot plummets down in a straight line, at this stage there only FNs added when increasing the threshold. Looking at the more average scoring plots, instead of these two extremes, we see that the score depends on how much the model drops after the first peak.

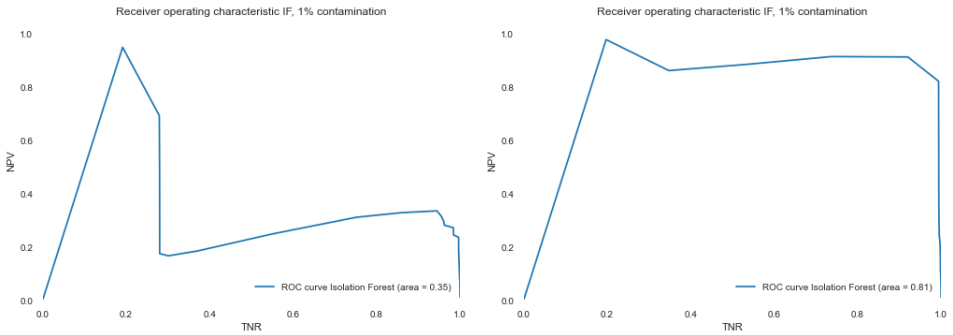

Figure 8: Negative Predictive Value/True Negative Rate curve Isolation Forest (IF) model. Dataset used: KDDcup99.

Instead of plotting the NPV and TNR against each other, we also have plotted the NPV and TNR separated in one graph. The NPV and TNR scores per threshold value from the best scoring run can be found in Figure 38 of Appendix subsection D.1.

Last, the performance of the IF model trained on different contaminations can be seen in Figure 9.

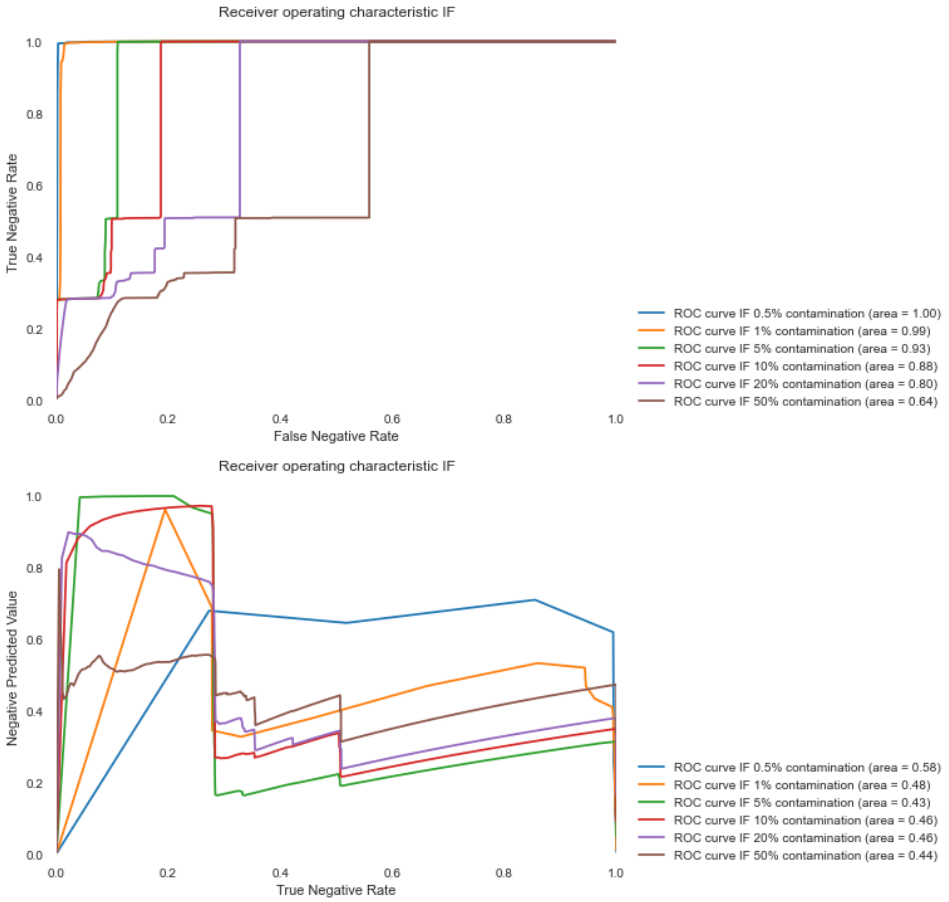

Figure 9: Isolation Forest (IF) model trained on different contaminations with Area Under the Curve score. Dataset used: KDDcup99.

Figure 9 shows a decrease of the TNR/FNR AUC score and a slight decrease of the NPV/TNR score with an increase of the contamination, except for a contamination of five percent.

SOM: A SOM with a grid size of 39 is trained with 10.000 epochs on the KDDcup99 dataset with a contamination of one percent. Another SOM, with the same parameters, is for the LOF model from a subset of 10.000 rows from the KDDcup99. The SOM maps the 41 features into a two-dimensional space. Because the data is mapped on a two-dimensional space we are able to visualise the data.

Figure 10: Visualisation of output data Self Organising Map in a hexbin plot

Figure 10 shows the output of the SOM. A couple of clusters are visible by its dark(er) green colour, the darker the colour the more points present within that hexbin. In theory, those clusters represent the normal instances, and points that lays outside those clusters should be the outliers. Each data point from the KDDcup99 is mapped in a 39 by 39 space, so each data point is assigned a coordinate within that space. These new coordinates serve as input for the outlier detection models.

The new coordinates for each data point are now used to train the three outlier detection models. The OCC and IF models are trained ten times and the LOF model fifty times. The results can be found in Table 7. The corresponding figures can be seen as Figure 42, Figure 43 and Figure 44 of Appendix subsection E.1. Table 7 shows that the performance of all three models is worse when trained on the SOM.

Table 7: Comparison performance outlier detection models trained on Self Organizing Map (SOM) and without SOM. Area Under the Curve score True Negative Rate plotted against False Negative Rate (TNR/FNR), Area Under the Curve score Negative Predictive Value plotted against True Negative Rate (NPV/TNR), Local Outlier Factor (LOF), One Class Classification (OCC), and Isolation Forest (IF). Dataset used: KDDcup99.

Model		**TNR/FNR**	**NPV/TNR**
LOF	on SOM	0.898 (95% CI 0.895 to 0.901)	0.091 (95% CI 0.089 to 0.093)
	without SOM	0.927 (95% CI 0.925 to 0.929)	0.145 (95% CI 0.144 to 0.146)
OCC	on SOM	0.452 (95% CI 0.452 to 0.452)	0.012 (95% CI 0.011 to 0.013)
	without SOM	0.988 (95% CI 0.988 to 0.988)	0.368 (95% CI 0.367 to 0.369)
IF	on SOM	0.919 (95% CI 0.911 to 0.927)	0.126 (95% CI 0.070 to 0.182)
	without SOM	0.990 (95% CI 0.986 to 0.994)	0.497 (95% CI 0.395 to 0.599)

Table 7 shows an overall decline in performance for all three models. This decline is especially visible in the NPV/TNR AUC score of the OCC and IF model.

5.2.2 ISCXIDS2012

LOF: As for the KDDcup99, the optimal kNN parameter setting of 15% of the total dataset (see Appendix subsubsection A.3.2 is for the entire ISCXIDS2012

dataset computational too expensive to run. Therefore, we take a ten percent subset of 25.000 rows and run the LOF with the kNN set on 3750. Instead of the normal ten runs with a contamination of 1%, we run this smaller subset fifty times, because of the variety between the different subsets.

The average TNR/FNR AUC score over fifty is 0.987 (95% CI 0.986 to 0.988). The average NPV/TNR AUC score is 0.846 (95% CI 0.842 to 0.85). An example of the TNR/FNR plot, the NPV/TNR plot, and the NPV and TNR plotted separate can be found respectively in Figure 28 , Figure 29 , and Figure 30 of Appendix subsection B.2.

Next, the LOF model is trained on subsets with different contaminations, because the size of each subset differs, the kNN parameter changes accordingly to keep the parameter on 15% of the total dataset. The amount of inliers stays the same over the subsets and is set on 12.500 because the used hardware can not handle a ten percent subset with a contamination of 50%. The results can be seen in Figure 11.

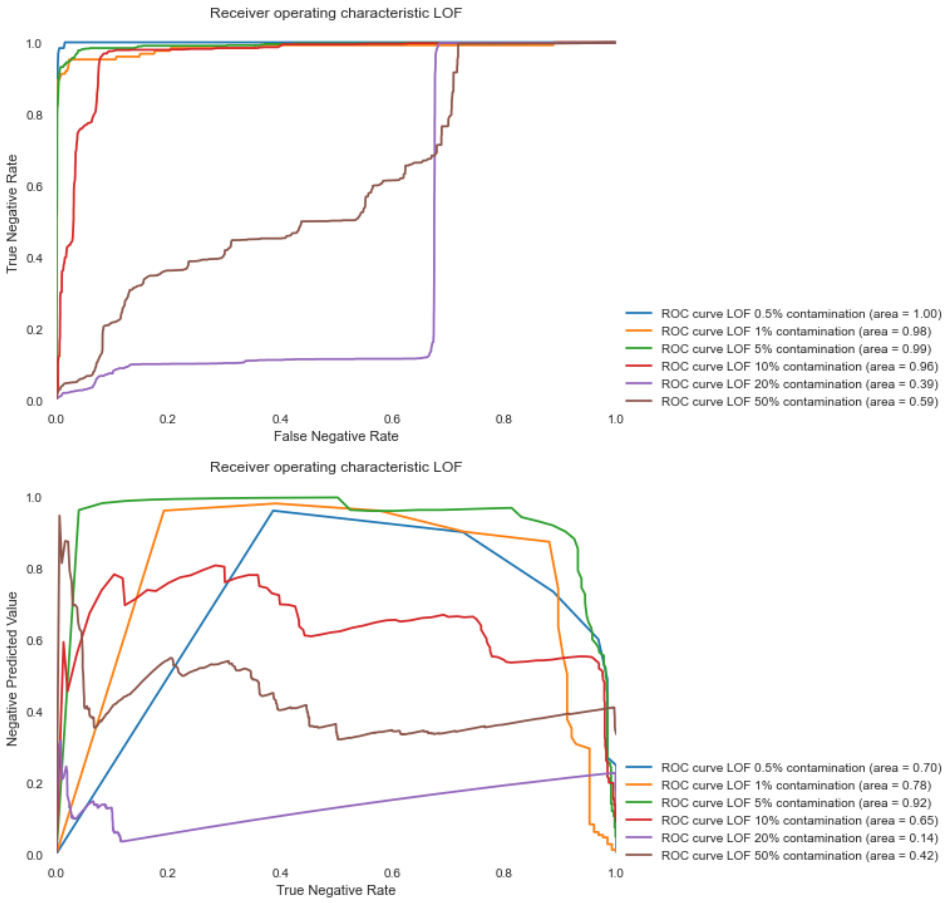

Figure 11: Local Outlier Factor (LOF) model trained on different contaminations with Area Under the Curve score. Dataset used: ISCXIDS2012.

Figure 11 shows an overall decrease in performance looking at both the TNR/FNR and NPV/TNR scores when increasing the contamination. There are three exceptions. The first exception is the contamination of 0.5%, the NPV/TNR AUC score is lower than the NPV/TNR AUC score of a contamination of one percent, where a higher score is expected. The contamination of five percent is the second exception, both the TNR/FNR and NPV/TNR scores are greater than the contamination of one percent. The last exception can be or the contamination of twenty percent or the contamination of fifty percent. Because, or the contamination of twenty percent is extremely low or the contamination of fifty percent is too high.

OCC: The OCC model is run ten times on the ISCXIDS2012 set with a contamination of 1%. The average TNR/FNR AUC score is 0.825 (95% CI 0.823 to 0.827) and the average NPV/TNR score is 0.206 (95% CI 0.203 to 0.209). An example of the TNR/FNR plot, the NPV/TNR plot, and the NPV and TNR plotted separate can be found respectively in Figure 34 , Figure 35 , and Figure 36 of Appendix subsection C.2. Then, the OCC model is trained on subsets with various contaminations, the results can be found in Figure 12 .

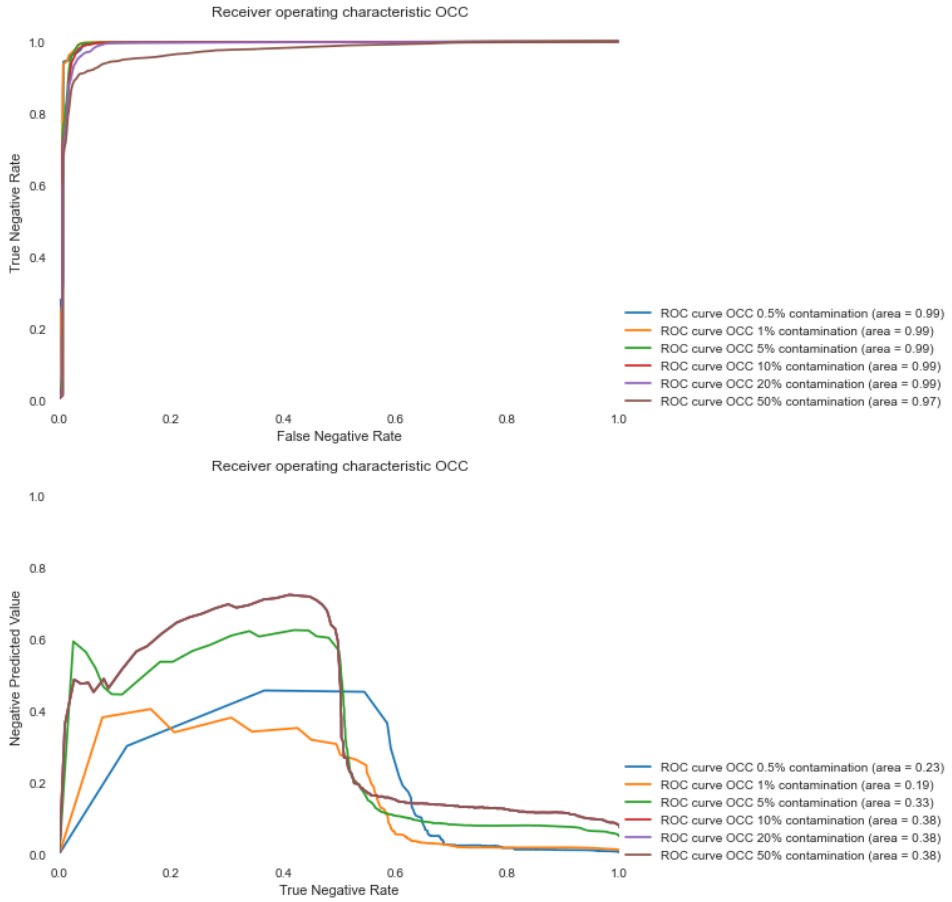

Figure 12: One Class Classification (OCC) model trained on different contaminations with Area Under the Curve score. Dataset used: ISCXIDS2012.

Figure 12 shows a stable TNR/FNR score with different contaminations, except for the contamination of fifty percent. The NPV/TNR AUC score increases up to a contamination of ten percent, then the score stabilises, except for a contamination of one percent.

IF: First, the IF is trained ten times on the ISCXIDS2012 dataset with a contamination of 1%. The average TNR/FNR AUC score over ten runs is 0.979 (95% CI 0.976 to 0.982). The average NPV/TNR score over ten runs is 0.280 (95% CI 0.231 to 0.329). An example of the TNR/FNR plot, the NPV/TNR plot, and the NPV and TNR plotted separate can be found respectively in Figure 39, Figure 40, and Figure 41 of Appendix subsection D.2. Secondly, the IF is trained on multiple subsets of the ISCXIDS2012 dataset with different contaminations while keeping the random state the same. The TNR/FNR and NPV/TNR plots can be seen in Figure 13.

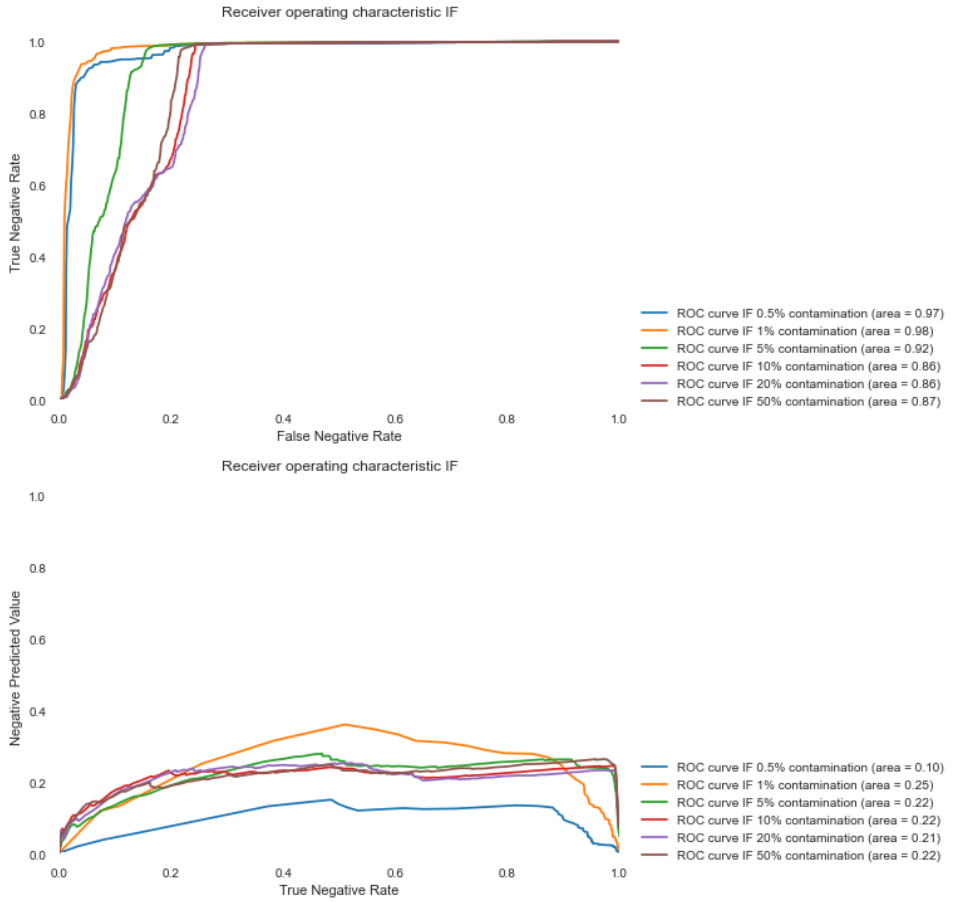

Figure 13: Isolation Forest (IF) model trained on different contaminations with Area Under the Curve score. Dataset used: ISCXIDS2012.

Except for a contamination of one and fifty percent, Figure 13 shows a decrease in both the TNR/FNR and NPV/TNR scores.

SOM: A SOM is trained with 10.000 epochs on the ISCXIDS2012 dataset with a contamination of 1%. The SOM has a grid size of 39. A second SOM is trained on a 12.500 record subset of the ISCXIDS2012 for the LOF model to run on. A visualisation of the high dimensional ISCXIDS2012 on a two-dimensional grid can be seen in Figure 14 .

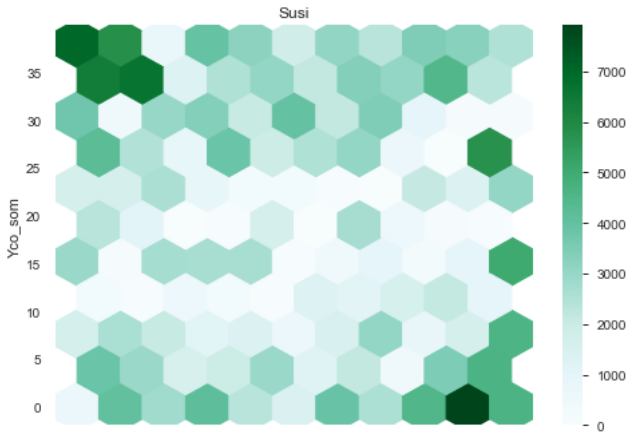

Figure 14: Visualisation of output data SOM trained on the ISCSIDS2012 in a hexbin plot.

The new coordinates for each data point are now used to train the three outlier detection models. The OCC and IF models are trained ten times and the LOF model fifty times. The results can be found in Table 8. The corresponding figures can be found as Figure 45, Figure 46 and Figure 47 of Appendix subsection E.2. Table 8 shows that the performance of all three models is worse when trained on the SOM.

Table 8: Comparison performance outlier detection models trained on Self Organizing Map (SOM) and without SOM. Area Under the Curve score True Negative Rate plotted against False Negative Rate (TNR/FNR), Area Under the Curve score Negative Predictive Value plotted against True Negative Rate (NPV/TNR), Local Outlier Factor (LOF), One Class Classification (OCC), and Isolation Forest (IF). Dataset used: ISCXIDS2012.

Model		**TNR/FNR**	**NPV/TNR**
LOF	on SOM	0.921 (95% CI 0.919 to 0.923)	0.064 (95% CI 0.056 to 0.072)
	without SOM	0.987 (95% CI 0.986 to 0.988)	0.846 (95% CI 0.842 to 0.850)
OCC	on SOM	0.433 (95% CI 0.432 to 0.434)	0.042 (95% CI 0.040 to 0.044)
	without SOM	0.814 (95% CI 0.811 to 0.817)	0.144 (95% CI 0.138 to 0.150)
IF	on SOM	0.854 (95% CI 0.821 to 0.887)	0.038 (95% CI 0.032 to 0.044)
	without SOM	0.979 (95% CI 0.976 to 0.982)	0.280 (95% CI 0.231 to 0.329)

Table 8 shows also an overall decline in performance for the LOF and IF models. However, the TNR/FNR AUC score of the OCC model increases, while the NPV/TNR AUC score decreases. Another difference with the KDDcup99 is the greater difference in score between the NPV/TNR AUC score of the LOF trained on the SOM and without the SOM.

5.3 Discussion

During the discussion of the control datasets, we answer the first sub-question: *How accurate are machine learning models in detecting anomalies in terms of the trade-off between being as precise as possible and detecting all anomalies.* But first, we summarize and discuss the results. We compare the results from the outlier detection models to each other, among control datasets, and to literature. Then, we answer the sub-question, state the limitations, and state our recommendations for future research. We end by discussing the suitability of the used model for the case study.

5.3.1 Control datasets

Because the control datasets are labeled, we can use performance measures used for supervised learning. Those performance measures give a detailed representation of the performance of each model. First, a summary of the results can be found in Table 9. Then we discuss the results obtained from the TNR/FNR plots. Next, we discuss the results from the NPV/TNR plots and the effect different contaminations have on the performance of each model. We end this section with the performance of the SOM.

Table 9: Overview performance outlier detection models trained on KDDcup99 and ISCXIDS2012 with a contamination of one percent. (+) is positive effect on performance with an increase of the contamination and (-) is negative effect on performance with an increase of the contamination. Area Under the Curve score True Negative Rate plotted against False Negative Rate (TNR/FNR), Area Under the Curve score Negative Predictive Value plotted against True Negative Rate (NPV/TNR), Local Outlier Factor (LOF), One Class Classification (OCC), and Isolation Forest (IF).

Model	**TNR/FNR**	**NPV/TNR**	**Contamination**
LOF KDDcup99	0.927 (95% CI 0.925 to 0.929)	0.145 (95% CI 0.144 to 0.146)	+
OCC KDDcup99	0.991 (95% CI 0.991 to 0.991)	0.490 (95% CI 0.489 to 0.491)	+
IF KDDcup99	0.990 (95% CI 0.986 to 0.994)	0.497 (95% CI 0.395 to 0.599)	-
LOF ISCXIDS2012	0.987 (95% CI 0.986 to 0.988)	0.846 (95% CI 0.842 to 0.850)	-
OCC ISCXIDS2012	0.825 (95% CI 0.823 to 0.827)	0.206 (95% CI 0.203 to 0.209)	+
IF ISCXIDS2012	0.979 (95% CI 0.976 to 0.982)	0.280 (95% CI 0.231 to 0.329)	-

TNR/FNR: The TNR/FNR ROC AUC score represents the degree of separability. The higher the AUC, the better the model is in distinguishing between classes, or in this case between normal instances and outliers. For example an AUC score of 0.60 means there is 60% chance the model separates the outliers from the normal instances. An AUC score of 1.0 implies a model is perfectly able to distinguish the outliers from normal instances. Where a score of 0.50 means a model has no

discrimination capacity. All scores less than 0.50 implies random guessing a point would score better.

The TNR/FNR AUC score will normally be higher for outlier detection models using imbalanced datasets. The higher AUC score is caused by the FNR. The FNR is calculated by dividing the FNs by all normal instances. The normal instances makeup 99% of the dataset, compared to the 1% outlier class. So even if there are relatively many FNs compared to TNs, the FNs compared to the TPs are almost negligible. For example, take a dataset of 100.000 points with 1000 outliers. By calculating the TNR/FNR AUC score, the threshold moves evenly from 0% to 100%. When the threshold is at for example at 3%, the 3000 lowest scoring points are now classified as an outlier. Those 3000 points consist of, for example, 900 TN and 2100 FN. 2100 FN is a lot compared to the TNs, but when calculating the FNR the 2100 FNs are divided by 99.000, resulting in a low FNR (0.02). The TNR is at this point 0.90, so the point in the graph lays somewhere high against the Y-axis. Connecting all the dots close to the Y-axis result in an extensive area under the line, because this line is following the Y-axis almost all the way up.

The TNR/FNR AUC scores from the LOF, IF and OCC models are high (Table 9) due to the reason stated above. Where 95% CI over ten runs is stable for the OCC and LOF model, the average AUC score of the IF model can differ with 0.004. This difference can be explained through the way each IF is constructed. Because of the random selection of features and the split values, each forest is slightly different each run, so the performance changes too.

NPV/TNR: Unlike the TNR/FNR, the NPV/TNR ROC curve is not biased by the imbalanced classes. The NPV is the percentage of negative points (outliers) classified correctly, where the TNR is the percentage of outliers found. The NPV/TNR is a negative version of the better-known precision/recall curve. An outlier is classified as a negative, therefore we use the NPV/TNR instead of the precision/recall curve.

Normally we expect the NPV/TNR curve to start in the left upper corner and as the threshold is raised usual the FNs increase too, so the line drops slowly to the right corner. When creating the NPV/TNR plot (and the other plots too), the points are ordered from the lowest score to the highest-scoring points. The lower the score, the more likely this point is an outlier. Therefore, the NPV should be the highest in the beginning (following the dotted line) because the model scores those points as 'highest possibility to be an outlier'.

If the origin is point (0,0), an immediate increase is expected to the left upper corner, if started in point (0,1) a slight decrease is expected. We see this increase happen for example with the LOF trained on the ISCXIDS2012 (Figure 15 right).

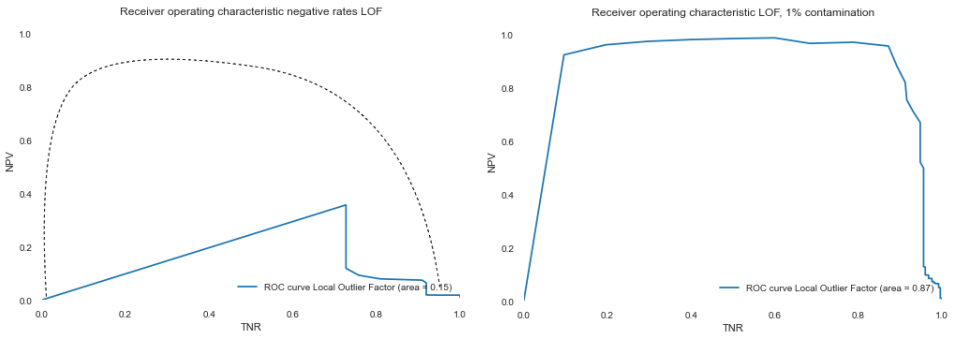

Figure 15: Negative Predictive Value/True Negative Rate (NPV/TNR) curve Local Outlier Factor (LOF) model and expected NPV/TNR line (dotted). Dataset used: KDDcup99 (left) and ISCXIDS2012 (right)

However, the immediate increase to the left upper corner can not be seen for the LOF model trained on the KDDcup99 (Figure 15 left). As mention, wherein the beginning lots of TNs are expected, the LOF model trained on the KDDcup99 classified the first one percent of the dataset wrong.

Where the NPV/TNR AUC of the LOF model improved when trained on the ISCXIDS2012 compared to the KDDcup99, the NPV/TNR score of the IF and OCC models worsened, see 10.

Table 10: Overview difference in Area Under the Curve score for the Negative Predictive Value plotted against the True Negative Rate when applied on the KDDcup99 compared to when applied on the ISCXIDS2012 based on the range of the 95% Confidence Interval. Local Outlier Factor (LOF), One Class Classification (OCC), and Isolation Forest (IF).

Model	**Difference**
LOF	+ (0.696 - 0.706)
OCC	- (0.280 - 0.288)
IF	- (0.066 - 0.368)

The decrease in performance might be explained by the more realistic ISCXIDS2012 compared to the outdated KDDcup99. The ISCXIDS2012 is based on a more realistic network with more realistic attacks [48]. It can be harder to detect or separate outliers from inliers because they are now more similar to normal instances. On the contrary, the LOF model performs better on the ISCXIDS2012 dataset.

The difference performance, for the LOF model, might be explained through the curse of dimensionality. Especially, (local) density-based algorithms, such as the kNN method or more advanced method LOF, are affected by this phenomenon [62]. When increasing the dimensionality of the average length between points, the measure that is used to decide if a point is considered an inlier or outlier, all tend to converge to the same value, also known as the central limit theorem. This loss of numerical contrast makes it difficult for the LOF model to classify outliers correctly, hence the difference performance [62]. Where the KDDcup99 contains 41 features, the ISCXIDS2012 only contains 19 features.

A method to overcome the curse of dimensionality is by feature selection. Not

only can feature selection improve the performance of the LOF model trained on the KDDcup99, but feature selection also reduces over-fitting and run time too. There are different ways to select features such as selecting the features that minimizing the error rate or selecting features with the highest mutual information. But, labeled data is needed to calculate the error rate, mutual information, or other measures. So we can not use well-founded feature selection methods. Therefore, we select the features from the hospital data that we, in consultation with experts from the hospital, think are the best features to detect outliers with.

Variety of contaminations: We see a decrease at the TNR/FNR AUC score of the LOF model trained on the KDDcup99, while the NPV/TNR AUC score is increasing, see Figure 6. The decrease is caused by the calculation of the FNR. The FNR is disproportionately high because of the high number of positive points, the inliers, compared to the negative points, the outliers. Increasing the number of outliers, straighten this imbalance a bit, making the FNR score lower. A lower FNR results in a lower AUC, but that does not imply that the performance is worse too. The same effect can be seen in less severely with the OCC model. This effect is less visible because the performance improves so much, that this improvement out weights this effect, see Figure 7.

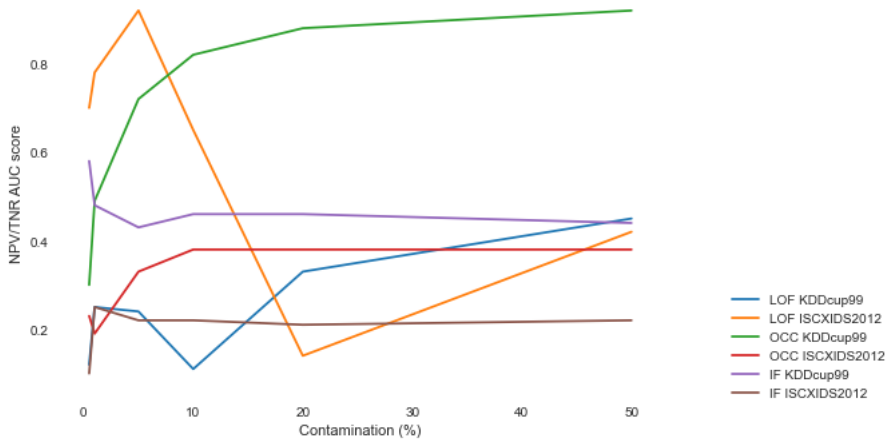

Figure 16: Area Under the Curve (AUC) score for the Negative Predictive Value plotted against the True Negative Rate (NPV/TNR) for various contamination. Local Outlier Factor (LOF), One Class Classification (OCC), and Isolation Forest (IF)

Increasing the contamination has a slightly negative effect on the NPV/TNR AUC score of the IF, while the NPV/TNR of the OCC increases, see Figure 16. This difference may be caused by the way each model classifies outliers, while the IF focuses on the negatives, actively searching for negative points, the OCC focuses on the positive points. Increasing the number of negative points increases the chance of negative points being mapped close together. Two negative points laying close together makes it more difficult for the IF to isolate the points, thus a negative influence on the performance.

The effect different contaminations have on the outlier detection models is lower

for the IF than for the OCC model (Figure 16, the difference in performance between the lowest and highest contamination is respectively only 0.02 compared to 0.62. So without knowing the exact contamination the IF is the safer choice. If the contamination is known and is relatively high, for example more than 5%, the OCC model is the better option, but if the contamination is less the IF model will presumably perform better.

We see deviations in the performance while increasing the contamination. For example, the LOF model trained on the KDDcup99 (blue line of Figure 16. While overall the blue line increases, there is a decline at a contamination of ten percent. This deviation may be explained by the variations between subsets. While all the inliers remain the same, the outliers differ over subsets because of downsizing. So, it can happen that a random sample does not completely represent the entire group of outliers, this sample can be in favor of the performance or the opposite. So not only the change in contamination has an effect on the performance but the different random subsets too.

This phenomenon is more visible for the LOF model, where not only the outliers are sampled but the inliers too. Moreover, the entire subset is around five percent of the ISCXIDS2012 and the smaller the subset the higher the chance the subsets vary. But in the big picture, the performance of the LOF model decreases while increasing the contamination. In contrary to the KDDcup99. We are not sure if this decrease, instead of increase, is caused mainly by the increase of outliers, by the more realistic data, by the difference between subsets, or the fact that the performance of the LOF model with a contamination of one percent is significantly higher on the ISCXIDS2012 than on the KDDcup99. Another explanation can with higher contaminations the chance increases that (more) outliers are used too, to calculate the density. Where normally outliers are not supposed to be in dense areas, thus large distances to other points. But with more outliers, those new outliers may lay close to other outliers, so the distance to other points decreases. However, increasing the contamination too much violate the assumptions about outlier that outliers are sparse and different.

SOM: Looking at the performance, all models score worse when trained on the SOM, see Table 7 and Table 8. Especially the OCC model, with a TNR/FNR AUC score less than 0.5 for both datasets, the OCC its ability to detect outliers better than chance.

The difference in performance can be explained by the dimensionality reduction. Ideally, the outliers are different enough to be mapped to other points on the grid than the normal instances, but when the characteristics of outliers are too close to the inliers, it may even happen that the outlier is mapped too close, or even to the same coordinates as an inlier. When both, the outlier and inliers, are represented by the same coordinate, it impossible for the detection models to differentiate the outlier from the inliers.

5.3.2 Literature

To set these three models in perspective, two other papers tested outlier detection models in approximately the same setting, namely the KDDcup99 dataset with a contamination of one percent. Portnoy et al. were able to "detect a large number

of intrusions while keeping the false positive rate reasonably low" [40]. Their best scoring dataset has a TNR of 0.557 and a FNR of 0.099. This paper classifies an outlier as positive, so the TNR is called the detection rate and the FNR is called the false positive rate. It is difficult to compare the results because Portnoy et al. have not included any AUC scores or used other performance measures. But, based on our graphs the OCC and IF model performs better and the LOF model slightly less.

Eskin et al. present three algorithms for outlier detection [18]. These algorithms consist of a cluster-based algorithm, a kNN based algorithm, and an OCC based algorithm. As for the previously mentioned paper, this paper has not used any other performance measure other than the TNR and FNR. However, a TNR/FNR ROC plot is included, without the AUC score. To compare the score we have to estimate the AUC score. Based on previously seen ROC curves, we estimate that the cluster-based and OCC algorithm has an AUC score of around 0.95, as the kNN algorithm has an AUC score of around 0.90. The OCC and IF in this thesis are performing better, but to make a better comparison, more performance measures are preferred.

A third paper tested also an Outlier detection method on the KDDcup99 [6]. However, Bivens et al. separate the attacks so each subset contains a single attack on the simulated network because Bivens et al. had trouble getting the neural net to detect all types of attacks simultaneously. Bivens et al. used a SOM in combination with a clustering method. The difficulty Bivens et al. had with the SOM can be seen in our results too, where the outlier detection models perform better on their own than when trained on a SOM.

5.3.3 Limitations and recommendations

A limitation of this research is the hardware used for the control datasets. We are not able to run the LOF on the entire data set, making it harder to compare the LOF to the other models, because the LOF uses subsets and the other two not.

In future research, we can investigate more the effect different contaminations have on the SOM. Because of the limited time-span, we only trained the SOM on a contamination of one percent. We do not know how the SOM performs with higher or lower contamination.

5.3.4 Suitability models case study

In this section, we discuss what we expect of the outliers detection models when trained on the EHR logging data based on the results from the control datasets.

At this point, we exclude the SOM and LOF model. The SOM model does not contribute to a better performance in any way and therefore not used in the case study. While the LOF model shows promising results on the ISCXIDS2012 dataset, the LOF model is computational too heavy and the range of the optimal parameter setting is too small. The optimal setting of the kNN parameter for the control datasets is around 7.5% of the entire set for one dataset and around 15% for the other. One month of hospital logging data consists of almost 9.000.000 rows. Despite the better hardware used at the hospital, we do not expect that hardware can handle such computations. Even if the hardware could manage the optimal parameter setting, the range is too small. For example deviating five percent from

the optimal setting results in a NPV/TNR AUC score, for the KDDcup99, of almost zero. Ideally, this range would be larger, so without knowing the precise optimal setting, estimating this setting based on the control datasets still results in a decent score. However, the optimal setting even differs between the two control datasets. Using the optimal setting from one control for the other control dataset results in a score worse than chance.

From the two remaining models, the OCC and IF model, we expect the IF to perform better than the OCC. The IF is more stable over different contaminations, while the performance decreases with higher contaminations, the performance increases with less. We expect the hospital dataset rather contain less than one percent outliers than more than one percent outliers. On the contrary, the performance of the OCC model is more susceptible to contamination changes and the OCC performs better on higher contaminations.

5.3.5 Sub question 1

The first sup-question, *How accurate are machine learning models in detecting anomalies in terms of the trade-off between being as precise as possible and detecting all anomalies* addresses two parts 'being precise' and 'detect all anomalies'. The 'precise' part can be seen as the NPV and the 'detect all anomalies' can be represented by the TNR. The answer of the first sup-question can be seen for the OCC model in Figure 33 and Figure 36 (of Appendix subsection C.1 and Appendix subsection C.2) and for the IF model in Figure 38 and Figure 41 (of Appendix subsection D.1 and Appendix subsection D.2). Two examples of those figures can be seen below in Figure 17. In those two figures, we find the NPV and TNR plotted separately, so we see the effect the increase of the threshold has on the two individual performance measures.

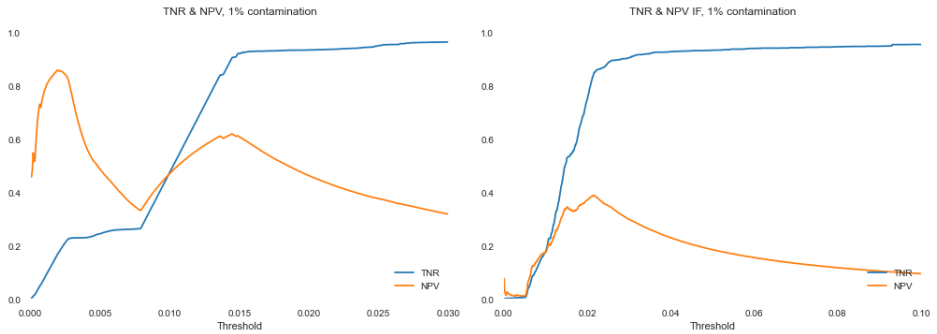

Figure 17: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value. The threshold value is the part of the dataset that is classified as outlier ranging from zero to one. One Class Classification model on the KDDcup99 dataset left and Isolation Forest model trained on the ISCXIDS2012 dataset right.

The most difficult part for the hospital is setting the threshold. Both outlier detection models score each point in the dataset. The lowest scoring points have the highest possibility of being an outlier. Ordering the data points on their score results in a list from most likely to be an outlier to most likely to be an inlier. The trade-off stated in the question is the trade-off between (i) finding all anomalies,

so suspicious accesses that might be inappropriate accesses to patient records, and (ii) keeping the FN as low as possible because FN means extra unnecessary work, thus costs, for the hospital. More FNs means more work, the hospital needs to investigate more accesses. Such investigation is difficult and time-consuming.

Keeping the threshold low will end in a relatively high NPV, but a low TNR, ideally at the top of the orange line of Figure 17. A high NPV means less FNs, but because of the low TNR, lots of inappropriate accesses are missed. Increasing the threshold increases the TNR, but the NPV decreases. If the TNR is approaching the one, almost all inappropriate accesses are found, but the number of FNs increases too.

6 Case study

As the previous section has focused solely on the control datasets, the focus of the next section is on the case study. The structure of this section corresponds with the previous section. We start with the evaluation plan, show the results, and end with discussing those results.

6.1 Evaluation plan

In this section, we discuss how we evaluate the results from the case study. For the hospital EHR logging data set, the labels are unknown as opposed to the control datasets. Therefore, we apply our approach and evaluate the results with domain experts.

To evaluate the performance of the models, the initial plan was to use sample-based evaluation to approximate performance measures, such as the TNR and NPV, and evaluate their performance. However, after the first initial meeting, the labeling of the results proved to be more difficult than expected. A thoroughly and very time-consuming investigation is needed to tell if an access is appropriate or not and even then it is not always possible to say the assess is justified or not.

The difficulty starts with not knowing exactly why a model returns a particular log. Therefore, the reason has to be deduced from the features and context of the log. This context is limited at the moment, making it difficult to decide if that particular access is appropriate or not. Foremost, this way of auditing logs is new for the hospital.

Therefore, it is not possible for the expert team, consisting of the data protection officer, chief information security officer and, quality consultant care processes (next referred to as ' the hospital'), to review a given list of outliers. Instead, we use a scenario-based approach. The approach is the following. Based on the results we create use cases. The use cases are selected with two different methods. The first method is based on the score. The lowest scoring logs or log sessions per model can be used to create use cases. The second method is to take the top X lowest-scoring logs and calculate the times a certain employee or patient appears within this top X rows.

For each created use case we discuss with the hospital what the hospital would do if a model would return such result in practice.

6.2 Results

In this section, we discuss the initial results first. Then, we present an overview of the results. Next, we discuss the first use cases. Based on the discussion of the first use cases, we run the models again and create some more use cases. We end this section by ranking the use cases from most interesting to least interesting, according to two hospital employees.

6.2.1 Initial result

We have first applied the IF on the log sessions database. While linking the log sessions back to original log data, found out that the top 5 outliers are log sessions with an extreme number of logs. The first outlier session contains 300.000 accesses, the second outlier session contains 200.000 accesses to patient records. As explained before, a log session is created when an EHR instance is made, i.e. opening the EHR on a device. A log session is ended when the same EHR instance is closed again. Normally, a session takes no longer than a work-day. The EHR instance is closed by the employee at the end of its workday or automatically after several hours of inactivity. But, these outlier sessions span over multiple days, up to eighteen days.

Looking at the users of theses outliers sessions, the acronyms of these users all start with the same letters. Those users turn out to be eHealth portals where patients or general practitioners can consult certain information from the EHR.

For now, we exclude the eHealth portal users because this thesis focuses solely on internal data breaches. We have excluded the portal users from the data before once more applying the IF on the datasets and then the OCC model too. Table 11 shows the case study dataset sizes before and after the exclusion.

Table 11: Size case study datasets before excluding of eHealth portals and after exclusion of eHealth portals

Dataset	**Before exclusion**	**After exclusion**
Case study EHR logging data single logs	7.970.083	6.971.918
Case study ER logging data log sessions	20.184	20.166

6.2.2 Comparison models and datasets

In this section, we make a comparison between the results found by both models, the IF and OCC, and the two datasets, single logs, and log sessions. A cross-table with the number of identical outliers found by the different models and methods can be found in Table 15 of Appendix F.

Of the top 500 outliers from the single log dataset found by the IF and OCC models, two aspects stand out. First is the type of each log. The type can be 'I' for a normal view, 'O' for overview, and 'N' for emergency procedure (the BTG procedure). Of the top 500 outliers found by the OCC model, 499 used the BTG procedure. However, only 33 of the top 500 outliers, found by the IF model, used the BTG procedure. The second aspect is the difference in the time the outlier log happens. Most of the outliers found by the OCC occurred in the morning and

afternoon, 238 and 180 of the top 500 respectively, most outliers found by the IF occurred in the night and evening, 277 and 165 respectively.

6.2.3 Use cases

As we mentioned before, we use two different methods to select the use cases. The first method is based on the scores. We select the lowest-scoring outlier sessions from both the IF and OCC models and the lowest scoring logs from both models as use cases. The second method is to derive use cases from the top 500 outliers. From the top 500, we examine per feature the occurrence of a certain value, for example the times a particular employee or patient appears within the top 500, resulting in three other use cases. We discuss the seven use cases next.

We start by explaining the use case. We tried to provide some context about each log or session with information we can deduct from the database but are limited by the features. Then, we discuss what the hospital would do if the model would return such a result in practice.

Use case A: Extremely frequent accesses by a single physician. A physician has consulted 160 unique records in a single session. This session spans over two days, almost 27 hours in total. This session consists of the most unique patient consulted by this physician this month. We ask the hospital their opinion on whether this use case is suspicious or not.

Recommendation: it is suggested to further investigate the relationship between the physician and patient and activities when these accesses have taken place. However, this relationship can be difficult to find. Even, if the relationship is found and there is no known relationship, it does not mean the log is inappropriate. Without a known relationship the access can still be legitimate. For example, it is not necessary for a physician to be the primary practitioner of a patient to do a bedside round. The duration of the session could explain the excessive number of logs but the activities the physician does during this session matters too. Some activities require checking multiple records within a short time, such as an outward shift, checking orders, bedside rounds, or catch up on work. Without knowing all these factors it is impossible to decide whether this session contains an abnormal number of accesses to patient records.

Use case B: Extremely frequent accesses by a user with an unknown role. A user has consulted 478 unique patient records in a single session. This session started at 08:01 and ended at 15:53. This session consists of the most unique patient consulted by this user this month.

Recommendation: it is suggested to figure out the function of the user first. The function could explain the high number of unique patients. Then, find out the normal number of accesses for this particular function. The hospital could determine that this user is a member of the secretariat. It is ordinary for a member of the secretariat to access more unique patients per day than for example a physician.

Use case C: Rare single access. A user, without known function, consulted a patient at 23:02. This patient appears in total five times this month, all on the

same day. The patient is consulted by three different users, no nurse, in three different sessions.

Recommendation: it is suggested to try to further investigate the log because a possible explanation can be a visit to the emergency room. A visit to the emergency room would explain the number of logs and time, but normally a nurse should be one of those three users too.

Use case D: Rare single access using an emergency procedure (BTG). A user used the BTG procedure on a patient's record at eight in the morning. The session, containing this outlier log and one other log, is the only session from this user this month. The session takes four minutes in total. Also, there are no other logs from this patient or user. The function of this user is not known.

Recommendation: it is suggested to find out, as for use case B, the users' function first. In this case, the function does not explain this odd session. Therefore, further analysis is needed of why the BTG is used and why the user and patient have only occurred once this month.

Use case E: Extremely frequent occurring patient. Within the top 500 outliers, a single patient occurred 81 times. Those 81 logs are from twelve sessions and by six unique users. This patient number appears in 1143 sessions this month and is consulted by 275 unique users. Looking at the dates from each session, the patient is admitted for at least this whole month.

Recommendation: it is suggested to find out more about the patient. The hospital assumes this patient was admitted to the intensive care unit. While it is not common for a patient to lie in the intensive care unit for over a month, it can happen especially now during the Covid-19 pandemic. With three different shifts a day and constant surveillance explain the number of unique users and sessions.

Use case F: Extremely frequent occurring user. The sixth use case is based on the frequency that a user occurs in the top 500 outliers. This user, from the secretariat, occurs 75 outlier log rows. The log rows are all divided over two sessions. The first session, at the beginning of the evening, takes 18 minutes and within the 18 minutes, this user has consulted 47 unique patients. The second session, also at the beginning of the evening, takes 31 minutes, in which the user has consulted 6 patients.

Recommendation: as for use case A/B, it is suggested to investigate the activities when these accesses have taken place. The hospital can imagine some activities where the secretary goes through lots of records in a limited time, but not that much as found by the model. Moreover, the time of this session makes this outlier more interesting.

Use case G: Extremely frequent occurring user. The seventh use case is based on the occupancy of a user within the top 500 outliers of the OCC model. This user appears 71 times out of the 500. Those 71 outlier logs are divided over 16 sessions, this user has 17 sessions in total this month. However, other than at least a single outlier log per session for this user, we can not see anything special about the sessions or logs.

Recommendation: Sometimes it is not possible to tell what differentiates the behaviour of this user from others, due to the limited information. So, the hospital can not explain either why this user appears often as an outlier.

6.2.4 Use cases 2

From the discussion during the first round of use cases, we have learned that the session time can be important too. The session time of use case A could have been the cause of the excessive number of accesses. This excessive number of accesses can be rectified by the session time. So, we have included the session time as an extra feature of the log sessions database.

We have run the models again and extended the cross table, as can be seen in Table 16 of Appendix F, with the new results. The new results are presented to the hospital.

We choose to present four log sessions at the hospital. The log sessions are chosen based on the decision score of both models. The first two sessions are chosen because one session has the lowest decision score, so the highest chance of being an outlier, according to the IF model and the other one according to the OCC model. The remaining two sessions are chosen because the sessions are the lowest scoring top ten sessions by both models.

Use case H: Multiple outlier log sessions from the same user. Use case 8 is compiled from three different sessions. The sessions are put together because the sessions are all from the same user. This user is the same as to use case A. Now, with session time as a feature, it is determined that the session time is not the main reason why use case A is labeled an outlier, as thought before. Besides that, the other three sessions are in accordance with a 'normal' workday and are still labeled an outlier.

Recommendation: it is suggested to investigate the user, but without asking the user to justify its actions. Asking the user to justify their actions should only be done when the hospital is almost certain that this user has done something against their policy. Asking the user would harm the trust between the hospital and its employees. A possible explanation of this use case is that the account of this user is used by multiple employees and therefore used more and differently than only one employee would have used an account.

Use case I: Extremely frequent accesses within a limited time. The last use case is one user that generated 135 logs within four minutes for one patient. The function of this employee is not known.

Recommendation: it is suggested to first determine the function of the user and then try to explain the number of logs. It is not clear why this session is considered an outlier other than the number of logs within a limited time. The function of this user is missing. The function is important for the first context. There are activities or windows within the EHR that can explain this number of logs, but that depends on the function of this employee too.

Ranking Use cases: Because it is not possible to classify each use case as appropriate or inappropriate, we have asked the hospital to rank the use cases from

most interesting, i.e. most suspicious, to least interesting. The ranking of the use cases by two hospital employees can be seen in Table 12. The two employees both got different functions in the hospital.

Table 12: Ranking from (1) most interesting, i.e. suspicious, to (9) least interesting per use case and average score over both employees

	A	B	C	D	E	F	G	H	I
Employee 1	9	5	6	1	8	3	2	4	7
Employee 2	4	1	9	5	8	2	7	6	5
Average	6.5	3	7.5	3	8	2.5	4.5	5	6

6.3 Discussion

The discussion of the case study consists of three parts. First are the difficulties during the validation discussed, then the limitations of this study and recommendations for future research. We end this section, by answering the second sub-question "*to what extend can the machine learning models detect anomalous accesses when applied to real EHR logging data*".

6.3.1 Encountered problems during validation

The evaluation of the outlier detection models is the most difficult part of this thesis and is proven to be more difficult than expected. The validation of unsupervised machine learning models is a known problem. It is difficult and time-consuming to correct all the outliers found. Even correcting a small sample turns out to be challenging. This approach of finding inappropriate accesses is new for the hospital. There are no guidelines/policies available to assists the hospital in their judgement. The hospital has great difficulty in deciding whenever a single log is considered to be an inappropriate access or not. The ranking of use cases (Table 12) shows the importance of a guideline/policy. Two employees with different functions rank the use cases completely different, both got their idea of the characteristics of a suspicious log.

Next to the lack of a guideline/policy, the hospital has to deduce all the information from the log itself. Understandably, the hospital does not want to ask the user to explain himself about a log without a valid reason, other than the results of a model. Therefore, more information is needed about the log. The information the hospital needs can be divided into two. (i) The hospital wants to know is why a certain log is considered an outlier and another log is not. Unfortunately, this information is not possible with unsupervised machine learning. The only thing we know about an outlier is that that particular log is different from all other logs based on a combination of features we use. This phenomenon is also called 'the black box', data goes in, decisions come out, but whatever happens between the input and output is not always clear.

The second detail is (ii) the information available about a log. The more information we have about a log the better the hospital can validate the output. We only use the patient number, user ID, part-of-day, class ID of the EHR, and type of access. We can extract some more information out of the database to help the hospital in their judgement, such as the difference between the outlier sessions and other sessions of a certain user that month or other users that accessed a particular patient record that turns out to be an outlier. There exists an audit tool in the EHR that can help the hospital with more information about a log, but at this point, they do not have access to that tool yet.

Without the information of (i) and (ii) it is proven difficult to validate the results of the outlier detection models. The long-term objective of these outlier detection models, which are part of a bigger application, is to support the demands of the AP to proactively search for cases where patient records are inappropriately accessed. We are not able to approximate performance measures, such as the TNR of NPV, but we have found interesting use cases. A fully automated access violation system is far away, but the outlier detection model can direct the hospital to use cases that

are potentially inappropriate. By directing the hospital towards such use cases, the limited human resources will be used more efficiently.

6.3.2 Limitations and Recommendations

Ideally, we would use the features proposed by Boxwala et al. Boxwala et al. have extracted features that are likely to be useful to detect suspicious accesses, i.e. potentially inappropriate accesses, from EHR logging data [9]. If we compare the features we used to the features proposed by Boxwale et al., we are missing some important features, such as 'Is provider', 'Had recent visit', or 'Access on clinic day'. Also, we are missing more features constructed for a specific kind of inappropriate access such as 'Work in same department' (Employee can access a coworkers record) or 'Same family name' ('Same family name' works only with rare last names, so we do not expect this feature to work in practise). While such features would benefit the performance, extracting this information out of the EHR is proven difficult.

Some of this information can be made available to us in the future, but not all. There can be several reasons why we can not use all the information we think is needed, for example the function table within the EHR is almost empty as for the user roles. Another reason is the way the EHR works, it can be too difficult to extract some information. Furthermore, we are simply not (yet) allowed to access some information, due to the EHR data's highly sensitive nature.

As a side note, adding more information about the employee or patient can have a positive effect on the performance. But, adding information about the employee or patient invades their privacy too. This results in a more philosophical question: *to what extend can we invade someone's privacy to protect someone else's privacy?*. The privacy we invade by adding more features has proportional to the goal, finding internal data breaches.

Another limitation of this thesis is the hardware. The hardware will be upgraded soon, but for now we are only able to run one month of data. Outliers found in that month can be considered exceptional for that month but normal behaviour when looking at for example a whole year. Looking at use case E, a patient laying a whole month on the intensive care unit may only happen once a month but can happen multiple times within a year, so making this use case less of an outlier.

We expect to have access to the audit tool. With the use of the audit tool of the EHR, we can validate outlier better and easier and possibly can say more about the performance of the outlier detection models used. When enough data is validated we can even use supervised machine learning models as an addition to the current approaches.

We also want to find a way to include the eHealth portal. By including the portal we do not only find inappropriate logs from the inside but made from the outside too.

6.3.3 Second sub-question

We cannot answer the second sub-question, *to what extend can the machine learning models detect anomalous accesses when applied to real EHR logging data*, with quantitative measures. Labeling a log appropriate or inappropriate was not possible during our case study. Even labeling a small sample is proven to be challenging. There is not only a need for lots of contextual information, that is not available

up to now, but also excessive domain knowledge. Even among employees holding this domain knowledge, the belief of what characterises a suspicious log varies. Therefore, a hospital-wide policy is needed that specifies when a given log has to be inspected further.

Nevertheless, we consider the outlier detection models a step in the right direction. This idea of using machine learning to find inappropriate accesses has arisen from the demand of the AP to implement a proactive and intelligent way to find inappropriate accesses. Already, we have found interesting cases that we, without the help of the outlier detection models, never would have encountered and in that way closer to compliance with the demands of the AP.

7 Conclusion

This thesis aims to answer the question *given a set of unlabeled logging accesses to the electronic patient records of a hospital, which machine learning model performs the best to detect anomaly accesses to the patient records in terms of the trade-off between being as precise as possible and detecting all anomalies.* This question can be divided into two sub-questions.

The first question, *How accurate are machine learning models in detecting anomalies in terms of the trade-off between being precise and detecting all anomalies.*, is answered using unsupervised outlier detection models trained on labeled datasets. Therefore it was possible to see the trade-off between (i) finding all the finding all suspicious accesses, i.e. inappropriate accesses, and (ii) keeping the FNs low to avoid extra time-consuming work. From the four machine learning models used, the OCC and IF performed best. On average the IF performed better than the OCC on both control datasets, but the performance of the OCC is more stable. Changing the contamination has a different effect on each model. With an increase in the contamination, the performance of the IF model will slightly decrease, but the performance of the OCC increases. The performance of the outlier detection models applied to the SOM is worse compared to the performance without the SOM. The optimal parameter setting of the LOF is based on a set percentage of the dataset, this percentage differs between datasets. However, the LOF is computational too expensive to use.

We are unable to answer the second sub-question, *to what extend can the machine learning models find anomalous accesses when trained on real EHR logging data.* We could not validate the results of the outlier detection models. Even labeling a small sample is proven to be challenging. There is not only a need for excessive contextual information, but there is also great domain knowledge needed to audit the logs. Even among employees holding this domain knowledge, the belief of what characterises a suspicious log varies.

To build a practical application that can detect inappropriate accesses, a hospital-wide policy has to be developed first. This policy has to state not only when a returned log needs further investigation, but also how to do the investigation. We have proven with the control datasets that the model can detect internal data-breaches, but without a policy or guideline, the hospital cannot effectively use these outlier detection models.

Nevertheless, the outlier detection models have already discovered some interesting cases that otherwise, with the old method of sampling, would never have been found. The AP demands the implementation of a proactive and intelligent system to detect data breaches. By finding such use cases, the hospital is closer to compliance with the demands of the AP.

References

[1] Amir Ahmad and Lipika Dey. "A k-mean clustering algorithm for mixed numeric and categorical data". In: *Data & Knowledge Engineering* 63.2 (2007), pp. 503–527.

[2] *Anomaly Detection with Isolation Forests using H2O*. Nov. 2019. URL: https://www.h2o.ai/blog/anomaly-detection-with-isolation-forests-using-h2o/.

[3] Dario Antonelli, Giulia Bruno, and Silvia Chiusano. "Anomaly detection in medical treatment to discover unusual patient management". In: *IIE Transactions on Healthcare Systems Engineering* 3.2 (2013), pp. 69–77.

[4] Ajit Appari and M Eric Johnson. "Information security and privacy in healthcare: current state of research". In: *International journal of Internet and enterprise management* 6.4 (2010), pp. 279–314.

[5] Rafae Bhatti and Tyrone Grandison. "Towards improved privacy policy coverage in healthcare using policy refinement". In: *Workshop on Secure Data Management*. Springer. 2007, pp. 158–173.

[6] Alan Bivens et al. "Network-based intrusion detection using neural networks". In: *Intelligent Engineering Systems through Artificial Neural Networks* 12.1 (2002), pp. 579–584.

[7] Aaron Boddy et al. "Data Analysis Techniques to Visualise Accesses to Patient Records in Healthcare Infrastructures". In: (2018).

[8] A. Bounsiar and M. G. Madden. "Kernels for One-Class Support Vector Machines". In: *2014 International Conference on Information Science and Applications (ICISA)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2014, pp. 1–4. DOI: 10.1109/ICISA.2014.6847419. URL: https://doi-ieeecomputersociety-org.proxy.library.uu.nl/10.1109/ICISA.2014.6847419.

[9] Aziz A Boxwala et al. "Using statistical and machine learning to help institutions detect suspicious access to electronic health records". In: *Journal of the American Medical Informatics Association* 18.4 (2011), pp. 498–505.

[10] Markus M Breunig et al. "LOF: identifying density-based local outliers". In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 2000, pp. 93–104.

[11] Guilherme O Campos et al. "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study". In: *Data Mining and Knowledge Discovery* 30.4 (2016), pp. 891–927.

[12] Adrian DC Chan et al. "Hidden Markov model classification of myoelectric signals in speech". In: *IEEE Engineering in Medicine and Biology Magazine* 21.5 (2002), pp. 143–146.

[13] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[14] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 27:1–27:27.

[15] You Chen and Bradley Malin. "Detection of anomalous insiders in collaborative environments via relational analysis of access logs". In: *Proceedings of the first ACM conference on Data and application security and privacy*. 2011, pp. 63–74.

[16] National Research Council et al. *For the record: protecting electronic health information*. National Academies Press, 1997. Chap. 1.

[17] Doedotdev. *doedotdev/local$_o$utlier$_f$actor*. Nov. 2018. URL: `https://github.com/doedotdev/local_outlier_factor`.

[18] Eleazar Eskin et al. "A geometric framework for unsupervised anomaly detection". In: *Applications of data mining in computer security*. Springer, 2002, pp. 77–101.

[19] Meghan Hufstader Gabriel et al. "Data breach locations, types, and associated characteristics among US hospitals". In: *Am J Manag Care* 24.2 (2018), pp. 78–84.

[20] Michael Hanke et al. "PyMVPA: a unifying approach to the analysis of neuroscientific data". In: *Frontiers in neuroinformatics* 3 (2009), p. 3.

[21] *Hidden Markov Models¶*. URL: `http://scikit-learn.sourceforge.net/stable/modules/hmm.html`.

[22] Victoria Hodge and Jim Austin. "A survey of outlier detection methodologies". In: *Artificial intelligence review* 22.2 (2004), pp. 85–126.

[23] *Hoe goed is mijn ziekenhuis?* URL: `https://www.ziekenhuischeck.nl/`.

[24] Jmschrei. *jmschrei/pomegranate*. URL: `https://github.com/jmschrei/pomegranate`.

[25] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. "Detecting anomalous access patterns in relational databases". In: *The VLDB Journal* 17.5 (2008), pp. 1063–1077.

[26] Sevvandi Kandanaarachchi et al. "On normalization and algorithm selection for unsupervised outlier detection". In: *Data Mining and Knowledge Discovery* 34.2 (2020), pp. 309–354.

[27] *KDDcup1999*. URL: `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`.

[28] Shehroz S Khan and Michael G Madden. "One-class classification: taxonomy of study and review of techniques". In: *The Knowledge Engineering Review* 29.3 (2014), pp. 345–374.

[29] Teuvo Kohonen. "The self-organizing map". In: *Proceedings of the IEEE* 78.9 (1990), pp. 1464–1480.

[30] Hans-Peter Kriegel et al. "Interpreting and unifying outlier scores". In: *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM. 2011, pp. 13–24.

[31] Hsuan-Tien Lin and Chih-Jen Lin. "A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods". In: *submitted to Neural Computation* 3 (2003), pp. 1–32.

[32] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. ICDM '08. USA: IEEE Computer Society, 2008, pp. 413–422. ISBN: 9780769535029. DOI: 10.1109/ICDM.2008.17. URL: https://doi.org/10.1109/ICDM.2008.17.

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

[34] *Mammography dataset*. URL: http://odds.cs.stonybrook.edu/mammography-dataset/.

[35] Swapneel Mehta, Prasanth Kothuri, and Daniel Lanza Garcia. "Anomaly Detection for Network Connection Logs". In: *arXiv preprint arXiv:1812.01941* (2018).

[36] Aditya Krishna Menon et al. "Detecting inappropriate access to electronic health records using collaborative filtering". In: *Machine learning* 95.1 (2014), pp. 87–101.

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

[38] Hoang Vu Nguyen and Vivekanand Gopalkrishnan. "Feature extraction for outlier detection in high-dimensional spaces". In: *Feature Selection in Data Mining*. 2010, pp. 66–75.

[39] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[40] Leonid Portnoy. "Intrusion detection with unlabeled data using clustering". PhD thesis. Columbia University, 2000.

[41] Felix M Riese, Sina Keller, and Stefan Hinz. "Supervised and Semi-Supervised Self-Organizing Maps for Regression and Classification Focusing on Hyperspectral Data". In: *Remote Sensing* 12.1 (2020), p. 7.

[42] Jacob Schreiber. "Pomegranate: fast and flexible probabilistic modeling in python". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 5992–5997.

[43] Erich Schubert et al. "On evaluation of outlier rankings and outlier scores". In: *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM. 2012, pp. 1047–1058.

[44] Sevamoo. *sevamoo/SOMPY*. Feb. 2020. URL: https://github.com/sevamoo/SOMPY.

[45] Andrii Shalaginov and Katrin Franke. "A new method for an optimal som size determination in neuro-fuzzy for the digital forensics applications". In: *International Work-Conference on Artificial Neural Networks*. Springer. 2015, pp. 549–563.

[46] Yurii Shevchuk. URL: http://neupy.com/2017/12/09/sofm_applications.html.

[47] Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim. "One-class support vector machines—an application in machine fault detection and classification". In: *Computers & Industrial Engineering* 48.2 (2005), pp. 395–408.

[48] Ali Shiravi et al. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection". In: *computers & security* 31.3 (2012), pp. 357–374.

[49] *sklearn.ensemble.IsolationForest¶*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

[50] *sklearn.neighbors.LocalOutlierFactor¶*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html.

[51] *sklearn.svm.OneClassSVM¶*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html.

[52] R. Smith et al. "Clustering approaches for anomaly based intrusion detection". In: *Proceedings of Intelligent Engineering Systems Through Artificial Neural Networks* (Jan. 2002), pp. 579–584.

[53] Alex J Smola, Bernhard Schölkopf, and Klaus-Robert Müller. "The connection between regularization operators and support vector kernels". In: *Neural networks* 11.4 (1998), pp. 637–649.

[54] Robin Sommer and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection". In: *2010 IEEE symposium on security and privacy*. IEEE. 2010, pp. 305–316.

[55] Sebastiaan A Terwijn. "On the learnability of hidden Markov models". In: *International Colloquium on Grammatical Inference*. Springer. 2002, pp. 261–268.

[56] "Tientallen onbevoegden bekeken medisch dossier Barbie". In: *NOS* (Apr. 2018). URL: https://nos.nl/artikel/2225867-tientallen-onbevoegden-bekeken-medisch-dossier-barbie.html.

[57] *Welcome to H2O 3¶*. URL: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html.

[58] Wan-Shiou Yang and San-Yih Hwang. "A process-mining framework for the detection of healthcare fraud and abuse". In: *Expert Systems with Applications* 31.1 (2006), pp. 56–68.

[59] Tahera Yesmin and Michael W Carter. "Evaluation framework for automatic privacy auditing tools for hospital data breach detections and an application case". In: *International Journal of Medical Informatics* (2020), p. 104123.

[60] Stephanie O Zandieh et al. “Challenges to EHR implementation in electronic- versus paper-based office practices”. In: *Journal of general internal medicine* 23.6 (2008), pp. 755–761.

[61] Yue Zhao, Zain Nasrullah, and Zheng Li. “PyOD: A Python Toolbox for Scalable Outlier Detection”. In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: `http://jmlr.org/papers/v20/19-011.html`.

[62] Arthur Zimek, Erich Schubert, and Hans-Peter Kriegel. “A survey on unsupervised outlier detection in high-dimensional numerical data”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 5.5 (2012), pp. 363–387.

List of Figures

List of Tables

Appendices

A Preprocesses

A.1 Categorical data handling

Three different approaches for category handling are mentioned in section 3.5.2, one-hot encoding, IDF, and removing the categorical features. To decide which method to use, we test the performance on the ten percent KDDcup99. The three outlier detection models are trained with input data handled in three different ways. The parameters used: for the OCC model a threshold of 0.01 and the parameter of the LOF model is set on the 20 nearest neighbors.

Looking at the TNR/FNR AUC score there is little to no difference in performance. The plots can be found as Figure 18. However, the training on the 1-to-n encoding data takes on average 3.7 times longer than the other 2 methods, because of the 42 extra features. Leaving the categorical variables out does have a negative effect on the performance of the IF model. Thus, based on these results we choose the IDF method as a method to convert categorical variables.

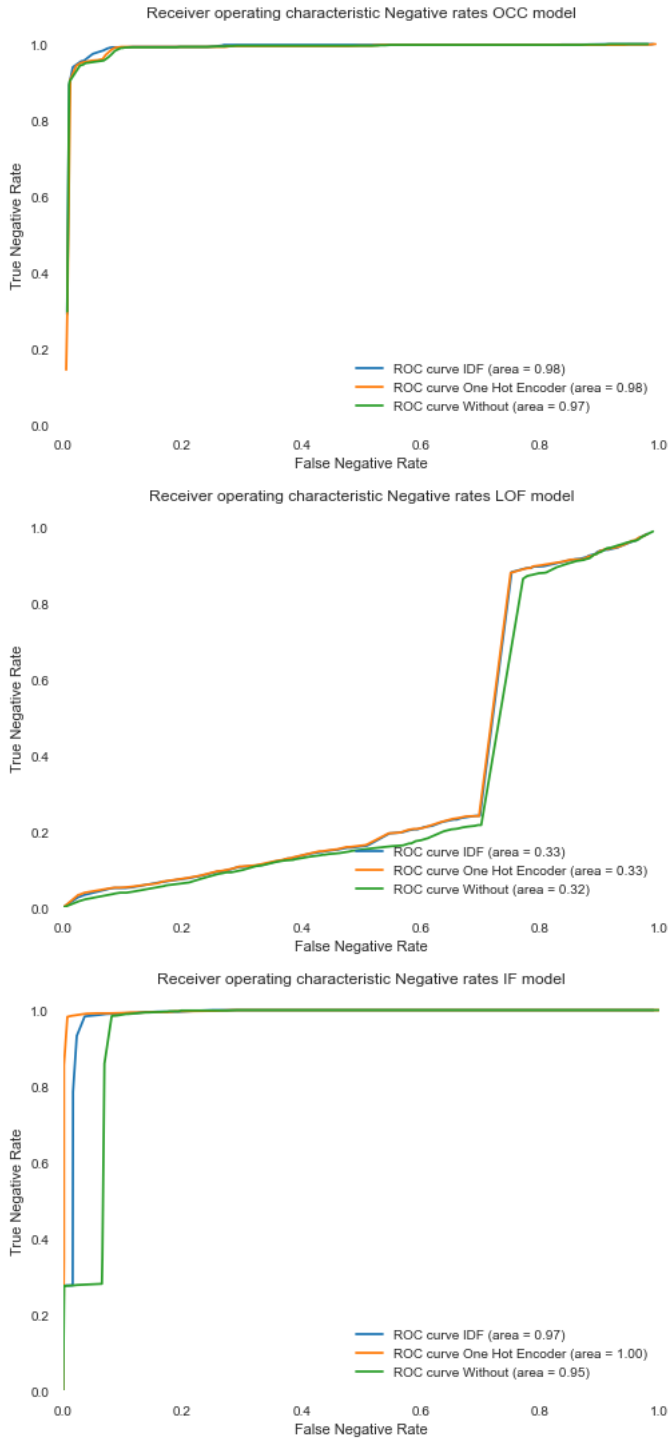

Figure 18: Performance of three different methods to convert categorical data into numeric (Area Under the Curve scores). Three outlier detection models used: Local Outlier Factor, One Class Classification, and Isolation Forest. Dataset used: ten percent KDDcup99

A.2 Package selection

In this section are the results of the different packages compared. All packages for all models are based on the same principle. Most of the time a package uses the other package that we test as the foundation. We do not expect any significant variation in performance. We do not only look at performance but at run-time too. In practice, the run time can be a liming factor for a successful implementation. Therefore, we attempt to keep the run time to a minimum as long as this does not compromise the performance by more than 0.05.

A.2.1 LOF

Two packages are selected for comparison, the Scikit-learn package and PyOD package. Both packages need the density parameter 'n_neighbors'. In this phase, we train the models on 10 and 20 neighbors. A threshold is needed for the PyOD packages, while a threshold is optional for the sklearn package. If not provided the threshold is set on 'auto' and determined by the Scikit-learn package as in the original paper [10]. The threshold is set on 0.01 for the PyOD package and for the Scikit-learn on auto. While the threshold has to bet set to train the models, the setting does not affect ROC curves. The ROC curves are computed by taking various thresholds. The result of the comparison can be found in Figure 19.

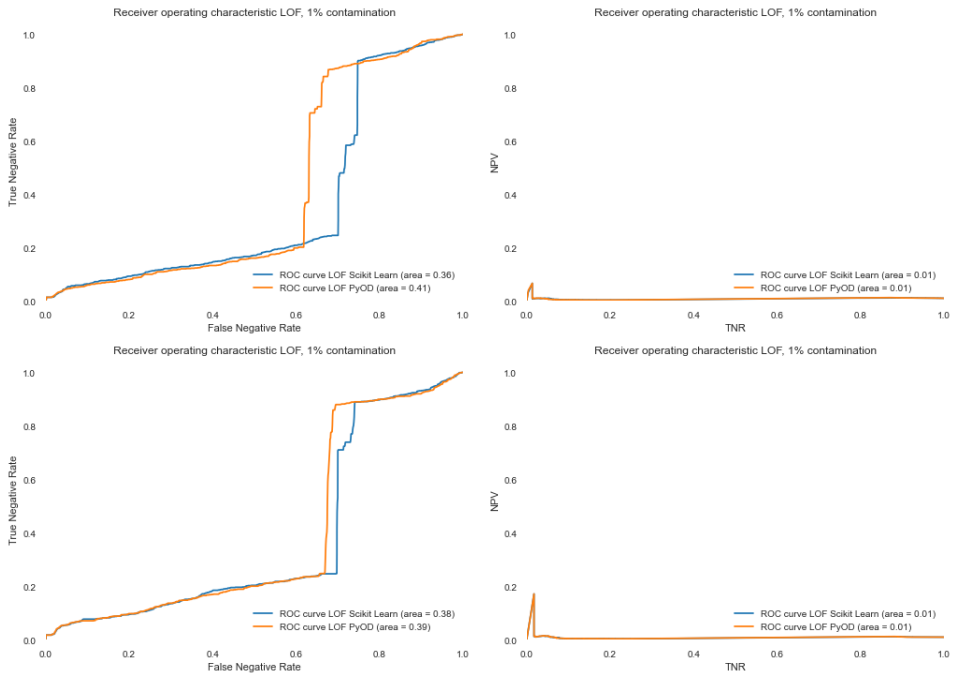

Figure 19: Performance two different packages to build a Local Outlier Factor (LOF) model. Number of neighbours parameter is set on ten for the figures above and is set on twenty for the figures below. True Negative Rate (TNR) and Negative Predictive Value (NPV). Dataset used: ten percent KDDcup99

As seen in Figure 19, the PyOD packages performs slightly better. This small

improvement can be explained by the fact that PyOD a wrapper is of sklearn with more functionalities. Even though the PyOD package performs better than the Scikit-learn package, the PyOD package takes two times longer to train, approximately 600 seconds as opposed to the 300 seconds from the Scikit-learn package. The complexity of the model is polynomial, so this difference is something to consider. So, while the PyOD package performs better, but overall poor for both packages, a TNR/FNR AUC score smaller than 0.5, the run time for a LOF is already high compared to other models and both packages are based on the same principle, we choose the Scikit-learn package over the PyOD package.

A.2.2 OCC

First, are the three different packages, Scikit-learn, LibSVM, and PyOD, trained on the ten-percent KDDcup99 dataset with a contamination of one percent. The 'nu' parameter is set on 0.01. The TNR/FNR and NPV/TNR AUC plots can be found in Figure 20.

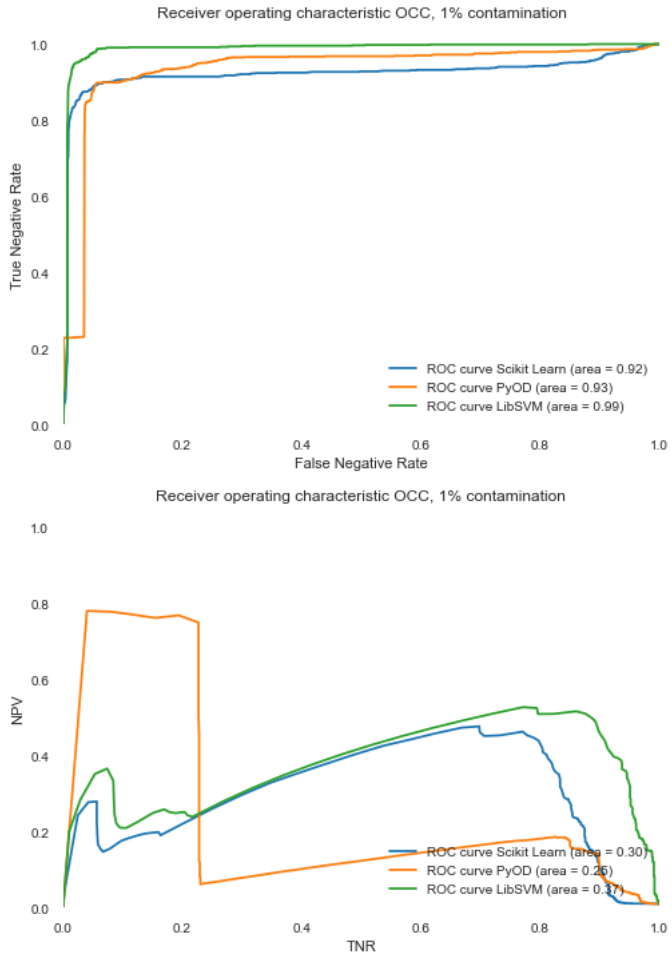

Figure 20: Performance three different packages to build an One Class Classification (OCC) model. True Negative Rate (TNR) and Negative Predictive Value (NPV). Dataset used: ten percent KDDcup99

The first noticeable difference, is the difference in run time 184 seconds for Scikit Learn, 2097 seconds for PyOD, and 39 seconds for LibSVM. While the LibSVM package is the fastest, LibSVM is also the best scoring one. So therefore we are going to use the LibSVM package for the OCC model.

A.2.3 IF

Three packages are initially found to build an isolation forest. Isolation forest uses random sampling, so comparing the packages can be difficult. However, is it possible to set the Scikit-learn and PyOD packages on the same random state, unfortunately this can not be done with the h2o package. Comparing the packages is still possible through i.e. bagging, but considering the results from the sklearn and the PyOD package, it is not necessary. The results can be found in Figure 21. The isolation forests are trained on the ten percent KDDcup99 with a contamination of one.

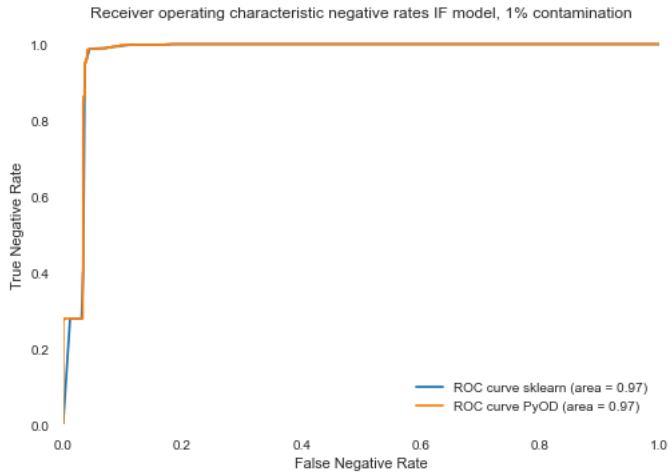

Figure 21: Performance two different packages to build an Isolation Forest (IF) model. Dataset used: ten percent KDDcup99

The performance of the two packages are the same, but the PyOD package takes 1.5 times longer to train. Therefore, we use the Scikit-learn package.

A.2.4 SOM

Four packages are chosen: Minisom, NeuPy, Susi, and PyMVPA. During implementation some problems occur installing PyMVPA. Because of the similar results are obtained from the other three packages, that can be seen later on, we exclude PyMVPA. we do not expect a difference in performance that out weights the extra time and effort needed to successfully implement the PyMVPA package.

The Minisom package is trained on the previously mentioned parameters: grid size 39, learning-rate of 0.01, and 10.000 epochs. The training takes about a minute. Then, the NeuPy package is run on the same parameters. However, after 72 hours the training was still not complete. Therefore, the number of epochs is changed to 100 instead of 10.000. The training takes around 1.75 hours. At last, the Susi package is trained on a grid size of 39 and 10.000. Where it is possible for the other two packages to set the learning rate, it is not possible for the Susi package. The way Susi set the learning rate is stated in their paper [41]. Training the Susi SOM takes around 1.5 minutes.

The new coordinates are used to train an IF model. The IF model is chosen because of, a TNR/FNR AUC score greater than 0.5 and training the model results in the same outcome each run if the random state is kept the same. The TNR/FNR curves of the IF model trained on the various SOMs and trained without a SOM on the ten percent KDDcup99 dataset can be found in Figure 22.

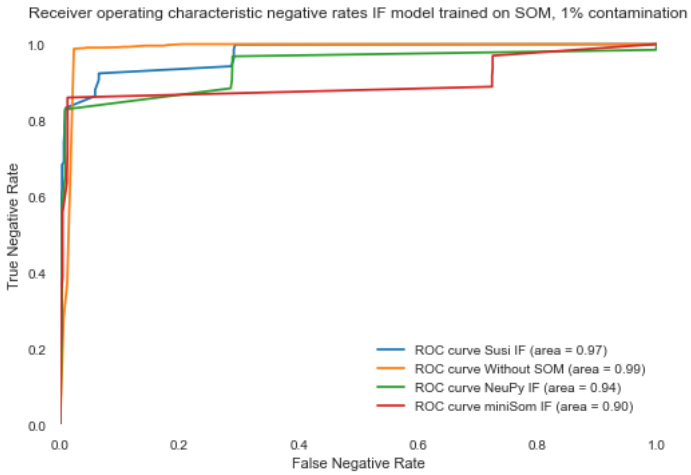

Figure 22: Performance Isolation Forest model three different packages to build an Self-Organising Map (SOM) and without the SOM. Dataset used: ten percent KDDcup99

Based on that all packages are using the same principle, run time, and the results from Figure 22, we have selected the Susi package.

A.3 Parameter selection

In this section, we compare the different parameter settings for the LOF model on both control datasets. The outcome, in combination with literature, is used to choose the best parameter setting for the LOF model.

A.3.1 LOF trained on KDDcup99

Before training the LOF model on the entire dataset we need to determine the k nearest neighbor parameter. According to the original paper choosing the number of neighbours between the ten and twenty should work well [10]. So first, we trained the LOF model on the ten percent KDDcup99 dataset. We use various kNN settings ranging from five to thirty to cover the suggested ranged and more. The results can be found in Figure 23.

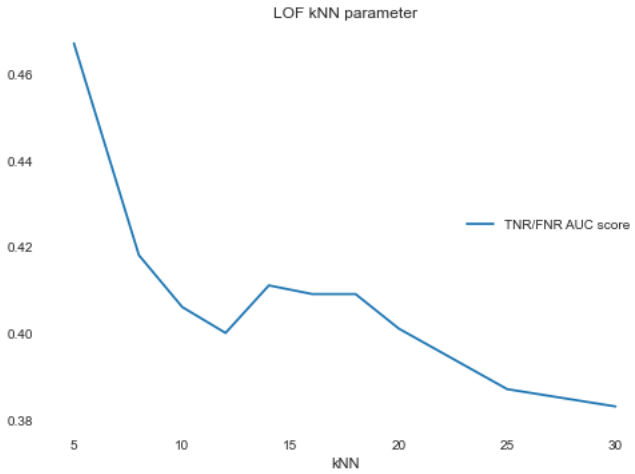

Figure 23: Performance of Local Outlier Factor (LOF) model on various k nearest neighbor (kNN) settings. True Negative Rate (TNR), False Negative Rate (FNR), and AUC (Area Under the Curve). Dataset used: ten percent KDDcup99.

All parameter settings results in a TNR/FNR AUC score less than 0.5, meaning the models score worse than chance. Then, we raise the number of neighbour to far over the suggested range. We create an even smaller subset than the ten percent KDDcup99, this is necessary because of the high computation costs of the LOF model. The computational costs increases with larger datasets or a raised kNN parameter. We create multiple subsets to find a correlation between the size of the set and the extent of the kNN parameter. The sizes of the subset are 10.000 rows, 20.000 rows, and 40.000 rows. The performance of the LOF models on those three subsets is tested with a kNN parameter ranging from 1% of the subset up to 11% of the subset. The results of the NPV/TNR curve can be seen in Figure 24. The TNR/FNR curve and table with precise data can be found in Figure 23 and Table 13.

Table 13: Performance Local Outlier Factor model on different k-Nearest Neighbours (kNN) parameter settings and different size datasets (10.000, 20.000 and 40.000 rows). Area Under the Curve score True Negative Rate plotted against False Negative Rate (TNR/FNR) and Area Under the Curve score Negative Predictive Value plotted against True Negative Rate (NPV/TNR). Dataset used: KDDcup99.

	10.000		20.000		40.000	
***k*NN**	**TNR/FNR**	**NPV/TNR**	**TNR/FNR**	**NPV/TNR**	**TNR/FNR**	**NPV/TNR**
1%	0.840	0.031	0.844	0.032	0.844	0.032
2%	0.903	0.057	0.904	0.056	0.902	0.054
3%	0.911	0.062	0.914	0.063	0.913	0.062
4%	0.923	0.082	0.924	0.083	0.927	0.083
5%	0.930	0.142	0.933	0.142	0.937	0.144
6%	0.916	0.141	0.907	0.140	0.921	0.142
7%	0.921	0.142	0.911	0.141	0.924	0.143
8%	0.924	0.143	0.916	0.143	0.926	0.143
9%	0.925	0.143	0.916	0.143	0.927	0.143
10%	0.915	0.082	0.905	0.081	0.898	0.059
11%	0.867	0.036	0.860	0.036	0.860	0.035

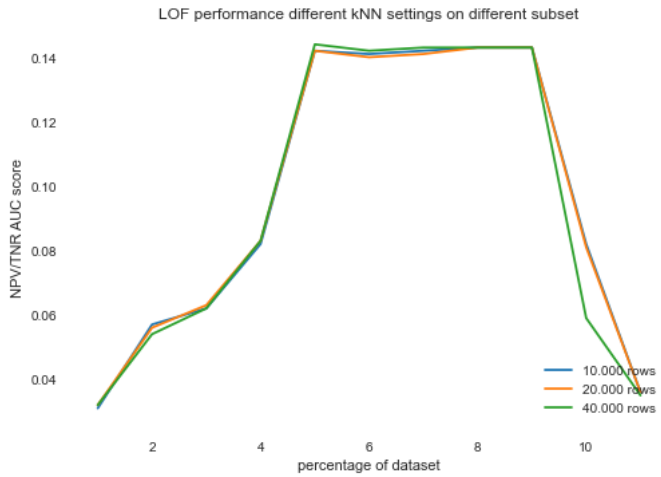

Figure 24: Performance of Local Outlier Factor (LOF) model with k nearest neighbor (kNN) parameter set on various percentages of dataset. True Negative Rate (TNR), Negative Predictive Value (NPV), and AUC (Area Under the Curve). Dataset used: randomly selected subsets of 10.000, 20.000 and 40.000 rows of the KDDcup99.

The performance of the three subsets are quite similar. Looking at the NPV/TNR AUC score, the optimal parameter setting should lie between the 5% and 9%. Further analysis shows that the optimal parameter setting is 7.5% for all three subsets.

A.3.2 LOF trained on ISCXIDS2012

After the optimal parameter is found for the KDDcup99, we do the same for the ISCXIDS2012 set. The results from the KDDcup99 shows that the optimal parameter setting is a percentage of the entire set. To validate this assumption we

test the performance of different kNN values on two different subsets of the ISCX-IDS2012 set. As for the KDDcup99, we first take 10% of the original downsized ISCXIDS2012. The second subset contains 12.500 rows. The NPV/TNR curve can be seen in Figure 25. The table with precise data can be found in Table 14.

Table 14: Performance Local Outlier Factor model on different k-Nearest Neighbours (kNN) parameter settings and different size datasets (12.500 and 25.000 rows). Area Under the Curve score True Negative Rate plotted against False Negative Rate (TNR/FNR) and Area Under the Curve score Negative Predictive Value plotted against True Negative Rate (NPV/TNR). Dataset used: ISCXIDS2012.

	12.500		25.000	
***k*NN**	**TNR/FNR**	**NPV/TNR**	**TNR/FNR**	**NPV/TNR**
5%	0.977	0.425	0.981	0.535
7.5%	0.977	0.551	0.982	0.692
10%	0.978	0.625	0.983	0.770
12.5%	0.981	0.723	0.984	0.809
15%	0.981	0.729	0.984	0.814
17.5%	0.982	0.722	0.984	0.801
20%	0.982	0.70	0.984	0.789
22.5%	0.980	0.658	0.982	0.758

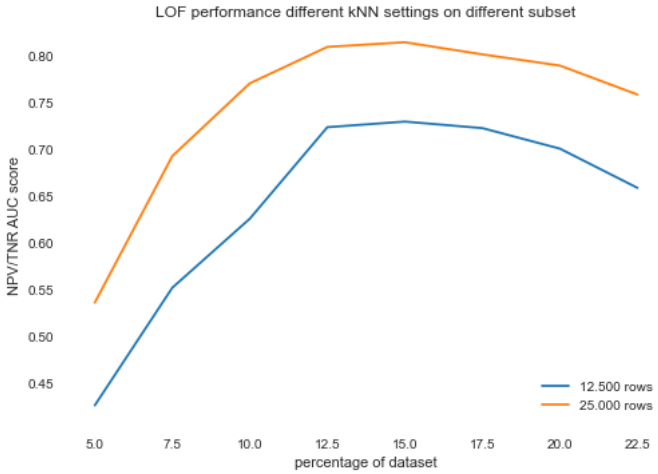

Figure 25: Performance of Local Outlier Factor (LOF) model with k nearest neighbor (kNN) parameter set on various percentages of dataset. True Negative Rate (TNR), Negative Predictive Value (NPV), and AUC (Area Under the Curve). Dataset used: randomly selected subsets of 12.500 and 25.000 rows of the KDDcup99.

The results show that the optimal LOF parameter setting is 15% of the dataset for both subsets.

B Results LOF

B.1 KDDcup99

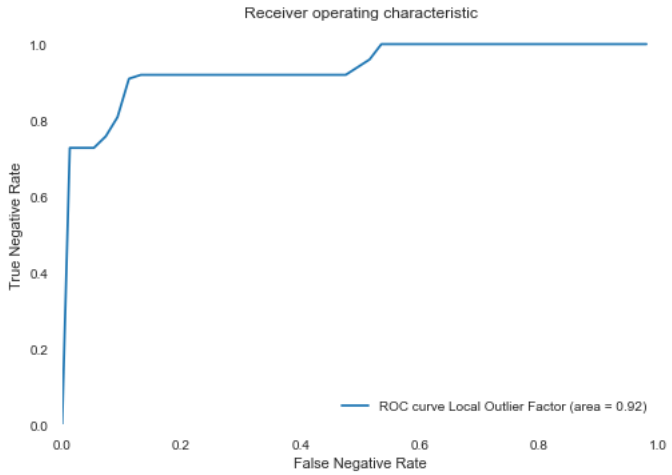

Figure 26: False Negative Rate/True Negative Rate curve Local Outlier Factor (LOF) model. Dataset used: KDDcup99

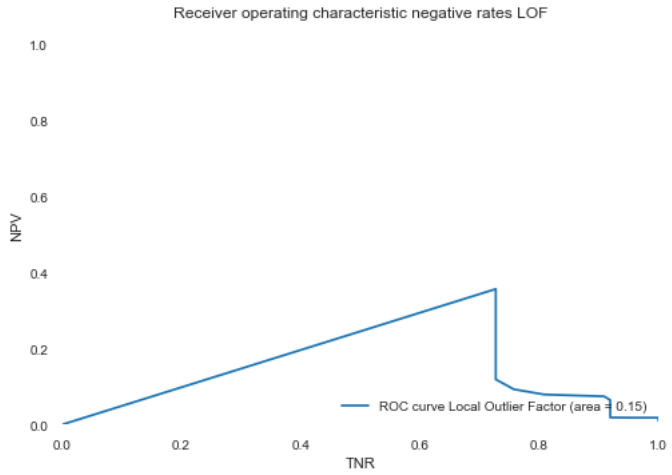

Figure 27: Negative Predictive Value/True Negative Rate curve Local Outlier Factor (LOF) model. Dataset used: KDDcup99

B.2 ISCXIDS2012

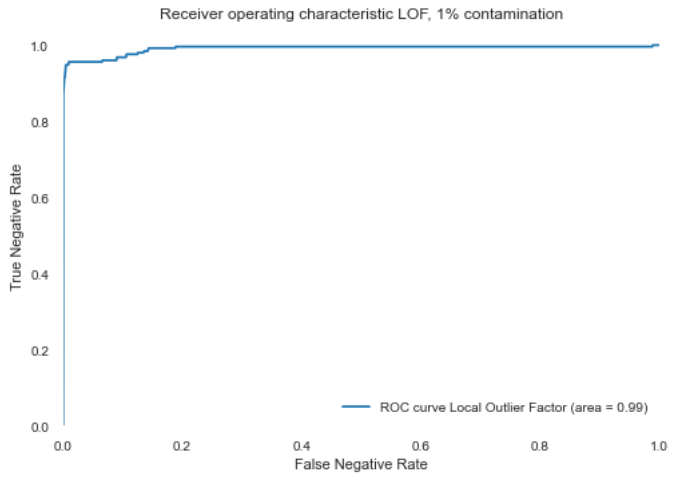

Figure 28: False Negative Rate/True Negative Rate curve Local Outlier Factor (LOF) model. Dataset used: ISCXIDS2012

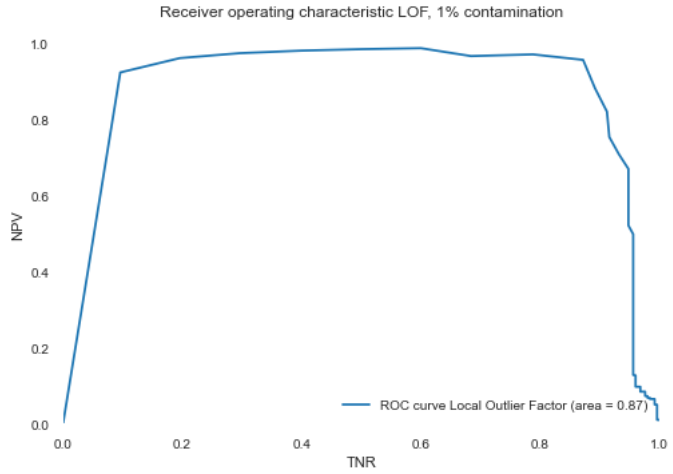

Figure 29: Negative Predictive Value/True Negative Rate curve Local Outlier Factor (LOF) model. Dataset used: ISCXIDS2012

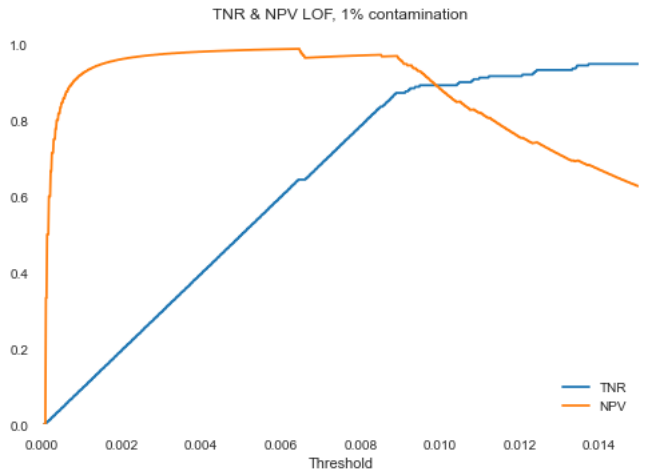

Figure 30: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value Local Outlier Factor (LOF) model, Dataset used: ISCXIDS2012

C Results OCC

C.1 KDDcup99

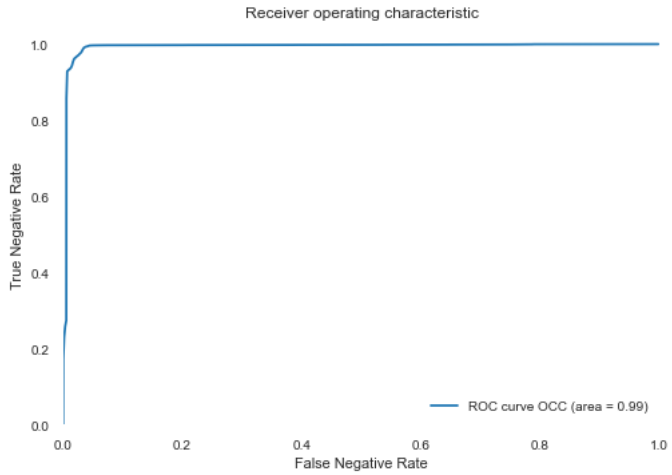

Figure 31: False Negative Rate/True Negative Rate curve One Class Classification (OCC) model. Dataset used: KDDcup99

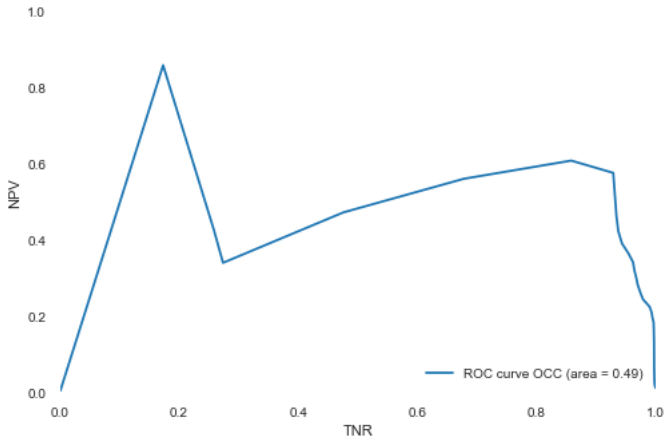

Figure 32: Negative Predictive Value/True Negative Rate curve One Class Classification (OCC) model. Dataset used: KDDcup99

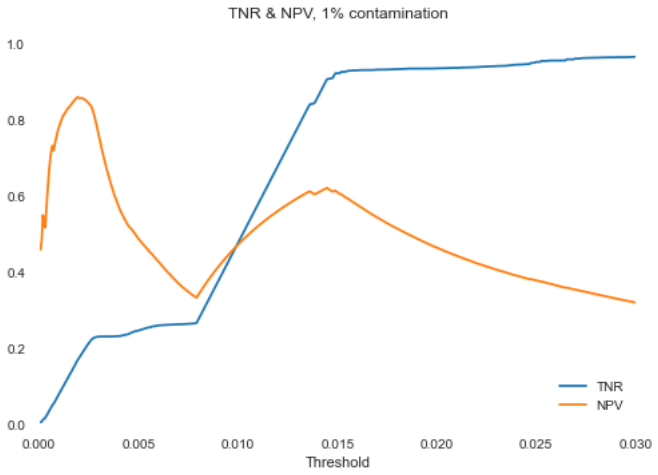

Figure 33: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value One Class Classification (OCC) model, Dataset used: KDDcup99

C.2 ISCXIDS2012

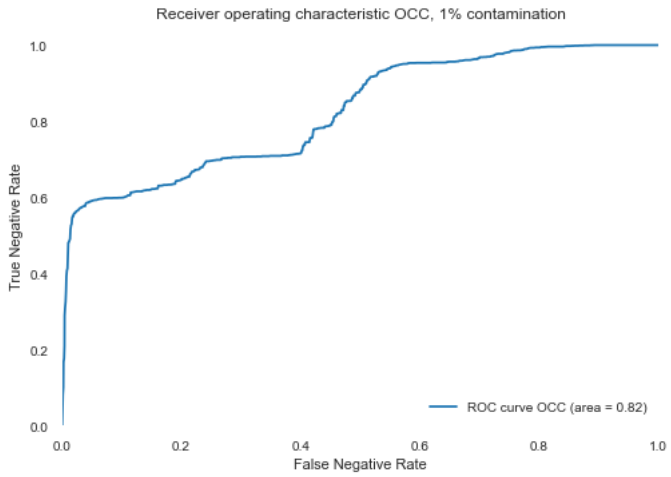

Figure 34: False Negative Rate/True Negative Rate curve One Class Classification (OCC) model. Dataset used: ISCXIDS2012

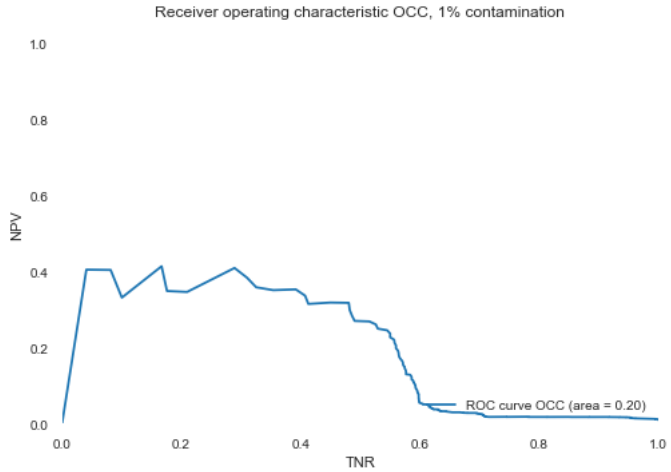

Figure 35: Negative Predictive Value/True Negative Rate curve One Class Classification (OCC) model. Dataset used: ISCXIDS2012

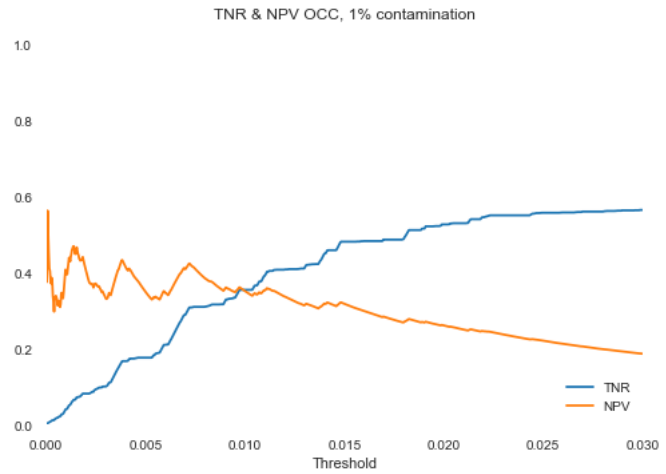

Figure 36: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value One Class Classification (OCC) model, Dataset used: ISCXIDS2012

D Results IF

D.1 KDDcup99

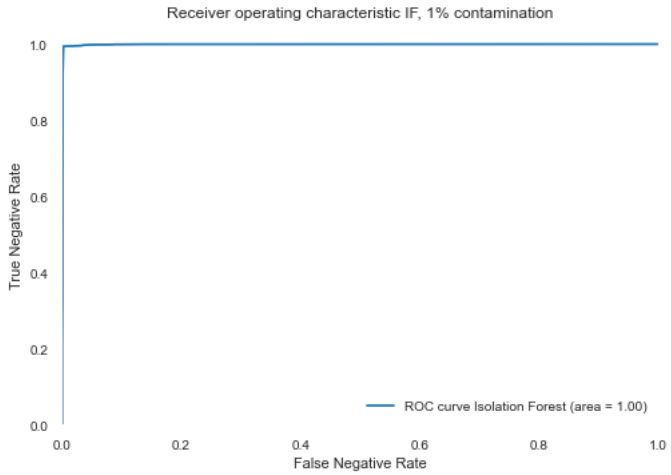

Figure 37: False Negative Rate/True Negative Rate curve Isolation Forest (IF) model. Dataset used: KDDcup99

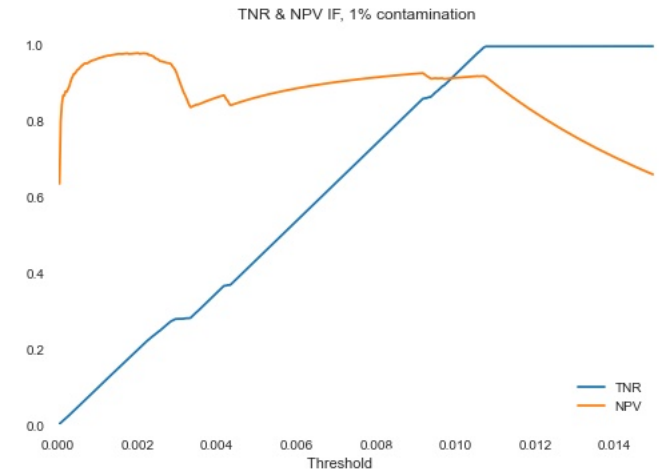

Figure 38: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value Isolation Forest (IF) model, Dataset used: KDDcup99

D.2 ISCXIDS2012

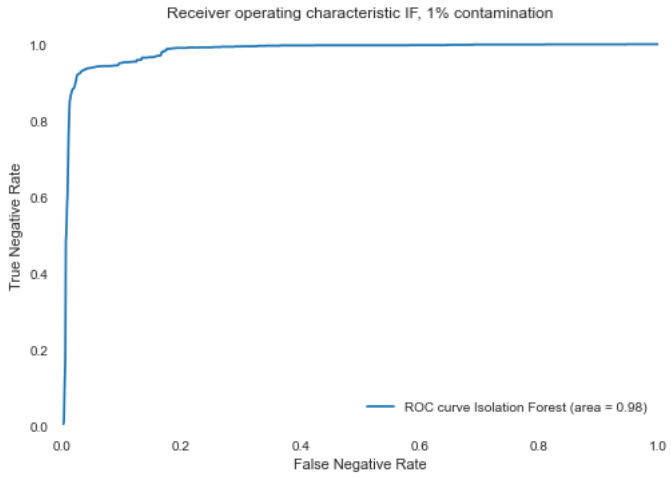

Figure 39: False Negative Rate/True Negative Rate curve Isolation Forest (IF) model. Dataset used: ISCXIDS2012

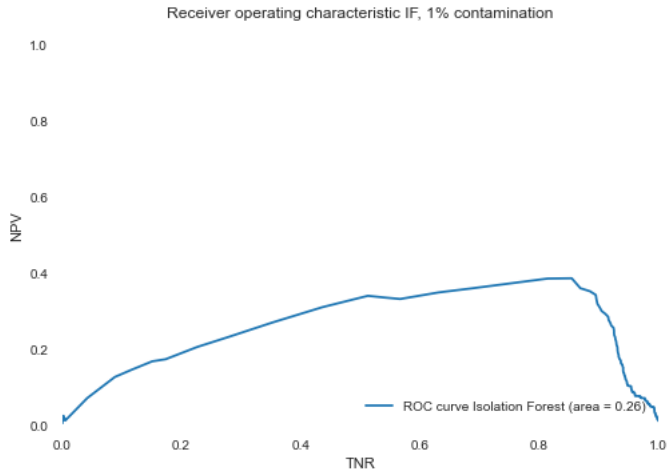

Figure 40: Negative Predictive Value/True Negative Rate curve Isolation Forest (IF) model. Dataset used: ISCXIDS2012

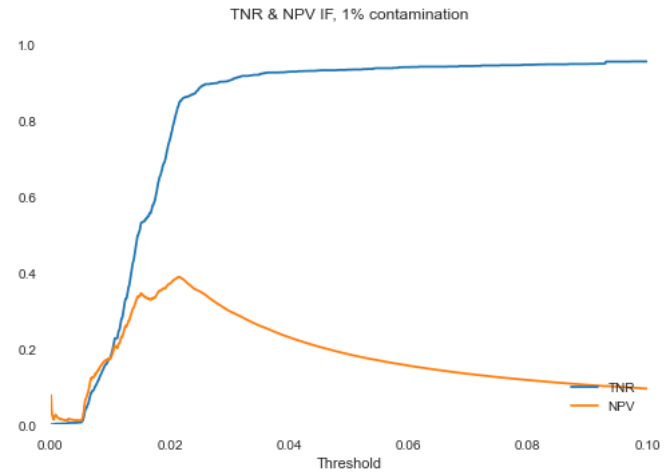

Figure 41: Negative Predictive Value (NPV) and True Negative Rate (TNR) per threshold value Isolation Forest (IF) model, Dataset used: ISCXIDS2012

E Results SOM

E.1 KDDcup99

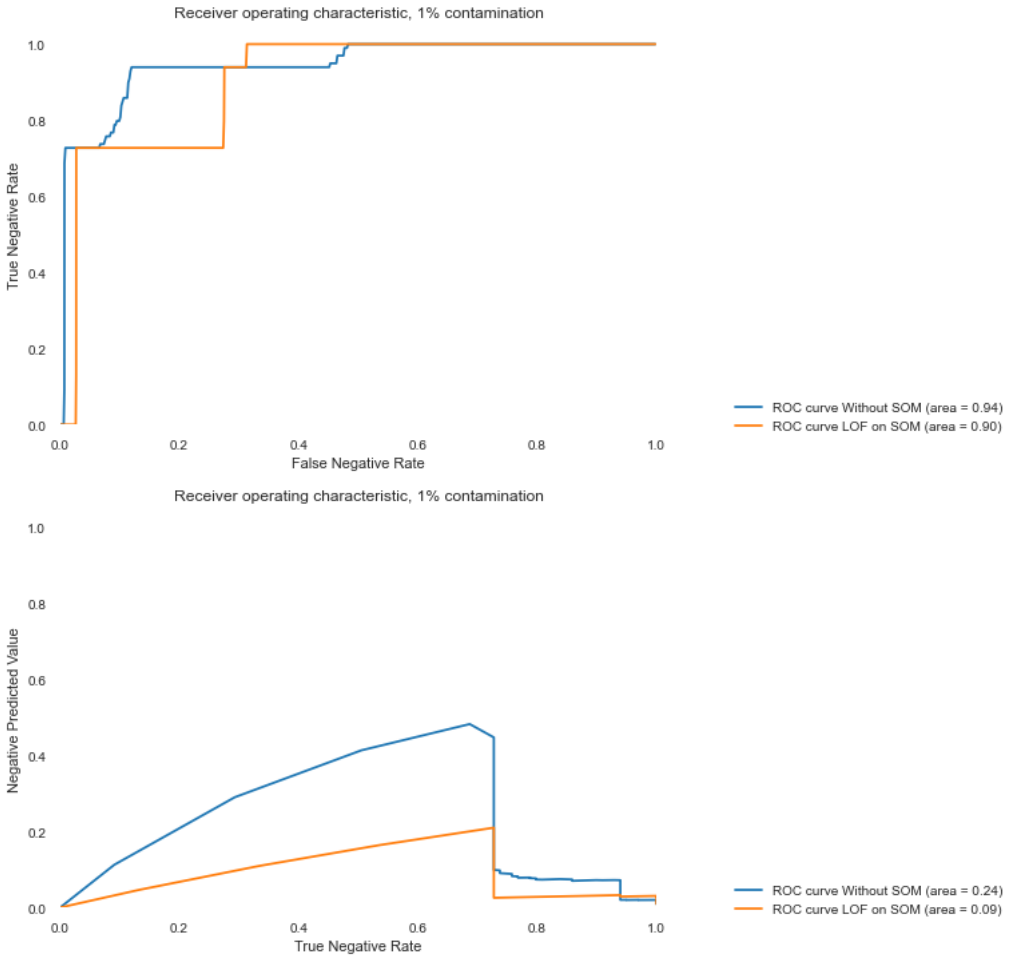

Figure 42: Negative Predictive Value/True Negative Rate curve and True Negative Rate/False Negative Rate curve Local Outlier Factor (LOF) model trained on Self Organising Map (SOM) and without SOM. Dataset used: KDDcup99

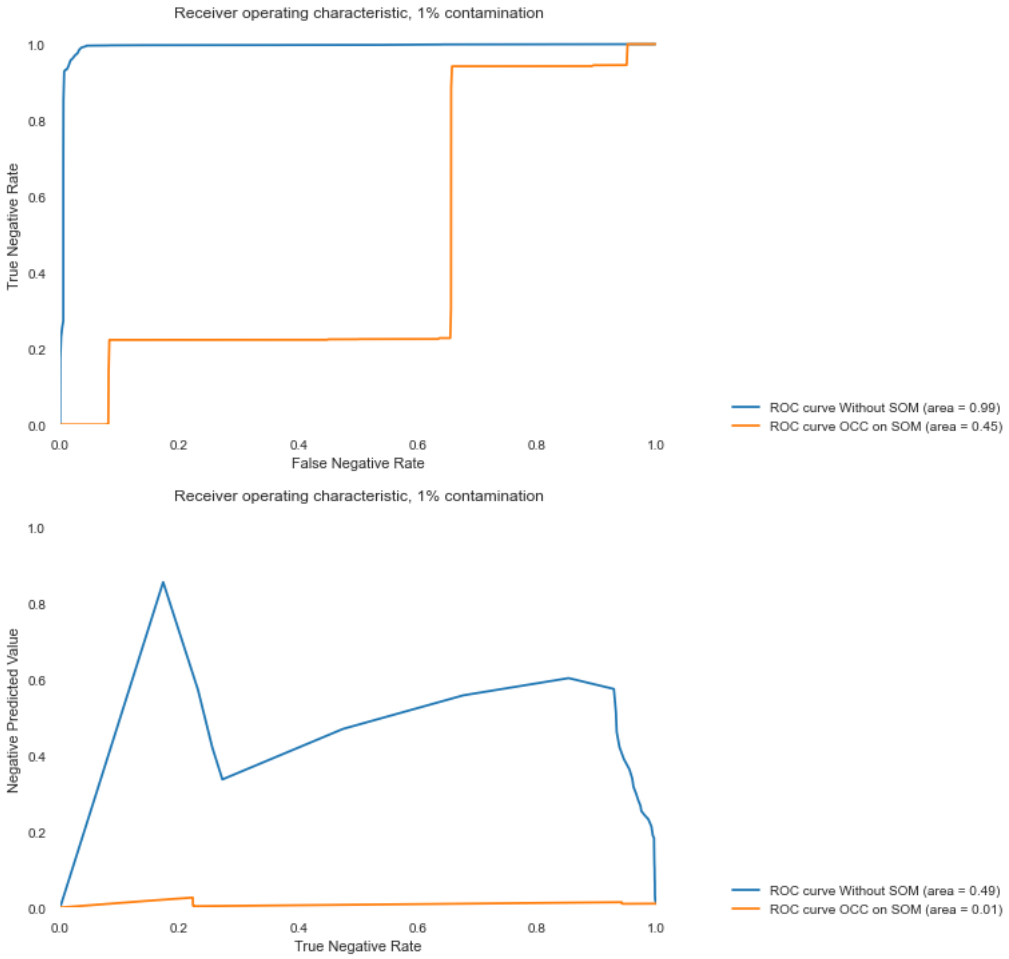

Figure 43: Negative Predictive Value/True Negative Rate curve and True Negative Rate/False Negative Rate curve One Class Classification (OCC) model trained on Self Organising Map (SOM) and without SOM. Dataset used: KDDcup99

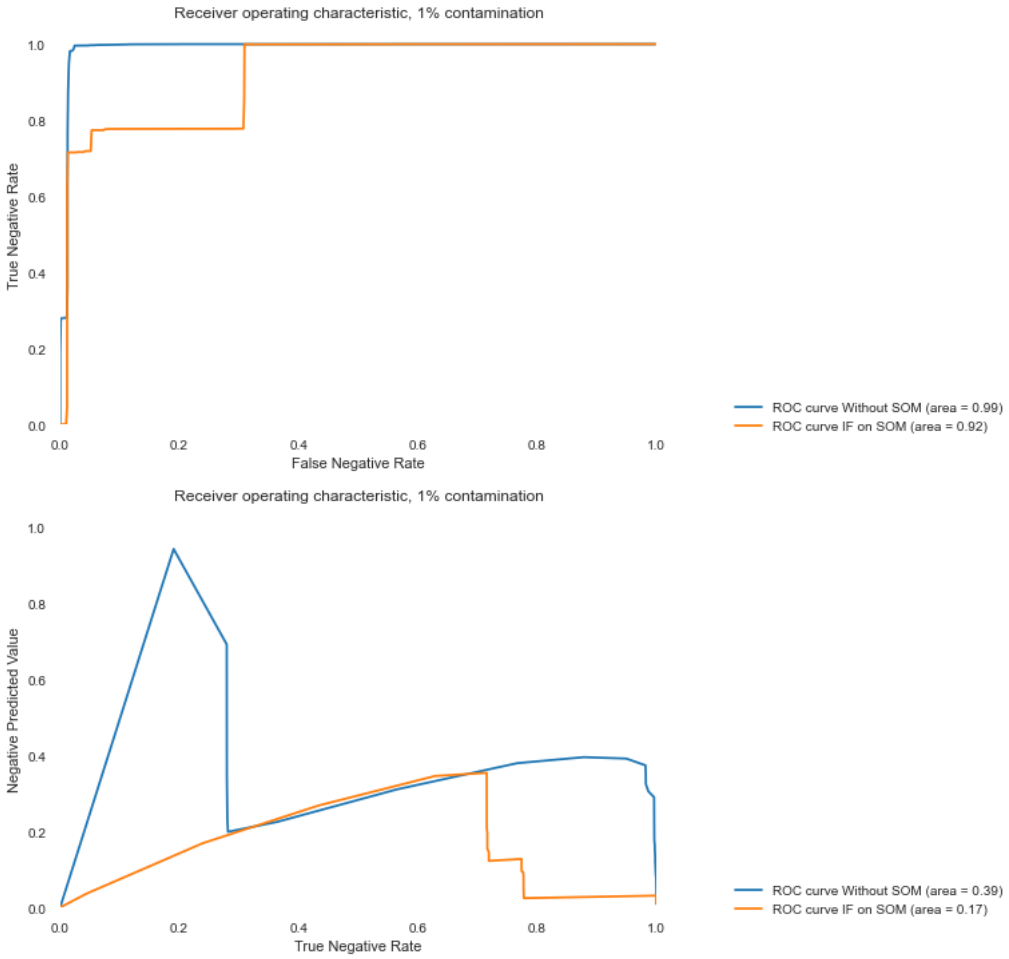

Figure 44: Negative Predictive Value/True Negative Rate curve and True Negative Rate/False Negative Rate curve Isolation Forest (IF) model trained on Self Organising Map (SOM) and without SOM. Dataset used: KDDcup99

E.2 ISCXIDS2012

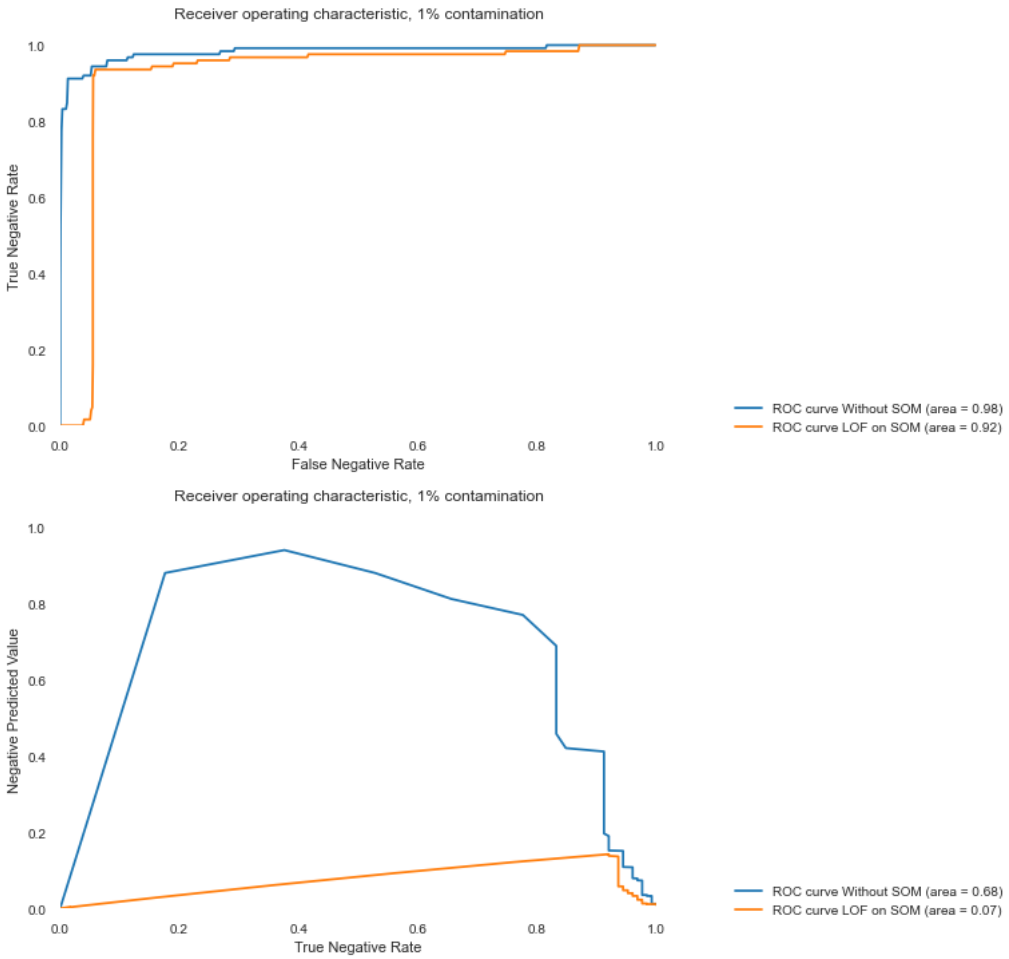

Figure 45: Negative Predictive Value/True Negative Rate curve and True Negative Rate/False Negative Rate curve Local Outlier Factor (LOF) model trained on Self Organising Map (SOM) and without SOM. Dataset used: ISCXIDS2012

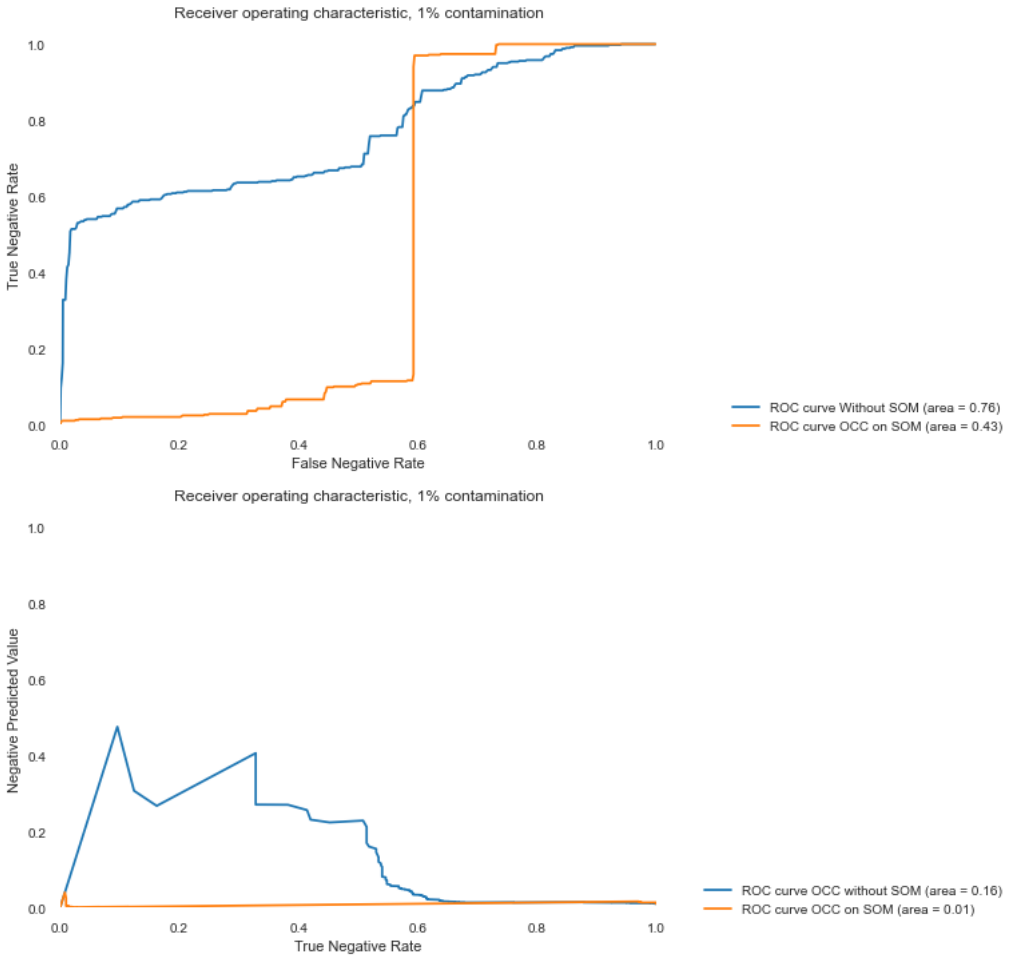

Figure 46: Negative Predictive Value/True Negative Rate curve and True Negative Rate/False Negative Rate curve One Class Classification (OCC) model trained on Self Organising Map (SOM) and without SOM. Dataset used: ISCXIDS2012

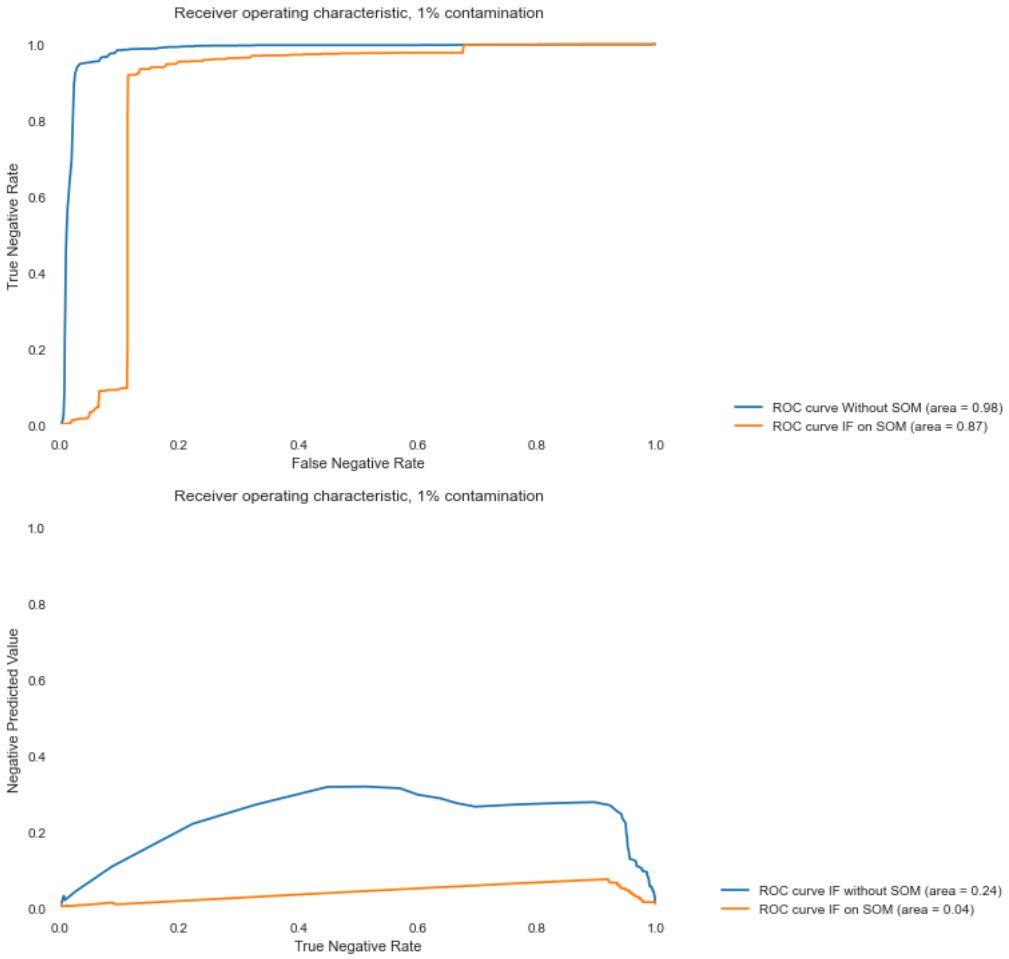

Figure 47: Negative Predictive Value/True Negative Rate curve and True Negative Rate/False Negative Rate curve Local Outlier Factor (LOF) model trained on Self Organising Map (SOM) and without SOM. Dataset used: ISCXIDS2012

F Case study

Table 15: Cross table: similarities in outliers found by the Isolation Forest (IF) and One Class Classification (OCC) models and two different methods of defining a log, taking single logs or summarizing log sessions.

X	IF (single logs)	IF (log sessions)	OCC (single logs)	OCC (log sessions)
IF (single logs)	500	2 log sessions 4 log rows	33 log rows	7 log sessions 25 log rows
IF (log sessions)	2 log sessions 4 log rows	500	19 log sessions 21 log rows	205 log sessions
OCC (single logs)	33 log rows	19 log sessions 21 log rows	500	22 log sessions 29 log rows
OCC (log sessions)	7 log sessions 25 log rows	205 log sessions	22 log sessions 29 log rows	500

Table 16: Cross table: similarities in outliers found by the Isolation Forest (IF) and One Class Classification (OCC) models and three different methods of defining a log, taking single logs, summarizing log sessions without session time and summarizing log sessions with session time.

X	IF (single logs)	IF (log sessions)	OCC (single logs)	OCC (log sessions)	IF (log sessions 2)	OCC (log session 2)
IF (single logs)	500	2 log sessions 4 log rows	33 log rows	7 log sessions 25 log rows	3 log sessions 7 log rows	14 log sessions 59 log rows
IF (log sessions)	2 log sessions 4 log rows	500	19 log sessions 21 log rows	205 log sessions	339 log sessions	253 log sessions
OCC (single logs)	33 log rows	19 log sessions 21 log rows	500	22 log sessions 29 log rows	23 log sessions 24 log rows	25 log sessions 30 log rows
OCC (log sessions)	7 log sessions 25 log rows	205 log sessions	22 log sessions 29 log rows	500	196 log sessions	288 log sessions
IF (log sessions 2)	3 log sessions 7 log rows	339 log sessions	23 log sessions 24 log rows	196 log sessions	500	290 log sessions
OCC (log sessions 2)	14 log sessions 59 log rows	253 log sessions	25 log sessions 30 log rows	288 log sessions	290 log sessions	500