Analysing machine learning models for reducing the workload of security analysts in a SOC

Maartje Veraart

TU Delft

Analysing machine learning models for reducing the workload of security analysts in a SOC

Maartje Veraart

by

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday March 31, 2022 at 14:00.

Student number:	4393791	
Project duration:	April 28, 2021 – March 31, 2022	
Thesis committee:	Dr. S.E. Verwer,	TU Delft, daily supervisor
	Dr. A. Panichella,	TU Delft
	J. Helder,	KPN, supervisor

An electronic version of this thesis is available at `https://repository.tudelft.nl/`.

Preface

This MSc thesis describes the work I have conducted at KPN. This thesis ends my career as a student of the TU Delft. Starting here as a student of the faculty Technology, Policy and Management at the TU Delft I could not have imagined that I would be where I am now. I will finish my student career as a graduate of the Master Computer Science, which makes my journey quite special. Looking back, I am very proud of what I have achieved.

I am both happy and proud to present to you my work on 'Analysing machine learning models for reducing the workload of security analysts in a SOC'. I would like to thank my supervisor Sicco Verwer for his guidance, critical attitude and creative ideas. I would also like to thank Jesse Helder for his guidance, explanations on the workings of the SOC and his ideas on how to improve the SOC. I would also like to thank Annibale Paninchella for being part of my thesis committee and taking the time to read my thesis.

Finally, I would like to thank my family, friends for their endless support and for encouraging me during this challenging time.

Delft, March 23, 2022

Abstract

The rise of alarming cyber breaches and cyber security attacks is causing the world to consider the security of our cyber space. A Security Operations Center (SOC) is a center where the security of a company is monitored to prevent cyber breaches. Security analysts in the SOC examine alerts that come from different devices and analyse what is causing these alerts. The SOC receives a high amount of false positive alerts and duplicates. Therefore security analysts will only react to alerts that seem critical. The problem is that analysts discard alerts that look like false positives but are actually genuine attacks. To tackle this problem of alert fatigue, related work has tried to implement machine learning models to reduce the number of alerts. However, the amount of workload that is reduced is still unsure. We argue that many machine learning models cluster the alerts that are easy for analysts to assess. This thesis compares traditional machine learning techniques with the state-of-the-art neural network DeepCASE and computes the amount of work that is reduced for the analysts. It also compares the machine learning models with a simple heuristic that reduces the duplicates in the dataset. We also enhance DeepCASE to find more sophisticated attacks. We show that using a simple heuristic is as good as using an advanced machine learning algorithm. We also show that using the enhanced version of a state-of-the-art neural network can find more sophisticated attacks.

Contents

1 Introduction

The rise of alarming cyber breaches and cyber security attacks is causing the world to consider the security of our cyber space. As hackers are becoming more skilled every day, the impact of the attacks and cyber breaches today are more alarming than it ever was before. When a cyber incident occurs within a company, it causes a significant impact on an enterprise.
The impact of these incidents is both financially damaging as well as damaging for its reputation as large cyber breaches usually reach the news. For example, the cyber attack in Maastricht in 2019 caused both financial and reputational damage. Intruders in the system of the University of Maastricht caused a severe Distributed Denial of Service (DDoS) attack where the whole network of the university went down. This went viral in the news ánd it paid 220,000 euros to solve this attack [1].
To mitigate the risk of being breached, many enterprises build a Security Operations Center (SOC) to monitor the cyber security of the company and mitigate risk. Several rules are set up on multiple devices in the network and these rules will be triggered when an event meets the criteria of the rule. This trigger is called an alert. An example of such a monitoring system is the Intrusion Detection System (IDS), which monitors the network and detects anomalous behaviour.
One single attack can cause several alerts because alerts can trigger different sensors at the same time [1]. These attacks are responsible for the enormous number of alerts raised in a Security Operation Center. Analysts that investigate these alerts have difficulties splitting the relevant ones from the ones that can be ignored. Thereby, there is always a chance that an alert turns out to be a benign activity, which is called a false positive [2]. A survey by CriticalStart found that false positives are very common in SOCs, where almost half of the respondents reports a rate of 50% false positives or higher [3]. It is very common that when SOCs get overwhelmed with alerts, analysts begin to ignore low to medium priority alerts [3]. If the number of false positives is high, analysts will only react to critical alerts.
Duplicate alerts (alerts from the same attack) and false positive alerts result in so-called alert fatigue. This creates a hazardous situation, where analysts will be ignoring the least significant alerts. Research conducted by the Enterprise Strategy Group found that 34% ignore between 26 and 50% of alerts. 20% ignore between 50 and 75%. And 11 % ignore more than 75% of all alerts [4]. Missing alerts have severe consequences, for example, missing installed malware, not updating security patches and missing intrusions by hackers.
An example of analysts ignoring specific alerts is the Target data breach in 2013. Here 40 million card records were stolen [2]. Target paid 18.5 million dollars for the data breach. The IT team got several alerts from the unfolding attack. However, the team did not react because previous similar alerts appeared to be false positives. The team classified these alerts from the actual attack as false positives and thereby ignoring the attack [5].
To avoid the problem of alert fatigue by the security analysts, the number of false positives and duplicate alerts need to be reduced. The use of machine learning models helps the reduction of these alerts. Ideally, you want to get only one alert for real attacks and no false positives. Unfortunately, removing all duplicates and false positives is a difficult task and exist only in an ideal world. This research will measure the performance of machine learning models and find out if these machine learning methods help reduce alerts.
Most academic works focus on common performance metrics, such as precision, accuracy and the percentage of alerts reduced. Using these metrics give an idea of how many alerts are reduced but do not indicate the

[1] https://portswigger.net/daily-swig/ransomware-attack-maastricht-university-pays-out-220-000-to-cybercrooks
[2] https://www.nbcnews.com/business/business-news/target-settles-2013-hacked-customer-data-breach-18-5-million-n764031

amount of work that is reduced for the analysts.
When a machine learning model has a high percentage of alerts reduced, this indicates that the model can find false positives and duplicates quite easily but the amount of work that is reduced for the security analysts is still unsure. We argue that machine learning models cluster the alerts that are relatively easy for analysts to assess. This study aims to analyse the reduction of the workload of security analysts using machine learning models. Is it truly helpful to use machine learning models in the SOC? We try to answer this question with different machine learning implementations.

1.1. Research questions

Having outlined the problem that has been identified, a research question is defined, that aims to answer the previously defined problem statement. The main research question is:

How can machine learning models be applied in the SOC to reduce the workload of security analysts?

The main research question can be broken down into three sub-questions. First, we identify different metrics that can be used for measuring the performance of security analysts. Then, we use multiple machine learning models to try to predict the false positives and duplicates.

Three sub-questions are now defined to answer different aspects of the problem statement. The sub-questions are:

1. **What are existing metrics for measuring performance of security analysts in a SOC?**
   To answer this question a literature study is done. Different studies are investigated to find the metrics that are useful for measuring the performance of security analysts. The metrics are compared to each other and the pros and cons are weighed. The found metrics can be used in the comparison of the performance of machine learning models.

2. **How do existing machine learning methods perform in a SOC?**
   Different machine learning methods are used. First, classical machine learning techniques are used to identify algorithms that predict false positives and duplicates. The performance is measured using different metrics found by the literature study.

3. **How do state-of-the-art neural networks perform in a SOC compared to existing machine learning methods?**
   The previous question identifies the performance of classical machine learning models. We now use a state-of-the-art neural network to see how this algorithm is performing in the SOC. These results of the machine learning models are compared to a state-of-the-art neural network. This question will be answered by running several experiments and comparing the results using different metrics.

1.2. Research contribution

While other works focus solely on the creation of machine learning models, the focus of this research is the comparison of machine learning models. The most important research contributions are listed below:

- We show that a simple heuristic performs as good as the current state-of-the-art neural network.
- We introduce a proof-of-concept for improving DeepCASE that correlates clusters and find attacks that are difficult to cluster.
- We introduce a new metric for measuring the amount correlation of alerts in the SOC.

1.3. Thesis structure

The remainder of this thesis is organized as follows: Chapter 2 introduces the background which is necessary for the theoretical framework. Chapter 3 describes related work on the topic of measuring the performance of security analysts and machine learning models in the SOC. Chapter 4 describes the method used to create machine learning models. Chapter 5 explores the dataset and attempts to predict the false positives and merged tickets using machine learning models. Chapter 6 implements the method DeepCASE. Chapter 7

compares the heuristic with DeepCASE and creates a new method for DeepCASE. Finally, Chapter 8 discusses its limitations, concludes and describes future work.

2

Background

The background gives a background overview of the information that is needed in the report. First, we explain what a SOC is, what devices are and we explain a challenge in the SOC (Section 2.1). Then, we outline several machine learning models (Section 2.2). We also explain different metrics for comparing machine learning models (Section 2.3). Finally, we explain deep learning (Section 2.4) and the state-of-the-art neural network DeepCASE (Section 2.5).

2.1. The Security Operations Center

A SOC provides a real-time view of the security status of an enterprise [6]. Deployment of a SOC is the best practice for companies to detect and mitigate cyber security incidents [7]. A SOC is a central system to monitor computer and network activity. Several use cases are developed and if such predefined use cases are triggered, SOC analysts investigate these alerts to decide if the alerts are risky. SOC analysts are trained to check if the alerts are false positives or worth investigating. The SOC acts as an intelligent security 'brain' that gathers data from all areas of the enterprise.
Security organisations are commonly divided into a red team and a blue team. The red team tries to attack the organisations to discover weaknesses and vulnerabilities in the network or on devices. In our research, we only look at the blue team. The SOC goes further than only passive alerting because it takes a proactive approach to mitigate security events by hunting for threats and searching for weaknesses in the system.
The framework of this defensive side of the SOC is shown in Figure 2.1. Different devices and sensors in the network are configured to alert unusual or anomalous behaviour. Events are triggered and stored in the Security Information and Event Management (SIEM) where a rule engine handles these events and triggers rules whenever unusual behaviour is taking place. There are many rules set on a device that handles data. When a rule is triggered an alarm is set off. This alert is handled by the SIEM where a more sophisticated alarm is triggered and sent to the security analyst.
An example of a rule is that a user unsuccessfully tries to log in to a system 50 times in a row within a short time. A rule is triggered by the device called the Active Directory (AD) (we will later explain how this works) and the event is sent to the SIEM. The SIEM normalises the event and stores it in its database and it is sent the ticket to the security analyst.
Now that we have given an overview of the main flow of an event in the SOC, in the next subsections we explain every subsystem within the SOC in depth. We first begin at the hear of the SOC, the Security Information and Event Management. Then, we explain several devices that are used in the network. Next, we explain the role of the security analyst. We conclude the subsection with the challenges in the SOC.

2.1.1. Security Information and Event Management

The Security Information and Event Management (SIEM) collects and analyses events coming from all sources within a network, such as firewalls, IDS, AD servers, proxies etc [8]. The SIEM collects, normalises and stores these events.
Each device and sensor in the network is configured to output security events with unusual or anomalous behaviour. These events configured in the network are all represented in a device-specific format. After creating an event by the device, the event is sent to the SIEM in a device-specific format.

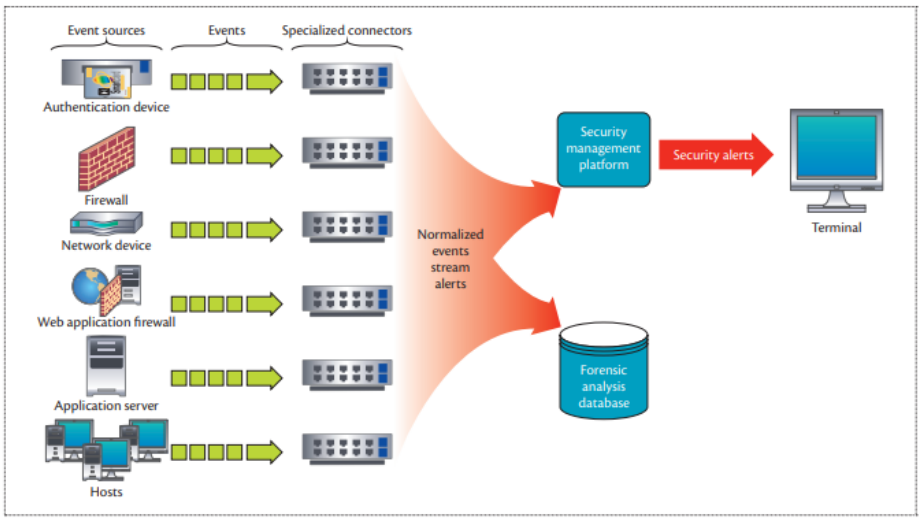

Figure 2.1: SIEM overview, source:

The events coming from the devices are represented in a device-specific format. An advantage of the SIEM is that it can correlate the events from different sources by using common attributes and normalising events to discover meaningful attack patterns. The SIEM can correlate the events from all different devices into a common format.
After the normalisation of the events, they are forwarded to a security management platform and stored in a historical database for future look-ups. The security management platform analyses a window of events with the created rules. The rules in the SIEM allow the correlation of events from different devices. There are two most used approaches for the creation of the rules of the SIEM. One approach is the use of SIEM developers such as Splunk, IBM, LogRhythm [9]. The other approach is the creation of rules by the analysts of the SOC themselves [8].
The difference between events triggered by devices and the alerts triggered by the SIEM is the correlation of the events in the SIEM. For example, devices will trigger an event if one failed login attempt has been made. But the SIEM will trigger an alert if the SIEM recognises multiple failed login attempts by the same user within a specific time window.
After the generation of alerts in the SIEM, the alert is shown to the SOC analysts. The security analysts will analyse the triggered alert. If the alert is a false positive, the analyst will discard the alert and will continue acknowledging alerts. On the other hand, if the alert is suspicious and needs more analysis and mitigation the security analyst will decide what to do next and whom to involve.
A SIEM is an important system within the SOC because it can handle a large number of events coming from different sources, with different formats. When an enterprise starts growing, the number of devices in the network will grow together with the number of events. It also increases the need for one main system to analyse threats. The SIEM can collect a large number of events and store them in a normalised database. This makes it easier to search through the history of events and use that in today's threat analysis.

2.1.2. Devices

Ticketing system After the normalisation of events by the SIEM, the alerts that need to be reviewed are sent to the security analyst. Showing the alerts to the security analyst is mostly done using a ticketing system [10]. The SIEM creates a ticket of the alerts and via the ticketing system, the analyst can examine the ticket. By using a ticketing system the analysts can track the status of the alert and measure different indicators, such as the creation time and the response time.

Network IDS Unauthorised activities within the network pose a threat to any kind of information system, which can be defined as an intrusion. An intrusion is a threat to the confidentiality, integrity or availability of the information within a system. The goal of a network IDS is to identify different kinds of malicious network traffic, such that the confidentiality, integrity and availability of the information system can be protected [11]. There are two types of IDS systems: Signature-based IDS and Anomaly-based IDS. Signature-based IDS use

pattern matching techniques to find a known attack and prevent the attack from happening again. When an intrusion signature matches with the signature of a previous intrusion, the IDS triggers an alarm. The downside of signature rules is that they cannot react to zero-day attacks, because it depends only on previous intrusions. Anomaly-based IDS create a normal model of computer systems using machine learning, statistical-based or knowledge-based methods. When a significant deviation between the observed behaviour and the normal behaviour is detected, an alarm is triggered. It is the strength of Anomaly-based IDS that it can detect zero-days because abnormal user activity does not rely on a knowledge database. When we talk about an IDS in this report we mean a Signature-based Network IDS.

AD The AD is a common repository for information about objects that are in the network [12]. It first started as a centralised domain management. but it became an umbrella title for a broad range of identity-related services, such as users, endpoints, WiFi access points. The AD is used for identity-related services, such as authentication and authorisation of users in a Windows domain network. The best-known service that the AD has is the information about members in the domain is stored and credentials are verified as well as the access rights of the users.
The AD is a crucial part of the network because it plays part in authentication, access management, account management and authorisation. A breach in the AD could lead to unauthorised access of malicious actors. The SOC needs to monitor and alert when suspicious or invalid activities appear in the AD.

2.1.3. Security analyst

To determine where security analysts spend the most time on, we first need to define what security analysts do and what their tasks are. The SOC has different areas of expertise and all departments have their tasks. Chandran et al. list the following tasks as the different roles within the SOC [13]. The tasks of the SOC can be split into multiple Tiers. An overview of the different Tiers is shown in Figure 2.2.

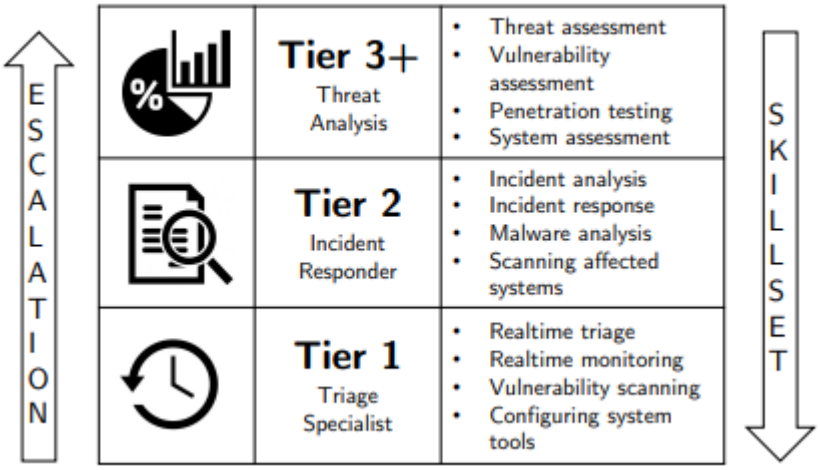

Figure 2.2: Tiers soc analysts [7]

In general, the analyst escalates tickets to a higher Tier and as the Tier level increases, the responsibility of the analyst increases. In general, the higher the Tier, the analysts have a more specific skill-set, as for the lower Tiers, their skill-sets are broad. The duration of resolving a ticket increases in a higher-level Tier because escalated incidents are more complex.
Below we give a rough overview of the different Tiers in the SOC. It is important to note that there may be differences between SOCs of organisations. Or there may be more tiers that perform other tasks, such as tooling, vulnerability assessment and penetration testing.

The first tier is the one where analysts are monitoring activities within the network and on user devices. The Tier 1 analyst receives alerts when a specific rule is triggered in the SIEM. For example, when a user tries to perform SQL injection on a website. These analysts are responsible for determining whether the alerts are real threats, and if so, they will escalate the alert to the Tier 2 analyst. The Tier 2 analyst performs an in-depth

analysis of the incidents escalated by the Tier 1 analyst. The analyst collects data, performs malware analysis and scans the affected systems to measure the threat level of the incident. A Tier 3 analyst has experience with penetration testing tools, data visualisation and scans for vulnerabilities in the systems and network. The Tier 3 analysts join the Tier 2 analysts when major incidents occur.

Next to the Tier analysts, there are other tasks and teams in the SOC that cannot be classified in a Tier. The engineering team is responsible for providing and supporting the SOC with the necessary hardware and software infrastructure. The team makes sure that the SIEM is fed with the data coming from the sensors. They will troubleshoot the performance issues in the infrastructure. Furthermore, an intelligence team will provide information on threats that might affect the organisation, for example, when a new vulnerability in a software system has been discovered or a new IP address that is likely to communicate with Command and Control servers.
The focus of this research is on the tasks of the first Tier analyst of the SOC.

2.1.4. Challenges in the SOC

We will review our example with the unsuccessful login attempts in Section 2.1.1. The SIEM triggers an alert when there are more than 50 times in a row within a short period. This rule identifies a problem in the SOC that is quite difficult for security analysts. This alert can be triggered by two actions:

- A malicious actor performs a brute-force attack on a system (True positive).
- An automatic service attempts to log in a user by using an old password (False positive).

Both of these explanations could be the reason of the alert and it is hard to define rules that only triggers the alerts that are harmful for the organisation. Acknowledging the false positives in the system is quite time-consuming and inefficient and the existing false positives make it difficult for a SOC analyst to distinguish the harmful alerts from the not harmful ones. Ideally, you want to filter out the false positive alerts by using specific rules. The rules that trigger the alerts need to be accurate enough that the false positives are distinguished from the true positives.

We examine a use case for the SOC, to give an in-depth explanation. We want to trigger an alert whenever a brute force attack is taking place but we do not want any invalid attempts by users. The rules that trigger the alerts need to be specific enough that the alerts will rule out the false positives. Let's say that a computer needs 1 second to guess one password. Although this is far too slow for modern computers, it is only an example. The human can fill in a password 1 time per 10 seconds. We have specified the use case and we need a rule that only alerts the brute-force attack. In Table 2.1 different rules are explained.

Table 2.1: Example alerts

	Access attempts	**Time frame**	
Rule 1	Alert when there are 3 access attempts		Too broad
Rule 2	Alert when there are 10 access attempts	Within 1 second	Too narrow
Rule 3	Alert when there are 10 access attempts	Within 1 minute	Good

Rule 1 in Table 2.1 shows an example of a rule that will trigger an alert when there are 10 invalid login attempts. This alert will cause many false positive attempts because humans may fill in an invalid password 3 times in a row. Next to that, the rule did not specify a time frame making the rule far too broad.
Rule 2 shows a rule that can be too narrow. An alert is triggered when there are 10 access attempts within 1 second. In our example, a human, as well as an automated process by an attacker, is not able to fill in a password 10 times in a row within 1 second. Then we can say that rule 2 is far too narrow for our example and the rule needs to be re-visioned to create a better rule for our use case.
The last rule, rule 3, is specific enough. A human is not able to fill in a password that fast because. In our example it will take 10x10 seconds for the human. The brute-force attack is on the other hand triggered by the rule because it takes a computer 10 seconds to complete the brute-force attack.
What we show here is that the creation of a rule is an iterative process. The example we have shown is simple but it can get quite difficult. The analysts that create rules begin at a rule that is broad and results in many false positives. The analyst will look at how the rule can be narrowed down to only the necessary alerts. It is

often seen that rules need to be re-visioned a few times until only the true positives are triggered. This means that rule creation is a time-consuming process, as the rules may not produce too many false positives and may not miss any true positives.
In an ideal world, a security analyst sees no false positives and all harmful events that need to be seen are triggered. However, a perfect world does not exist and there needs to be a right balance of a low number of false positives a low number of missed true positives.

2.2. Machine learning

Machine learning focuses on the process of learning from the surrounding environment and using input data to produce a particular outcome [14]. The algorithms can automatically alter the model, such that they become better at achieving the desired task.

2.2.1. Supervised vs unsupervised

Machine learning can generally be grouped into two categories, supervised and unsupervised learning. In a dataset, machine learning models are represented using a set of features. When a machine learning model is supervised the data is given with known labels. In contrast to supervised learning, unsupervised learning instances can learn to identify patterns themselves and no labels are needed. For the machine learning tasks, we only use supervised learning techniques.

2.2.2. Classification

Classification uses a model that is trained on data. Classification models are trained to predict a class that has a binary outcome, for example, malicious or not malicious. Every observation in the dataset has multiple features that the model can use to identify patterns or distributions. The aim is to label the unseen observations in the dataset.

2.2.3. Naive Bayes classifier

The Naive Bayes classifier is a logical approach that uses the foundation of the Bayes' theorem for classification [15]. The conditional probability is the probability of an event happening when it has a relationship with another event. We will try to find a probability for event A given that B is true. For example, the probability of getting breached given that you have a vulnerability in your computer system.
The following formulas will explain Bayes' theorem.

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{2.1}$$

The same holds for P(B | A):

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} \tag{2.2}$$

From the last formula we can derive that:

$$P(A \cap B) = P(B \cap A)P(A) \tag{2.3}$$

Then Bayes theorem is:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \tag{2.4}$$

Calculating the probability of A given B can be used for training the data for the prediction of a class. Variable A is the class label that needs to be predicted and variable B is the feature that we have information of. The formula will be applied to the training dataset. In general, we will need to find P($A_i|B_i$) for every A_i in A and B_i in B. By calculating the probability of A given B, we can predict the outcome for every A in the test dataset.

2.2.4. Decision tree classifier

The Decision Tree classifier is described in a tree structure [16]. The tree consists of a root node, the internal nodes and the leave nodes that have no child nodes in the tree. When classifying a data sample the tree is used to classify the sample into a class using multiple decision functions in sequence. The algorithm begins

at the top of the tree and for each node, the algorithm picks the best path. The algorithm is done when it reaches the leaf node which represents one class.
The methods for identifying the attributes in the nodes in each level are the Gini impurity and Information gain. There are multiple ways to select the attribute for the node. The most used variant for attribute selection is the Gini index.

Gini Impurity The Gini Impurity is a metric to measure how often a randomly chosen element would be incorrectly predicting a label. The best performing Gini Impurity is chosen for the next node.

$$Gini = 1 - \sum_{i=1}^{N} p_i(1 - p_i) \tag{2.5}$$

where p is the probability of i.

Information Gain The Information Gain is calculated using the concept of information entropy. It can be thought of as the amount of information in a dataset.

$$Entropy = -\sum_{i=1}^{N} p_i log_2 p_i \tag{2.6}$$

For every branch of a feature, the entropy is calculated. If the entropy's are known for the branches, the quality of the split can be calculated by weighing the entropy of each branch by the number of elements. We want the Information Gain that is as high as possible, then it will remove the most information from the dataset.

2.2.5. Random forest classifier

A Random Forest classifier consists of many decision trees on randomly selected data samples [17]. Each decision tree uses a subset of all features of the dataset. After the generation of the decision trees, each tree votes for each predicted result. The prediction with the most votes is the final prediction.

2.3. Metrics

The machine learning classifiers that are explained in the previous paragraphs need to be evaluated using different metrics. We will define multiple metrics that are used while building such learning models. The different metrics explained are accuracy, precision, recall and Receiver Operating Characteristic curve (ROC-curve) and Area Under the Curve (AUC).

2.3.1. Accuracy

When building classifiers the first classification metric that is widely used is accuracy. Accuracy is the metric to define the proportion of correct predictions, both True Positives and True Negatives among the total number of cases examined [18]. The equation of calculating the accuracy is shown below where TP is true positive, TN is true negative, FP is false positive and FN is false negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

2.3.2. Precision

While accuracy is the metric that is widely known in the learning field, it has its drawbacks because it is only useful when the dataset is balanced with an equal number of positives and negatives [19]. We present another set of metrics that can extend the metric accuracy. Precision is the fraction of the predicted true positives that are true positive [20]. Precision is a good metric when the cost of having false positive is high, for example in a spam filter no benign emails must end up as spam in the spam e-mail box. Below is the formula of the Precision metric.

$$Precision = \frac{TP}{TP + FP} \tag{2.8}$$

2.3.3. Recall

Recall calculates how many of the actual true positives are correctly captured as positive by the model [20]. Recall is a good metric when the cost of having false negatives is very high. This is quite important for our

example because leaving threats in the network undetected increases the risk of an adversarial intruding in the network. Below is the formula of the recall metric.

$$Recall = \frac{TP}{TP + FN} \tag{2.9}$$

The metrics explained above hold that the highest possible value is 1 and the lowest possible value is 0.

2.3.4. ROC

The ROC curve plots two parameters: True Positive Rate (TPR) and False Positive Rate (FPR).
TPR is defined as:

$$TPR = \frac{TP}{TP + FN} \tag{2.10}$$

FPR is defined as:

$$FPR = \frac{FP}{FP + TN} \tag{2.11}$$

The ROC curve plots the TPR vs. FPR at different classification thresholds. The curve prints different outputs at the classification thresholds and show them in a graph. An example is shown in Figure 2.3.

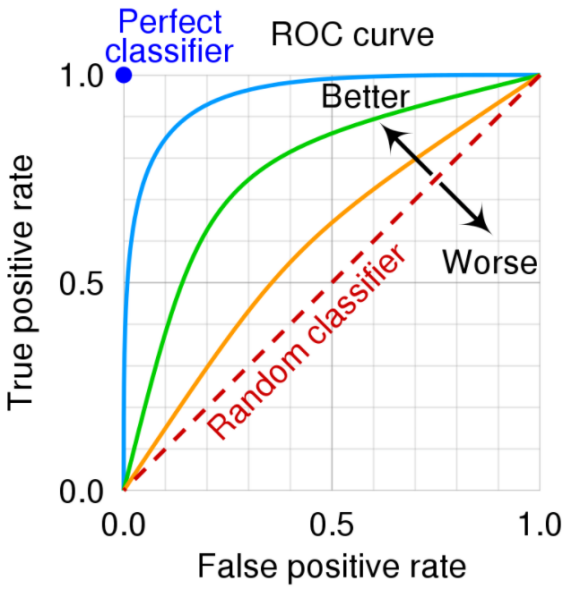

Figure 2.3: Example ROC curve

To evaluate the ROC curve the AUC is measured. The AUC measures the area underneath the ROC curve. A higher AUC means that the curve is better compared to a lower AUC.

2.4. Deep learning

Deep learning is a sub-field of machine learning methods and works with algorithms inspired by the structure of the brain [21]. For this work, activation functions, the recurrent neural network, the encoder-decoder framework the attention mechanism and the loss function is explained. We use these techniques because they are helpful for the algorithms that DeepCASE uses, which is explained in Section 2.5 this section.
The neural network is a collection of simulated neurons of the brain. The neuron can pass electrical signals, to other parts of the brain. This way the brain can pass on bits of information to another part of the body.

2.4.1. Activation functions

A neuron has a threshold and an impulse must exceed that threshold to initiate the impulse. This is how the activation function in artificial neural networks can be explained as well. An activation function defines if

a neuron is activated or not given a set of inputs. An example is the Rectifier Linear Unit (ReLU), which is defined as the positive part of the input. This means that the input can only be 0 or higher. The following equation is used for the ReLU function:

$$\phi(\mathbf{v}) = max(0, x) \tag{2.12}$$

where x is the input to a neuron. When an input is 0 or less than 0, the output of the neuron is 0. When the output is greater than 0, for example, 5, the output of the neuron is 5.
In our research, we use the softmax function in many parts of the research. The softmax function normalises a vector that sums to 1, such that they can be interpreted as probabilities. It is often used as the last activation function to normalise the output of a neural network. The output values of the neural network represent the probabilities as a sum to 1. The following function is used for the softmax function:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K \tag{2.13}$$

It applies the standard exponential function to each element z_i and normalises it by dividing it by the sum of all exponentials.

2.4.2. Recurrent neural network

A recurrent neural network (RNN) can be used to process sequence data because they allow previous outputs to be used as inputs and have hidden states [22]. The network can be used for sequential and temporal problems, such as language translation, natural language processing and speech recognition. In our research, this is very useful for dealing with time-series data and alert title processing.
While traditional deep neural networks, such as feedforward neural networks, assume that inputs and outputs are independent of each other, outputs of recurrent neural networks are dependent on their previous data point. As shown in Figure 2.4 a regular RNN has sequential connections between the hidden states and every input is the output from the previous hidden state.

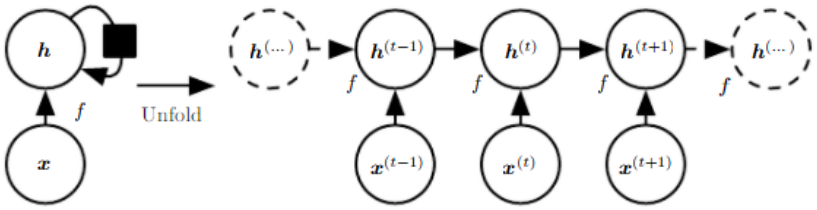

Figure 2.4: Recurrent Neural Network, adapted from [23]

A traditional RNN is a network that consists of a hidden state **h** and an output which operates on a variable-length sequence $\mathbf{x} = (x_1, ..., x_t)$. At each timestep, the RNN uses an activation function **f**, such as SoftMax or ReLU to calculate the output of the hidden state, which is used as input for the next state. The formula for creating the network is shown below:

$$\mathbf{h}_{\langle t \rangle} = f\left(\mathbf{h}_{\langle t-1 \rangle}, x_t\right) \tag{2.14}$$

2.4.3. Encoder-decoder framework

The encoder-decoder framework is useful for encoding a variable-length input sequence into a fixed-length vector representation [24]. The encoder-decoder framework is widely used in machine translation and speech recognition. The encoder encodes the length to a fixed-length vector and the decoder can decode the fixed-length vector back to the variable-length sequence. This is useful for the alerts that come in through the ticketing system and are not all of the same length.
In Figure 2.5 is the framework for the encoder-decoder architecture visualised. At every step, the RNN hidden state h_{t-1} is used from the previous step together with the input x_t to generate the new hidden state h_t with the next sequence.
The encoder summarises the entire input sequence. First, the encoder reads the variable-length input sequence. At each time step, the network will take one token of the input sequence. The hidden state of the

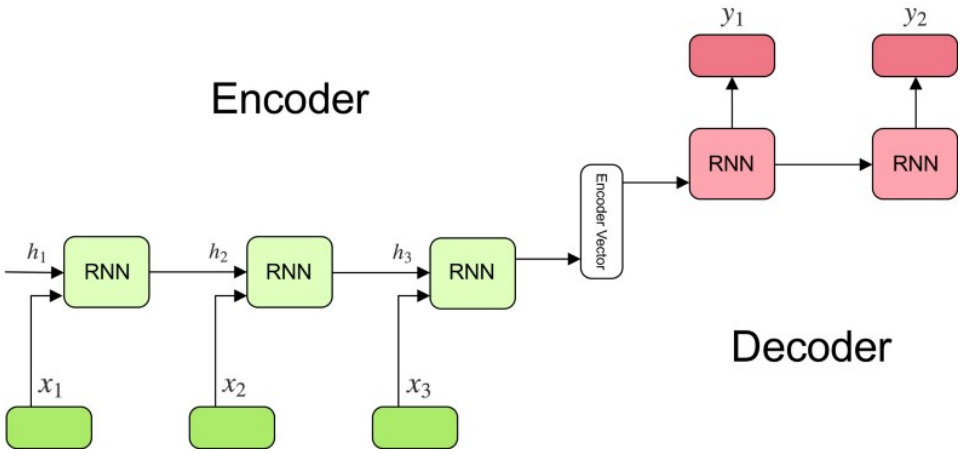

Figure 2.5: encoder-decoder framework, obtained from []

output of the RNN is used as input for the next RNN. The resulting hidden state of the encoder is a summary c of the whole input sequence. The intermediate hidden states are discarded. The output of the encoder encapsulates the input sequence as a whole, which is called the encoder vector.
The final states of the encoder are the initial states of the decoder. The decoder predicts at each time step t the output y_t. The output of the prediction is compared to the actual outcome. By this means, the decoder model learns the true output.

2.4.4. Attention mechanism

A drawback of the encoder-decoder model is its inability to extract strong contextual relations when sentences are quite long. To improve the encoder-decoder model the attention mechanism is introduced by Bahdanau et al [22]. The intuition of the attention mechanism is that solely the crucial parts of the sequence will be paid attention to.
The encoder is similar to the original encoder-decoder model. Instead of computing a single-fixed vector summary of the input, it now develops a context vector for each output time step.
The decoder is slightly different as well because it decides which parts of the context vector will be paid attention to. The decoder still trains to predict the next item, if you give it the context vector and the previously predicted tokens. The decoder calculates an alignment model. The alignment model scores how well each encoded input matches the current output of the decoder. A softmax function is used to normalise the output. These so-called alignment weights indicate the likelihood of each token to be relevant for the current output. These alignment weights will be multiplied by the encoder outputs. If the score of an input element is closer to 1, the effect is highlighted.

2.4.5. Loss

The decoder in the previous paragraphs trains on predicting the correct token. To measure the difference between the probability distribution of the predicted event and the actual event the loss is calculated. Here the Kullback-Leibler divergence is used as loss [25].
The Kullback-Leibler divergence is formally defined as

$$D_{KL}(p\|q) = \sum_{i=1}^{N} p(x_i) \log\left(\frac{p(x_i)}{q(x_i)}\right) \tag{2.15}$$

where $p(x\_i)$ is the true distribution for event x and $q(x\_i)$ is the predicted distribution of event x. The intuition here is that when p(x) is large and q(x) is small, the divergence is large. When the two have equal probabilities, the divergence is closer to zero.

2.5. DeepCASE

The previous section is relevant for the model that we will explain next because it uses the encoder-decoder framework with an attention-based mechanism for contextual analysis and clustering of security events. Van Ede et al. implemented a semi-supervised analysis of security events with the main goal to reduce the num-

ber of alerts that security analysts are receiving from the SIEM [26]. The idea behind this tool is that besides examining the security event itself, it is also necessary to analyse the security events preceding it. The research claims that it can reduce the manual workload of security analysts by 90.53%.

A high-level overview of a SOC with a DeepCASE implementation can be seen in Figure 2.6. First, the data is sent to a network security monitor or IDS. This system will handle the vast amount of packets and send the packets that are flagged as malicious to the DeepCASE tool. Normally, when security-relevant events are detected, a security analyst would receive this event and investigate the event as one single security incident. DeepCASE adds a new element in-between the device and the SOC. This intermediate step can be seen in Figure 2.6. Each event is analysed in the context of preceding events and the goal is to correlate relevant events that are similar. After sending the alert through the algorithm, the tool presents a piece of the cluster to the security analyst that can mitigate the alert.

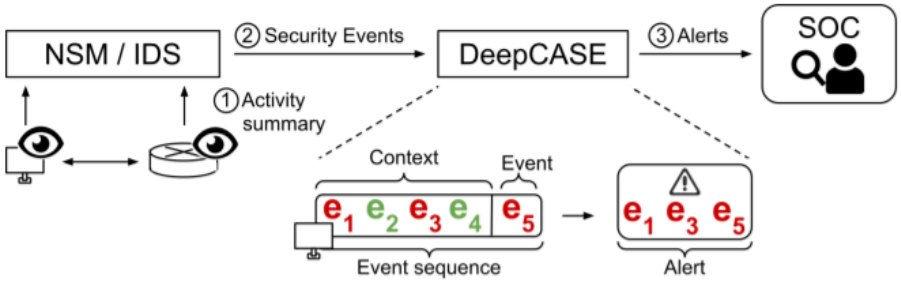

Figure 2.6: SOC with DeepCASE

2.5.1. Model

Using the overview in Figure 2.7 we will go through each step in the model.

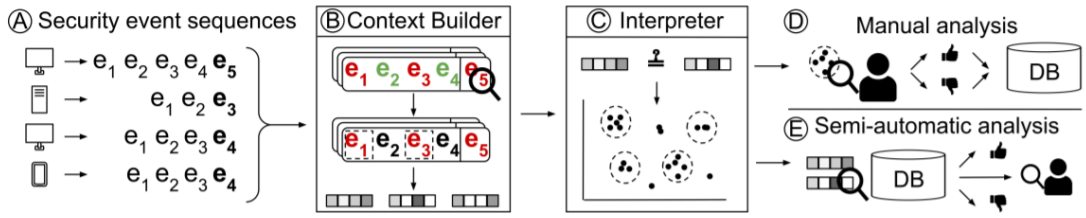

Figure 2.7: Overview of DeepCASE algorithm

First, the algorithm creates sequences of security events by grouping them per device. To reduce search space for relevant contextual events, DeepCASE takes a sliding window for each different device and a maximum time difference between each event and its context. They used a sliding window of 10 and a time difference between the event and its context of t = 84,600 seconds (1 day). The last event is the actual event that is coming in, the events before the event are called the context.
As an example we show a sequence with a sliding window of 3 below:

```
|--------Sequence--------|
|--Context--|     |-Event-|
[1,   2,   3] -->     4
[4,   5,   3] -->     4
[6,   7,   3] -->     4
```

After the creation of the sequences per device, we will use these event sequences in the Context Builder. Figure 2.8 gives an overview of the network architecture of the Context Builder. The aim is to identify relevant

contextual events by building an attention vector. This idea that an attention mechanism is used is borrowed from the natural language processing domain, which is similar to our problem, as alerts are titles that have textual descriptions and are of variable length. Attention is used to focus a neural network on identifying relevant parts of a sequence. DeepCASE uses the approach of an encoder-decoder construction, explained in Section 2.4.3, in combination with an attention mechanism in Section 2.4.4.

The encoder builds a recurrent layer for each contextual event, which will together form an abstract single vector, which is called the context event.
Second, in the attention decoder, the context vector is transformed into an attention vector. This attention vector represents the degree to which each corresponding context event contains information regarding the security event.

In the event decoder, we train a neural network to predict the next event in the sequence, such that it can be used to assign attention to the relevant parts of the contextual event. Because relevant context events give a higher attention score than irrelevant events, a neural network must be able to predict the following event, given the context events weighed by the attention vector. The event decoder takes the embedded context events and weighs them with the attention vector by using element-wise multiplication. The probability distribution over all possible events is now given.

What we want to achieve is to train the Context Builder to learn to calculate the attention vector. We are not interested in predicting the event itself as this is already known but we are interested in calculating the attention vector. After training the prediction of the next event, we can use the same model to calculate the attention vector, using contextual events with the next event.

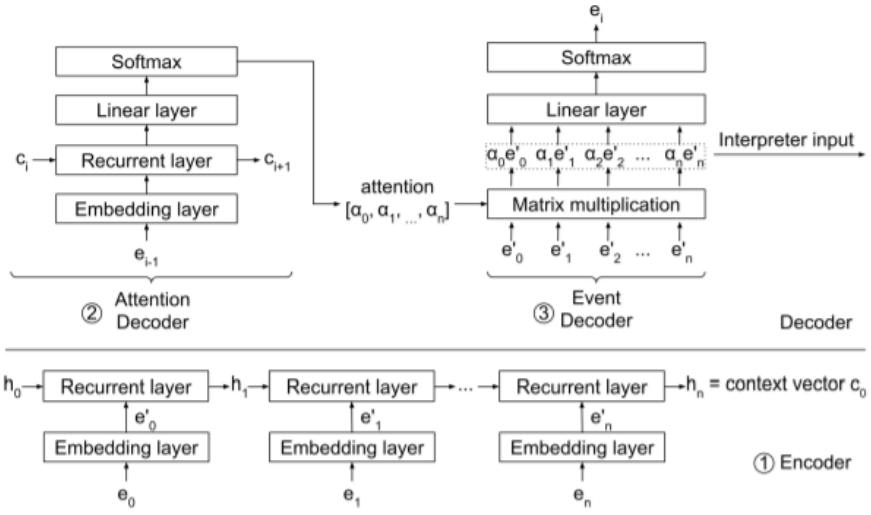

Figure 2.8: Context builder

The output of the predicted event is compared with the desired next event. While fitting the model in the Context Builder the generated output e_i needs to be compared to the desired output e_i through a loss. The loss function that is used here to calculate the loss is the Kullback-Leibler divergence [25].

Third, the interpreter groups the sequences into clusters, based on the total attention for each distinct event in a sequence. The interpreter uses the total attention to compare different sequences. The idea here is that event sequences with similar attention values for the same events are treated the same way by security analysts. This only makes sense if the correct event was predicted.

By combining the attention vector with the corresponding events, the sequences with similar attention vectors can be grouped into a cluster. Two clusters are similar if the distance function is smaller than the threshold ϵ=0.1. The distance function that is used here is the L1-distance given in Equation 2.16.

$$d(x,y) = \|x-y\|_1 = \sum_{i=1}^{m} \left|x_i - y_i\right| \tag{2.16}$$

After the creation of these clusters, these are either evaluated by the security analyst in what they call manual mode or semi-automatic mode. In the manual mode, each cluster is presented to an analyst who decides how the cluster should be treated. These clusters are stored and can be used in the semi-automatic mode by comparing new clusters with previously classified clusters and automatically warning the analyst in case a sequence matches a known malicious cluster.

2.5.2. Toy example

To explain the intuition of the DeepCASE model a toy example is used. Below are sequences shown from the dataset, with the context and the next event. We want to create clusters that show how the attention is calculated and the cluster is created. We outline several examples that will be used in the DeepCASE model. We explain the examples and their results below.

Sequence in place First, we use sequences that end with 3 and 4. This will be in place and the numbers in the rest of the numbers in the sequence are chosen at random.
The attention decoder will generate the attention vector. From the example, we see that the context event with the highest value is 3. We can simply say that this event has more weight. Our attention vector for the cluster looks as follows:

```
|-------Sequence------|
|Context|       |-Event-|
[1, 7, 3] -->       4
[1, 2, 3] -->       4
[8, 0, 3] -->       4
[7, 7, 3] -->       4
[0, 2, 3] -->       4
[4, 8, 3] -->       4
[0, 9, 3] -->       4
|Context|       |-Event-|
|-------Sequence------|

[0.0022, 0.0046, 0.9932]
[0.0027, 0.0029, 0.9944]
[0.0024, 0.0019, 0.9958]
[0.0042, 0.0037, 0.9921]
[0.0027, 0.0013, 0.9960]
[0.0028, 0.0015, 0.9957]
[0.0020, 0.0076, 0.9905]
```

In the event decoder, the attention weights can be used to learn to predict which event comes next. This is done by multiplying the context vector to the attention weights. From this example, we can see that the attention vector is highest for all events that have a 3. Using simple math we can conclude that the first two context vectors are ruled out and the third context has all the attention. The model has learned now that when a 3 is in the sequence, the 4 is the next event.

Sequence not in place DeepCASE also recognises that the number is in the sequence but not at the same place. This can be seen in the next example. We shuffled the number 3 throughout the tables but we made sure that 4 is always in the same place. See for example the following sequence:

```
|-------Sequence------|
|Context|       |-Event-|
[7, 3, 9] -->       4
[3, 6, 2] -->       4
[3, 6, 1] -->       4
```

```
[1, 5, 3] -->     4
[3, 7, 6] -->     4
[7, 6, 3] -->     4
[3, 6, 7] -->     4
[3, 9, 5] -->     4
[8, 1, 3] -->     4
[2, 3, 5] -->     4
[8, 1, 3] -->     4
|Context|      |-Event-|
|-------Sequence------|
```

Then we have the following attention vector:

```
[0.0451, 0.9528, 0.0021]
[0.9951, 0.0021, 0.0028]
[0.9910, 0.0021, 0.0069]
[0.0072, 0.0022, 0.9906]
[0.9638, 0.0340, 0.0022]
[0.0453, 0.0023, 0.9525]
[0.9640, 0.0023, 0.0338]
[0.9954, 0.0023, 0.0022]
[0.0043, 0.0049, 0.9909]
[0.0030, 0.9947, 0.0023]
[0.0043, 0.0049, 0.9909]
```

We can see here that the location of the number 3 is not important for DeepCASE. The attention vector gives the attention to the most relevant part of the sequence. After learning the next event, the model knows that when a 3 is in the sequence, a 4 comes next.

Multiple events in sequence in-place We can also show that when multiple events in one sequence are repeated, the attention vector has difficulty paying attention to all the important events. As can be seen in the next example, the most attention is given to the last event, but the second event also is present in every sequence.

```
-------Sequence------|
|Context|      |-Event-|
[0, 2, 3] -->     4
[7, 2, 3] -->     4
[5, 2, 3] -->     4
[1, 2, 3] -->     4
[1, 2, 3] -->     4
[6, 2, 3] -->     4
[0, 2, 3] -->     4
[7, 2, 3] -->     4
[0, 2, 3] -->     4
[7, 2, 3] -->     4
[5, 2, 3] -->     4
[8, 2, 3] -->     4
[9, 2, 3] -->     4
[8, 2, 3] -->     4
|Context|      |-Event-|
|-------Sequence------|
```

```
[0.0022, 0.0027, 0.9951]
[0.0019, 0.0028, 0.9953]
[0.0053, 0.0025, 0.9922]
[0.0039, 0.0024, 0.9937]
[0.0039, 0.0024, 0.9937]
```

```
[0.0045, 0.0025, 0.9929]
[0.0022, 0.0027, 0.9951]
[0.0019, 0.0028, 0.9953]
[0.0022, 0.0027, 0.9951]
[0.0019, 0.0028, 0.9953]
[0.0053, 0.0025, 0.9922]
[0.0018, 0.0029, 0.9953]
[0.0018, 0.0027, 0.9956]
[0.0018, 0.0029, 0.9953]
```

Multiple events in sequence random This also holds for when these events are placed at random. In the example, we randomly shuffle 2 and 3. We even place the 3 before the 2. Similar to the previous example, the attention vector assigns attention to the 3 in the sequence.

```
|-------Sequence------|
|Context|       |-Event-|
[2, 3, 0] -->       4
[1, 2, 3] -->       4
[3, 1, 2] -->       4
[3, 0, 2] -->       4
[3, 2, 0] -->       4
[2, 3, 8] -->       4
[2, 9, 3] -->       4
[2, 3, 8] -->       4
|Context|       |-Event-|
|-------Sequence------|

[0.0019, 0.9958, 0.0023]
[0.0024, 0.0017, 0.9960]
[0.9961, 0.0020, 0.0019]
[0.9957, 0.0023, 0.0020]
[0.9953, 0.0020, 0.0027]
[0.0018, 0.9966, 0.0017]
[0.0018, 0.0022, 0.9960]
[0.0018, 0.9966, 0.0017]
```

Similar events Another example we can show is when all events are similar. In the next example, we can see that 0 is the event and 1 is the padding event. We would expect that the attention in each attention vector is equally divided because the events are all similar but this is not the case.

```
|-------Sequence------|
|Context|       |-Event-|
[1, 1, 0] -->       0
[1, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
[0, 0, 0] -->       0
|Context|       |-Event-|
|-------Sequence------|
```

```
[0.0043, 0.0046, 0.9911]
[0.0056, 0.4838, 0.5105]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
[0.2771, 0.3463, 0.3766]
```

Removing the event with the most attention It would be interesting to see if the events that have the highest attention is removed, the second time that DeepCASE is run, would give a different result and can find the rest of the sequence. We explore this using 2 and 3 in the context and 4 as the event.

```
|-------Sequence------|
|Context|       |-Event-|
[0, 2, 3] -->       4
[7, 2, 3] -->       4
[5, 2, 3] -->       4
[1, 2, 3] -->       4
[1, 2, 3] -->       4
[6, 2, 3] -->       4
[0, 2, 3] -->       4
[7, 2, 3] -->       4
[0, 2, 3] -->       4
[7, 2, 3] -->       4
[5, 2, 3] -->       4
[8, 2, 3] -->       4
[9, 2, 3] -->       4
[8, 2, 3] -->       4
|Context|       |-Event-|
```

```
|-------Sequence------|
[0.0057, 0.0016, 0.9927]
[0.0038, 0.0022, 0.9940]
[0.0030, 0.0021, 0.9949]
[0.0030, 0.0024, 0.9946]
[0.0030, 0.0024, 0.9946]
[0.0026, 0.0026, 0.9948]
[0.0057, 0.0016, 0.9927]
[0.0038, 0.0022, 0.9940]
[0.0057, 0.0016, 0.9927]
[0.0038, 0.0022, 0.9940]
[0.0030, 0.0021, 0.9949]
[0.0021, 0.0027, 0.9952]
[0.0032, 0.0021, 0.9946]
[0.0021, 0.0027, 0.9952]
```

With the knowledge that 3 has the most attention, we remove all 3's from the dataset to find out if the 2 will be having the most attention. Hereby we remove one item from the sequence length because if we remove the 3, the following context vector will be formed: [4, random, 2]. Thus, removing 3 from the dataset creates an unwanted pattern. This is not supposed to and therefore we only use 2 context events.

```
|-----Sequence-----|
|Context|       |-Event-|
```

```
[0, 2] -->      4
[7, 2] -->      4
[5, 2] -->      4
[1, 2] -->      4
[1, 2] -->      4
[6, 2] -->      4
[0, 2] -->      4
[7, 2] -->      4
[0, 2] -->      4
[3, 2] -->      4
[5, 2] -->      4
[8, 2] -->      4
[9, 2] -->      4
[8, 2] -->      4
|Context|       |-Event-|
|-----Sequence-----|

[0.0036, 0.9964]
[0.0050, 0.9950]
[0.0047, 0.9953]
[0.0035, 0.9965]
[0.0035, 0.9965]
[0.0058, 0.9942]
[0.0036, 0.9964]
[0.0050, 0.9950]
[0.0036, 0.9964]
[0.0037, 0.9963]
[0.0047, 0.9953]
[0.0034, 0.9966]
[0.0036, 0.9964]
[0.0034, 0.9966]
```

We can see here that after removing the 3 in the dataset, the model will know that the 2 has the most attention.

In conclusion, we have seen that the model correctly clusters the events when the events have obvious patterns. The model correctly clusters the events despite the position of the most important event of We have also seen that the model has more difficulty dividing attention to the important context events when there are more patterns in the context. It was shown that when 2 and 3 are in every sequence, the last event would have the most attention. After removing the context event with the most attention, the other event that was in every sequence has now all the attention.

3

Related work

The purpose of this research is to try to give an idea of the performance metrics in a SOC and to compare different machine learning models that reduce security alerts. For this research, there are two relevant fields of work. First, in Section 3.1 related work is explained that define metrics for measuring the performance of security analysts. Most of these works discuss several challenges in the SOC as well. Second, in Section 3.2 we analyse different tools to reduce the number of alerts for security analysts and discuss which tools are relevant for our research. These tools can be divided into two subcategories. The first is the clustering of tickets that come from the same attack type. The second is the removal of false positives.

3.1. Different metrics for measuring the performance of security analysts

In this section, an outline of different works on measuring the performance of security analysts in a SOC is given. We define what metrics exist and what problems arise using these metrics. We also define different challenges in the SOC.

3.1.1. Early work on metrics in cyber security

Early work on the metrics and measures of cyber security cover the difference between measure and metrics [27]. A measure is a concrete, objective attribute, whereas a metric is an abstract, subjective attribute. Metrics can be approximated by an analyst, by collecting and analysing groups of measures. They state that metrics have different purposes in the SOC. It can help organisations verify that security controls comply with policies, processes and procedures, as well as can identify strengths and weaknesses. While this research solely defines problems on the use of measures, they do not define new metrics or measures.
Problems that arise are as follows: Measures are often defined imprecisely, the measurement methods are inconsistently used, the use of qualitative measures can often produce inaccurate or skewed results and the meaning of measures and metrics changes over time.
It is important to note that the use of measures and metrics are used interchangeably throughout related work. For consistency, we use the word metric throughout the report as an analysis of the metrics in a SOC.

3.1.2. On challenges and metrics in the SOC

Kokulu et al. discuss several issues in the SOC [7]. One of the problems the work discusses is the issue of the performance evaluation of the SOC. Current measures appear to be ineffective for measuring the success of SOCs because alerts have often different severity and consequences.
Kokulu et al. interview both SOC analysts and SOC managers. The opinions about implementing automation in the SOC are divided. Managers respond that the speed of response will increase by implementing automation. The analysts worry that the number of false positives and false negatives will increase [7].
About the false positives in the SOC Kokulu et al. state that false positives and false negatives are always a balancing act [7]. Tuning out the false positives can cause some of the true positives to be missed. Some analysts mentioned that the use of automation will help to reduce the repetitiveness of the tasks.
The speed of response and level of automation, evaluation metrics, and tool functionality are the top three controversial issues between analysts and managers. More issues are raised by the experts such as low visibility on devices, insufficient analyst training and high false positive rates.

The metrics that this paper addresses are as follows:

- Number of incidents, number of vulnerabilities discovered
- Number of tickets per analyst
- Time of ticket creation
- Elapsed time of resolution
- Elapsed time of remediation
- Elapsed time of mitigation
- Number of total tickets that are created and resolved
- Mean detection time
- Mean response time
- Mean time of incident closure
- Time taken to react to an incident
- Number of incidents that are not closed
- Number of known attacks prevented

Agyepong et al. give a systematic review of the challenges faced by SOC analysts and the performance metrics [28]. Especially the manual and repetitive processes, sophisticated attacks and workloads are a challenge. Via systematic literature review, they define metrics to measure the performance of the security analyst. The study observes that devising a useful metric for measuring the performance in the SOC is difficult because the number of existing metrics is limited.
Using systematic literature review, they find metrics and create a distinction between quantitative and qualitative metrics. Our focus is on quantitative metrics. In the research, the following quantitative metrics are examined:

- Time to detect an incident
- Average time taken to respond
- Number of alerts analysed
- The number of tickets closed per day
- The number of incidents detected within a specific timeframe
- Time spent on operations by the analyst
- Time spent on each ticket

Another work of Agyepong et al. proposes a framework to measure the performance of security analysts [29]. They notice that security metrics that are specific to SOCs is still an open area of research. The study defines new metrics for measuring the performance including the advantages and the drawbacks. The following defined metrics are:

- Number of incidents raised by an analyst
- Time taken to detect, respond
- Number of tickets closed

The following advantages and drawbacks of the defined metrics are observed: The number of incidents raised is how many incidents one analyst raises. The number of tickets closed is closely related to the number of incidents raised. These metrics are quite easy to implement and analysts get the drive to want to do more. On the other hand, they do not take into account the difficulty, severity or priority of an incident. Using one of these metrics an analyst can take up multiple simple alerts and will get approval from its peers and manager, whereas another analyst will take up one difficult alert and gets a lower score.
Another metric that the work defines is the time taken to detect and respond to an incident. This is useful for tracking if analysts are taking too long to respond to an incident and to test the analyst's vigilance. It is difficult to put a timeline on how quick analysts should identify an incident, as almost all incidents are different. It can also lead to analysts spending less time on defining the root cause of the incident.

When it comes to assessing the performance of SOCs, the industry generally rely on statistical data generated from algorithms [30]. The problem is that it fails to take into account the human factor. Sundaramurthy et al. sell this as a problem because it fails to take into consideration human factors [30]. The authors argue that the different functions and tasks expected from analysts could be one of the reasons why researchers find it difficult to devise an objective metric.

The success of the SOC depends on having the right tools and effective and efficient analysts. One of the issues in the SOC is the high burnout rates of security analysts [30]. This results in analysts making poor judgements and alert fatigue. They state that metrics can affect the perception that the management has about the usefulness of the SOC. The metrics they propose are:

- Number of events received per day
- Number of events that were bulk processed
- Number of events left unprocessed per day
- Average time taken to analyse an event

This paper raises concerns about these metrics as well. For example, analysts were concerned that the average amount of time taken to analyse an alert did not account for the time spent on meetings, operational tasks and projects. This metric can also not be projected to the actual effort spent in resolving an incident. When an analyst spends a few hours on a false positive, this will not be calculated as an incident. By this means, analysts waste time analysing false positives. This is a reason for analysts to discard alerts that look as if they are false positives.
Automation serves two main purposes in creating a human capital SOC. First, analysts can engage in challenging and interesting tasks by automating repetitive tasks. Also, the platform could provide analysts to express their creativity. They can, for example, build more tools to improve the platform.

The goal of Chandran et al. is to take an anthropological approach to address certain problems, such as team structure, training methodologies and metrics used for measuring SOC efficiency [13]. They state that much work towards the performance of security analysts is done through interviews in the SOC. However, they are useful, it is difficult to quantify their performance. There is also an issue of trust that limits the amount of information an analyst shares, which can create a quite shallow interview. Chandran et al. state that analysts are spending a few hours on an alert that appears to be a false positive. Current metrics are not able to capture the effort of the security analysts.

3.1.3. On creating tools for the SOC

Rosso et al. create a tool that enables security analysts to evaluate the performance of deployed SOCs [31]. The tool is called SAIBERSOC and relies on a framework that generates and injects synthetic attacks into a SOC. The experiment is done with 124 students that have the role of a security analyst. The goal of the project is to evaluate if the proposed method is effective in identifying differences in the performance of a SOC. They evaluate the results considering the following variables: the total number of reports submitted, the number of submitted reports dealing with one attack, the correctness of the submitted reports compared against the ground truth.
The results show that the methodology is effective in identifying the performance of the SOC. Although they identify differences in performance, the SOC is in a simulated environment. The students that joined the

experiment were already familiar with the specific attacks that were presented to them. In contrast to this experiment, attacks in a real SOC are not known.

Oesch et al. try to identify concerns in machine learning models [32]. They create attack campaign templates that contain several attacks. Both experts and students handle the simulated attack incidents. They conclude that there exists a lack of sufficient explanation of machine learning concepts. The usability of machine learning models must be improved to help analysts, who are not experts in machine learning, to know to use machine learning models.

Jaferian et al. propose multiple heuristics for evaluating the usability of IT security management tools [33]. The heuristics are visibility of activity status, flexible representation of information, rule and constraints, planning and dividing work between users, capturing, sharing and discovery of knowledge and verification of knowledge The project validates the effectiveness of the heuristics by using them in an evaluation process. The results show that no single heuristic can uncover all problems and there must be more heuristics applied while evaluating the IT security management tools. These metrics however are qualitative and will not be used in our research.

3.2. Alert reduction techniques

There are two methods for reducing the number of alerts in a SOC. One is the clustering of events that are from the same attack. The second is the filtering of false positives. We will discuss the related work for both methods.

3.2.1. Clustering techniques

Haas et al. propose a novel clustering algorithm that uses graphs to find and cluster multiple stages of an attack in the alerts [1]. Alerts are clustered based on the similarity between alert attributes, such as source IP and destination IP. Then, the attack patterns among the hosts within each cluster are identified. For example, one attacker is attacking one victim, or multiple attackers are attacking one victim. Third, when a multi-step attack is present, clusters of single attack steps are compared based on the communication patterns within clusters.

Husak et al. [34] present a framework for processing and clustering alerts. The components in the framework use sequential rule mining and complex event processing. In contrast to other works, the focus of the framework is on real-time stream-based processing of the data.

Hassan et al. present an algorithm to combat threat alert fatigue using contextual and historical information of generated threat alerts [35]. It generates a causal dependency graph of an alert event and assigns an anomaly score, based on the frequency with which related events have happened before. They state that their system decreases the volume of false positives by 84%. Following, they estimate that this saves time of more than 90 hours. This result is based on the estimate of the average time analysts analyse one ticket. One ticket can be more time-consuming than the other ticket. Therefore we do not know how much time the algorithm reduces.

Van Ede et al. design a system that leverages the context around events to determine which events require further inspection [26]. They make use of a recurrent neural network that highlights important events in the history of events. Then, the clusters are presented to the security analyst, who determines whether the combination of events poses a threat to the infrastructure. The model learns this decision and applies it to similar event sequences found in future events. They show that this approach automatically filters 86.72% of the events and reduces the workload of security analysts by 90.53%.

3.2.2. Filtering false positives

Meng et al. study the use of supervised machine learning methods to filter false positives in IDS. they construct labelled datasets and use this to train K-Nearest Neighbour, Support Vector Machines, Decision Trees and Naive Bayes [36]. They manage to filter out more than 80% of false positives.

Jallad et al. [37] study the difference between traditional machine learning models and deep learning models. They obtain a 10% better false positive rate than a Support Vector Machine.

Aminanto et al. [38] state that the enormous number of threat alerts leads to a severe problem known as threat-alert fatigue. To combat this problem, they propose a real-time screening scheme based on a fully-unsupervised isolation forest. Their research contribution is to use an automated screening system that is fully unsupervised. The system takes temporal information into account and predicts the current day's data trained on the previous days. Their results show that they reduced threat-alert alert logs by 87.41% without missing any true alerts. They do not examine the time reduction of the isolation forest in comparison with the normal set-up.

3.3. Conclusion

In summary, we have seen some works that outline metrics that measure the performance of security analysts. We have defined several problems that exist using these metrics and the challenges faced by security analysts in a SOC. We have defined tools that use machine learning to cluster alerts and reduce false positives.

3.3.1. Metrics

We see throughout the works a need for evaluation of the performance metrics in the SOC. The performance of security analysts is often done through interviews in the SOC. It is hard to devise useful metrics and appear to be ineffective for the success of the SOC. The performance metrics are usually hard to quantify, imprecise and inconsistently used. It is stated that are limited metrics available. The metrics that are stated in the works are combined and summarised in the following table together with their challenges:

Table 3.1: metrics

Metric	**Author**	**Challenges**
Time to detect an incident	[28], [29], [7]	Difficult to define the starting point of the incident.
Time taken to respond	[28], [29], [7]	Can lead to analysts spending less time to understand the root cause.
Time spent on each ticket	[28], [30], [7], [13]	Some tickets take longer than others, do not take into account the number of false positives.
Number of incidents raised by an analyst	[29]	Do not take into account the number of false positives, do not take into account the complexity of the ticket.
Number of alerts analysed	[28]	Do not take into account the complexity of the ticket.
Number of tickets closed per day	[28], [29], [7]	Do not take into account the complexity of the ticket.
Number of alerts received per day	[30], [7]	Does not measure the performance of analysts.
Number of events left unprocessed per day	[30], [7]	Some events need answers from others.
Number of tickets created per analyst	[7] , [29]	Do not take into account the number of false positives.
Number of known attacks prevented	[7]	Difficult measure.

The metric 'time spent on each ticket' is used as a measure for examining the reduction of the workload of the security analysts.

3.3.2. Clustering and filtering false positives

In Section 3.2 we have seen several tools that can be used for clustering events and filtering false positives. We have seen that several tools use the metric 'percentage of alerts reduced'. Although this can say a lot about the performance of the tool itself, it does not say anything about the workload reduction for the analyst. The papers do not have information on the amount of work an individual ticket costs for the analyst.
Thereby, automation in the SOC is often stated as a challenge overall and should be researched on how this could be applied in the SOC. Automation can remove repetitive processes and create room for analysts to express their creativity. The problem is that people do not trust automation and worry about the increasing number of false positives and false negatives. Another problem of machine learning models in a SOC that arises is that the security analysts are often not experts in machine learning. The usability of the models

needs to be improved to help analysts use and understand machine learning models.

3.3.3. Research gap

As best to our knowledge there is no research conducted on the actual workload reduction of alert reduction tools in a SOC. Previous works assume an average workload per ticket. They define the workload that is reduced by using this assumption. But there is no knowledge on what tickets costs more time than other tickets. Therefore, it is unclear what the workload reduction is of the alert reduction tools.
We will use the tool DeepCASE and use it on our dataset, obtained from the SOC, containing information on ticket processing time. We can use the metric 'time spent on each ticket' and compare the models with this metric. Because we have availability of the time on each ticket, we can analyse how much time the models reduces.

4

Methodology

This chapter describes the methodology for answering if machine learning models can help security analysts with the reduction of the workload. We describe here how we have collected the data (Section 4.1), prepared the data (Section 4.2) and what models we will use for our analysis (Section 4.3 and 4.4) .

4.1. Data collection

We collected data from the ticketing system of the SOC of KPN. As explained in the background in Chapter 2 the SIEM correlates events and sends them to the SOC as one ticket. When the SIEM has triggered an alert, the ticketing system automatically creates a ticket. These tickets are stored and security analysts examine the tickets. After the analysis of the analyst, the security analyst closes the ticket. The time between the creation of the ticket and the closing of the ticket is calculated and stored as the ticket duration. We can use this feature in our model to analyse the performance of different machine learning models.
Next to the ticket duration, the security analysts can check a box when an event is a false positive. In the following table, the features are outlined.

Table 4.1: Features and their description

Features	**Description of feature**
Title	Alert description
Queue	Which Inbox the ticket is in, senior investigation or SOC inbox
State	merged or closed
Agent	Agent that has committed the ticket
Priority	Low/Medium/High
Service	Name of the system that the alert is about, such as IDS or Windows ATP
Customer	Customer
False Positive	Checked/Unchecked, analyst checks when the alert is a false positive
Year	What year is the ticket created
Quarter	Which quarter of the year is the ticket created
Month	Which month is the ticket created
Week	Which week in the year is the ticket created
Day	Which day in the week is the ticket created
24x7	When is the alert generated
Response time (in minutes)	Time between ticket creation and first response
Ticket duration (in minutes)	Time between ticket creation and closing and merging ticket
Merged	Ticket number to where the merged ticket is merged to

4.1.1. Merged tickets

The feature *State* is a Boolean and can either be *merged* or *closed*. The closed tickets signify that the ticket has been seen for the first time. The *merged* ticket is a ticket that is from an attack that has been seen before.

For example, when someone commits a vulnerability scan and the attacker tries to find vulnerabilities on multiple devices in the network, different devices can trigger multiple alerts. Security analysts merge these tickets of the same alert to the first closed ticket. In Table 4.2 is an example shown of alerts that belong together. In our research, we call these tickets 'duplicate tickets'.
The feature *Merged* keeps track of where a merged ticket has been merged to. Every ticket has an index and if a ticket has been merged, this feature will state what the index is of the parent ticket. If the state of the ticket is *closed*.

Table 4.2: Example closed ticket with merged tickets

index	Title	State	Merged
1	[Sightline] Host Detection alert #1256406 incoming to IP	closed	1
2	[Sightline] Host Detection alert #1256407 incoming to IP	merged	1
3	[Sightline] Host Detection alert #1256408 incoming to IP	merged	1

4.1.2. Ticket duration time

It is important to note that the duration time of the ticket is not the time that an analyst spends on the ticket. It is the time that a ticket has not yet been closed or merged. It could be that the analyst is waiting for answers from other departments or analysts. The dataset gives little information on the actual time that an analyst spends on one ticket.

4.2. Data preparation

The dataset needs to be altered in a way that it can be used with the machine learning models. In the next section, we will explain what needs to be modified before using DeepCASE on the dataset. Table 4.3 shows two examples before all modifications.

Table 4.3: Before modifications

Title	Created	Queue	State	Priority	Service	Customer
[Sightline] Host Detection alert #1381424 incoming to <IP> done	2021-08-29 11:45:10	SOC Inbox	merged	4 high	Anti-DDoS	Customer KPN
[Sightline] Host Detection alert #1381425 incoming to <IP> done	2021-08-29 11:45:08	SOC Inbox	merged	4 high	Anti-DDoS	Customer KPN

Table 4.4: Continued

False Positive	Year	Quarter	Month	Week	Day	24x7	Response time (in minutes)	Ticket duration (in minutes)
Unchecked	2021	3	8	34	Sunday	Day	3	0
Unchecked	2021	3	8	34	Sunday	Day	3	0

4.2.1. Modifications

Now we modify the dataset. First, we remove random numbers in the titles to reduce the differences between titles that belong together. For example, one title is called [Sightline] Host Detection alert #1381424 incoming to <IP> done. This detection alert is a DDoS alert, which indicates a DDoS attack on a specific IP address. The number #1219233 is the specific ID for this alert. The alert is similar to other events that have the same IP address, but with a different ID, such as the next alert: "[Sightline] Host Detection alert #1381425 incoming to <IP> done."
The *Service* column is changed such that every column has its customer. The dataset has more customers, however, the dataset is not making a distinction between these customers. For example, in the column *Service* Customer 1 is called *Siem Extern* and this is also the case for Customer 2 where *Service* is called *Siem Extern*. Because these customers are completely different events, there needs to be a distinction between these customers.

Table 4.5: After modifications

Title	Created	Queue	State	Priority	Service	Customer
[Sightline] Host Detection alert incoming to <IP> done	2021-08-29 11:45:10	SOC Inbox	merged	4 high	Customer 1	Customer KPN
[Sightline] Host Detection alert incoming to <IP> done	2021-08-29 11:45:08	SOC Inbox	merged	4 high	Customer 1	Customer KPN

Table 4.6: Continued

False Positive	Year	Quarter	Month	Week	Dag	24x7	Response time (in minutes)	Ticket duration (in minutes)
Unchecked	2021	3	8	34	Sunday	Day	3	0
Unchecked	2021	3	8	34	Sunday	Day	3	0

There are some null values in the dataset as the workflow of the analysts changed over time. At the beginning of the dataset, we see that the analysts only filled in features such as *Service*, *Customer* and *False Positive* when an alert is not a merged ticket. This creates null values when the ticket is indeed merged. Assuming that the merged tickets have the same specifications as their parent ticket, we modify the merged ticket such that it matches its parent ticket. Later in the dataset, we see that the analysts change the way they operate and they fill in the information in the merged tickets as well.
In the ticket system, the analyst needs to check themselves if the ticket is a false positive. One setback is that the analyst does not check the box when the false positive is a merged ticket. We solve this by using the feature 'Merged' to collect for all merged tickets the parent tickets. When a parent ticket is a false positive, then the merged ticket needs to be a false positive as well.

4.2.2. Encoding

The dataset contains text and categorical values, for example, the alert title and the day in the week. An alert title can be for example: 'SERVER-WEBAPP F5 BIG-IP Traffic Management User Interface remote code execution attempt [1:54484] - Prio: High'. We have to convert these labels from categorical and text values to numerical values. The conversion is done by using Label-Encoding, One-Hot-Encoding and CountVectorizer. If the data is ordinal, which means that there is a hierarchy in the data, we use Label-Encoding. Label-Encoding is a well-known encoding technique because it is the easiest way of encoding a value. The way Label-Encoding is done is by converting every text value to a number in a sequence [39]. Take for example the feature *Priority*. Priority high is 3, Priority medium is 2 and Priority low is 1. The features we convert using this method are *State*, *Priority*, *False Positive*.

If there is no hierarchy in the data, the data is nominal. Using one-hot encoding is a good alternative for nominal data [39]. Each categorical value is converted into a value, assigning a 1 or 0 to each value in the row. Rows containing the value of the column are assigned a 1, otherwise a 0. The downside of this method is that you add more columns to the dataset. Especially when multiple features have a wide variety of values. For the features *Agent*, *Day*, *Queue*, *Service*, *24x7*, *Customer* the one-hot encoding is used.

The alert titles need a different approach. The approach here is to use a vectorizer to deal with natural language data. We used the library scikit-learn to do this [40].
We use an easy algorithm that is the Count Vectorizer. The vectorizer creates a matrix that counts all words and presents every unique word in a column. The row represents the word count in the sentence. When we have two alerts the matrix looks as follows:

Table 4.7: Count vectorizer

	Apache	Struts	remote	code	execution	attempt	class	access
Apache Struts remote code execution attempt	1	1	1	1	1	1	0	0
Apache Struts class access attempt	1	1	0	0	0	1	1	1

4.3. Selected machine learning models

The techniques we use are machine learning models. First, the classical classifiers are used to predict the labels. Then a state of the is neural network DeepCASE is used to try to cluster the events based on previous events.

4.3.1. Machine learning

The models are implemented in Python, using the algorithms from the library scikit-learn [40]. After training the model, we try to predict two features, the *False Positive* and *State* features. To predict the features the classifiers Naive Bayes Classifier, Decision Tree Classifier and Random Forest Classifiers are used. These models are widely used in anomaly detection.

4.3.2. Cross-validation

Cross-validation is a statistical method for evaluating and comparing machine learning models. The idea is that one sample is used for training a model and the other one is used for testing [41]. Multiple cross-validation methods are available. The method we use is the rolling cross-validation technique [42]. The data is split into several folds. Every fold is split into a training and test dataset. Starting with a small subset of the dataset. After the prediction of the test set, this dataset is included as part of the training data in the next fold. Figure 4.1 shows an example of a 4-fold cross-validation model. The last round includes the whole dataset.

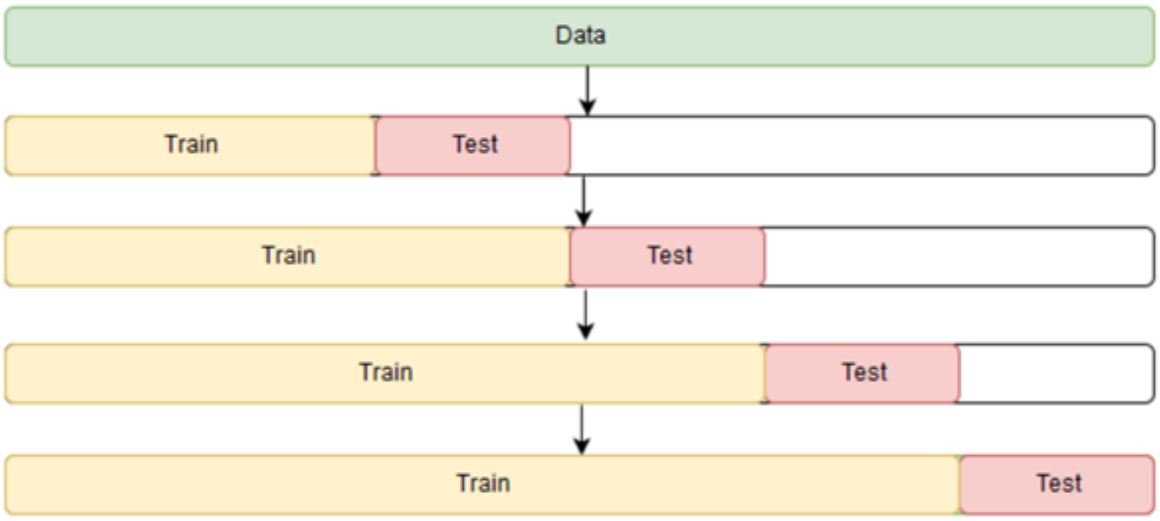

Figure 4.1: Time Series cross-validation. Obtained from medium.com [1]

4.3.3. Sliding windows

Sliding windows is a window of a defined size that slides across the dataset. The window contains the number of instances that the window can hold. In the example, the sliding window is 8 and at every time step, the window is moved one step to the right.

4.3.4. Hyper-parameters of Models

We use the default parameters that the classifying methods were given. We tune the parameters of the cross-validation and the sliding windows. The parameter tuning is explained in Appendix B.
In our models, the number of folds that are used is 10 because this was the best of our experiments shown in Appendix B. The size of the sliding windows is 1.

4.3.5. Evaluation Metrics

Different evaluation metrics are applied to evaluate the results. These are explained in Chapter 2.3. The metrics most used are accuracy, recall and precision, ROC curve and AUC.

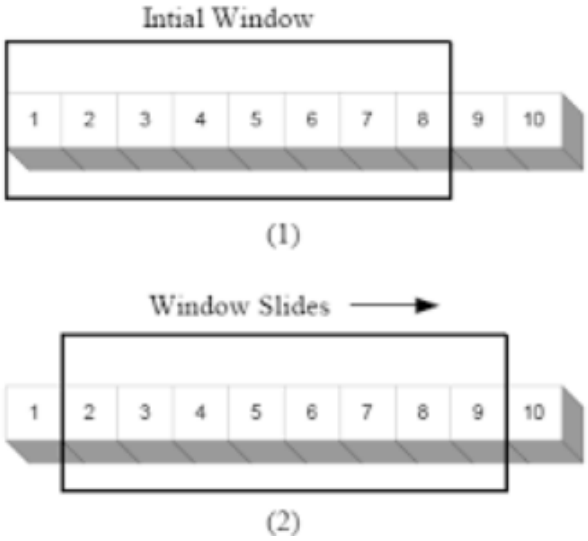

Figure 4.2: Sliding window

4.4. DeepCASE

The dataset that is applied to the machine learning models is now used in Chapter 6. For the DeepCASE model, the same dataset is used. The goal is to create clusters that are based on sequences of events on a device. DeepCASE is written in Python. The MIT licensed code is open-source and can be downloaded by using the Pip installation manager or from source, from Github: https://github.com/Thijsvanede/DeepCASE. The overview of the model is displayed in Figure 2.7.

4.4.1. Features in DeepCASE

DeepCASE uses the features *Event*, *Machine*, *Timestamp*. The feature *Event* is where the information about the alert is stored. In Section 4.1 is explained that *Title* is the name of the event is information on the event is stored. This is our *Event* feature. The *Machine* feature is the device for which the event is created. The *Service* column is the column where the relation is most similar to this explanation. The *Timestamp* is evaluated by the *Created* column.

4.5. New clustering techniques datasets

In Chapter 7, a new dataset is used. We will still use DeepCASE but there will be multiple stages and multiple datasets used in this chapter. The first dataset that is used on DeepCASE is directly coming from IDS devices in the network. These IDS devices have rules installed and whenever a rule is triggered by an event, the event is forwarded to the Security Information and Event Management where the events will be filtered and formatted. This leads to a reduced number of security events sent to the security analyst. This dataset is applied to DeepCASE to see how many events can be filtered by DeepCASE. This result can be compared to subsequent datasets, where duplicate events already have been filtered.

After the use of the IDS dataset with many events and many duplicates, we compare this dataset with the IDS tickets from the ticketing system. Previous chapters use the complete ticketing system dataset. It is important to note that in this Chapter only IDS tickets from the tickets are used because the IDS tickets of the ticketing system dataset have information on source IP and destination IP.
This is useful for combining the duplicates, which we will be explaining next. Out of all tickets, the number of IDS events in this dataset is 7015. This IDS data will be filtered out of the ticket system dataset.
Below we can find an overview of all the datasets.

Table 4.8: Overview datasets

Dataset	**In Chapter**	**Number of alerts**
Ticketing system dataset	5, 6	20,744
IDS dataset	7	9,372,579
IDS - ticketing system dataset	7	7015

5

Data analysis

In this chapter, the tickets are obtained from the KPN SOC. First, different features of the tickets are visualised and analysed (Section 5.1). Then we give an explanation of the engineering of multiple features for the prediction (Section 5.2). Afterwards, we try to predict different classifiers try to predict when a ticket is a false positive and when a ticket is a merged ticket (Section 5.3).

5.1. Data visualisation

In the following section, we use different visualisation techniques to deepen our understanding of the ticket dataset. We hypothesise that the time that it takes for a false positive to process takes less time than the true positives, as these must be mitigated and sent to different departments within the SOC.
A merged ticket is a ticket that is the same as a ticket that has been alerted before and is merged with the parent ticket. We argue that these duplicate tickets take less time than the original tickets.

In the heatmap in Appendix A, we see that there are some correlations between features. For example, it can be seen that *Title* has a positive correlation with *Service* because an IDS has similar alerts, as well as Windows ATP that have similar looking Titles. In the next subsections, we built on this and try to find more correlations between different features.

5.1.1. False positives

In Table 5.1 the number of false positives is shown. In total there are 19.595 tickets after cleaning the data. We can see here that the number of false positives is less than the number of true positives. There are a few explanations for this result. A lot of the false positives are already filtered by the SIEM that has a static white list that compares the incoming alerts with the white list and marks these as false positives and removes them from the list of alerts.
Moreover, the analyst marks the false positive alert and only when the analyst acknowledges and checks the false positive as a false positive, the alert will be 'Checked' at the False positive feature. Due to human error and inconsistent use of the checkbox by different analysts, it could be that the number of false positives should be more than that is stated here.

While the number of these false positives are quite low, we can see that the average duration of analysing these tickets are less than the true positives, but still more than 60 minutes. If one ticket will take on average 67 minutes and we have 1840 false positives per year. Then the sum of all these tickets is 122,785 minutes over one year. This means that the analysts spend 122,785 minutes on all false positives combined. This is an enormous number that needs to be reduced.

In the box plot of Figure 5.1 the number of false positives is divided into bins of 30 minutes. This means that the first bar consists of all false positives that have a ticket duration of 30 minutes or less. The second bar has ticket durations of 30 minutes to 60 minutes and so forth. The most false positives cost less than 30 minutes, however, there are a lot of false positive alerts that require more time. More tickets take more time than 30 minutes than there are tickets that take less time than 30 minutes.

Table 5.1: Number of false positives and average ticket duration

	Number of tickets	Average ticket duration (in minutes)	Total ticket duration (in minutes)
Unchecked	17787	286.251	4,774,960.0
Checked	1840	67.096	122,785.0
Total	19685	264.590	4,897,745.0

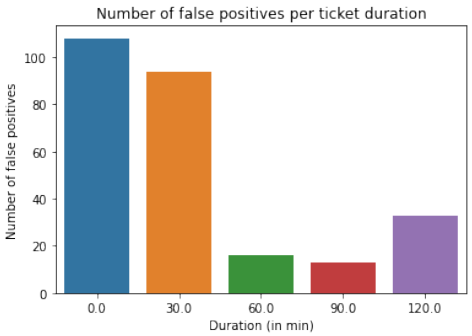

Figure 5.1: Number false positives per bin duration

5.1.2. Merged tickets

The number of closed and merged tickets and the average ticket duration are shown in Table 5.2. We can see that the number of merged tickets is more than the number of closed tickets. The average ticket duration of all these merged tickets is less than 1 minute.
In Table 5.2 the number of closed and merged tickets are shown, with the average ticket duration of that type of ticket. There are far more merged tickets than closed tickets. The number of tickets that are merged is 15026 and the number of actual unique tickets are 4659. Although the average ticket duration is quite low, it is important to examine how these merged tickets can be reduced because the analyst needs to merge them manually.

Table 5.2: Number of merged and closed tickets and average ticket duration

	Number of tickets	Average ticket duration (in minutes)
Closed	4659	1181.615257
Merged	15026	0.559

5.1.3. Ticket duration

In Figure 5.2 we see that there are several outliers in the dataset. However, we cannot see any clear correlations between the time of year and the ticket duration.
The agent who handles the ticket is dependent on the ticket duration. This can be seen in Figure 5.3. Some agents analyse the tickets that require more time, this could mean that these agents are Tier 2 agents, as the other agents handle the easier cases. The agent is also dependent on the length of the ticket duration.
In Figure 5.4 can be seen that during the evening the tickets are taking more time to be analysed. This could be explained by the fact that during the evening and night normal operations are closed and will continue in the morning. security analysts will need to wait for a response to their questions until the morning. However, during the night the tickets do not take much time, because there is not much happening at night.
In Figure 5.5 we see that the average ticket duration is more on Saturday.

5.2. Feature engineering

We can see from the data visualisation that the features that are created by the security analysts, such as *Ticket Duration (in minutes)*, *State* and *False Positive* are quite important for predicting labels. In Figure 5.1 can be seen that a large amount of false positive tickets take no longer than 30 minutes.
After the process of handling the ticket, new metrics are created that can be used by managers to analyse the performance of the SOC and the security analysts. These metrics are the features *Ticket Duration (in*

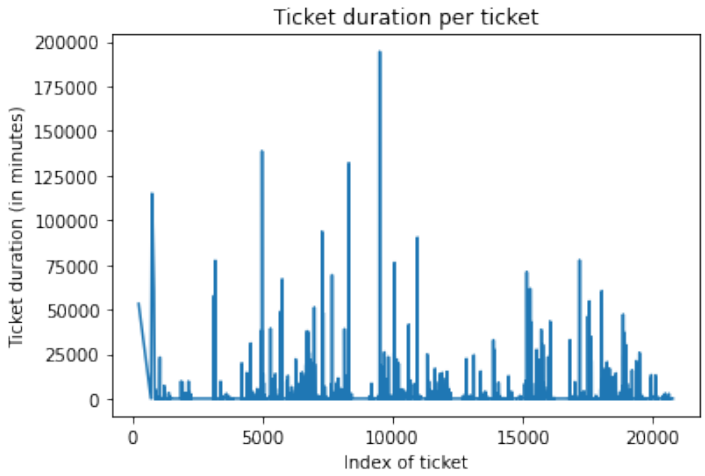

Figure 5.2: Duration per ticket

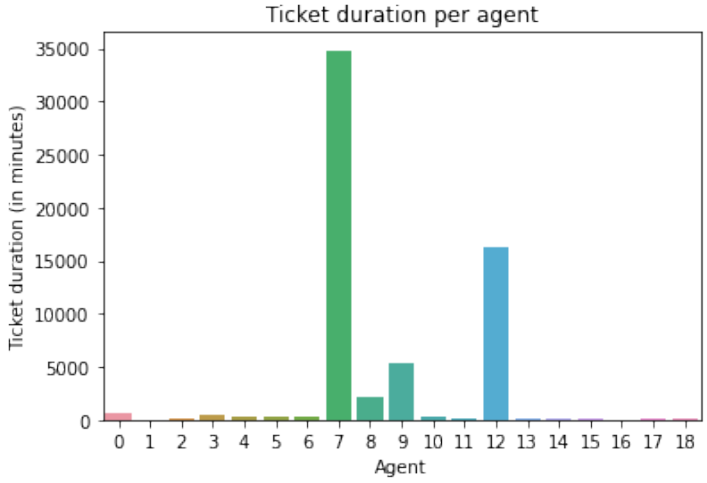

Figure 5.3: Agent owner duration

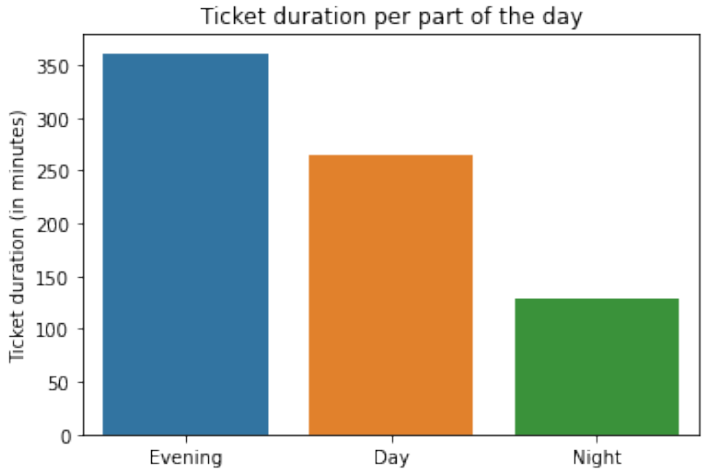

Figure 5.4: Duration per part of the day

minutes), *State* and *False Positive*. To predict a False Positive, we cannot use these features, because, at the

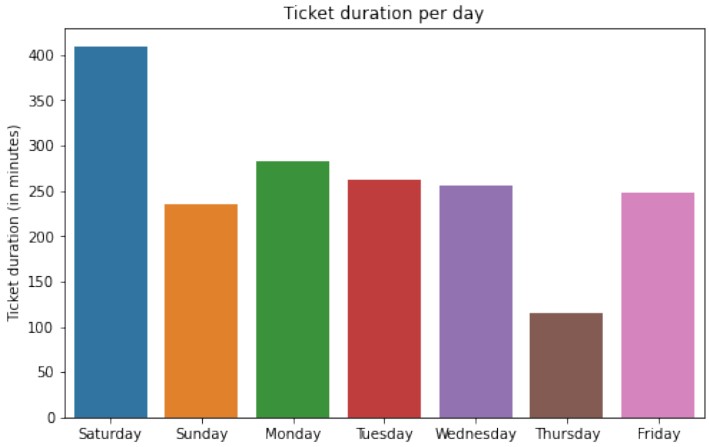

Figure 5.5: Average ticket duration per day

moment of analysis by the security analyst, we do not know the values of these features.
Instead of using the features directly, we can find correlations between other features and create new features using the heatmap in Appendix A. In Table 5.3 an overview is given of the created features. In the heatmap can be seen that *State* and *False Positive* are correlating with the label and with *Title*, we first build multiple features based on *State* and *False Positive*. Especially *State* and *False Positive* are interesting and these correlate with the *Title* feature, which may be used.
We often use the information of the previous ticket with the same title. That means that the latest alert that has the same alert title will be used for the current alert. For all designed features a table is used to explain the creation of these features.

Table 5.3: Feature engineering

Features	**Description of feature**
Cumul_Title	Cumulative count of the same titles
Duration_Mean	Average duration of alerts that have the same title
Prev_State_Title	Previous alert that had the same title is merged will have a 1
Prev_DLT_Title	Duration of the previous ticket with the same title
Title_State	Cumulative count of merged tickets with the same title
Title_FP	Cumulative count false positive is checked with the same title
Cumul_Alerts_per_day	Cumulative count of the number of alerts on the same day

In sequence-based data, the training must be trained on historic data. The counting and averaging will be done based on previous data, not future data. Therefore, we use the cumulative count, sum and mean to create our features.

5.2.1. Cumul_Title

First is the number of alerts that have the same title. An example of how this works is shown in Table 5.4.

Table 5.4: Cumul_Title

Title	**Cumul_Title**
Title1	0
Title1	1
Title1	2
Title1	3
Title2	0
Title2	1

5.2.2. Duration_Mean

We can create a measure for the duration of alerts that have the same title. We take the average of all previous alerts that have the same title as the current alert. An example of how this works is shown in Table 5.5.

Table 5.5: Duration_mean

Title	Ticket Duration (in minutes)	Duration_Mean
Title	5	0
Title	10	5
Title	8	10
Title	16	15.3

5.2.3. Prev_State_Title

When the alert of the previous ticket is also merged, then the value of this feature will be 1. If the previous ticket with the same title is closed, it will be 0. We do this because seen in the dataset is that there are often more merged tickets of the same title than just one. An example of how this works is shown in Table 5.6.

Table 5.6: Prev_State_Title

Title	State	Prev_State_Title
Title	merged	0
Title	merged	1
Title	closed	1
Title	merged	0

5.2.4. Prev_DLT_Title

We also use the duration of the previous ticket with the same title. An example of how this works is shown in Table 5.7

Table 5.7: Prev_DLT_Title

Title	Ticket Duration (in minutes)	Prev_DLT_Title
Title	5	0
Title	10	5
Title	8	10
Title	16	8

5.2.5. Title_State

We now count the number of merged tickets with the same title. An example of how this works is shown in Table 5.8.

Table 5.8: Title_State

Title	Merged	Title_State
Title	merged	0
Title	merged	1
Title	closed	1
Title	merged	2
Title	merged	0
Title	merged	1

5.2.6. Title_FP

The same is done for the number of false positives, where the number of false positives is counted by the same title. An example of how this works is shown in Table 5.9.

Table 5.9: Title_FP

Title	False Positive	Title_FP
Title	FP	0
Title	FP	1
Title	TP	1
Title	FP	2

5.2.7. Cumul_Alerts_per_day

We now use the *Day* feature to create a feature that counts the number of alerts per day. An example of how this works is shown in Table 5.10.

Table 5.10: Cumul_Alerts_per_day

Day	Cumul_Alerts_per_day
1	0
1	1
2	0
2	1

An overview is seen in the following table:

5.3. Classification

With the use of different classifiers explained in Section 2.2, we predict the labels *False Positive* and *State*. The label *False Positive* is a 1 when the alert is a false positive. The label *Merged* is a 1, when the alert is a merged ticket. We want the label *False Positive* to have good precision because we do not want any true positives being defined as false positives. We want for the label *State*, to have good precision because we do not want any closed tickets being predicted as merged.

5.3.1. Prediction of false positive

The label for *False positive* is 0 when the ticket turns out to be a true positive, thus a real incident, and 1 when the ticket is a false positive.

Table 5.11: False Positive classifiers comparison

	NaiveBayes	Decision Tree	Random Forest
Accuracy	0.56	0.78	0.90
Precision	0.14	0.26	0.21
Recall	0.56	0.36	0.04
AUC	0.56	0.61	0.84

Here we see that the models are not very good at predicting Precision and Recall.

5.3.2. Prediction of state

The label for *State* is 0 when the incident is closed and 1 when the incident is merged. Now we try to predict with the same methods when a ticket is merged or closed.
We see that the recall metric is 95%, which means that it can correctly predict if an event is merged with another event. We can explain this by the knowledge that events that are merged are similar to their parent events. Therefore we see that the recall is good. The precision is not good, this means that the model incorrectly classified closed tickets as merged tickets.

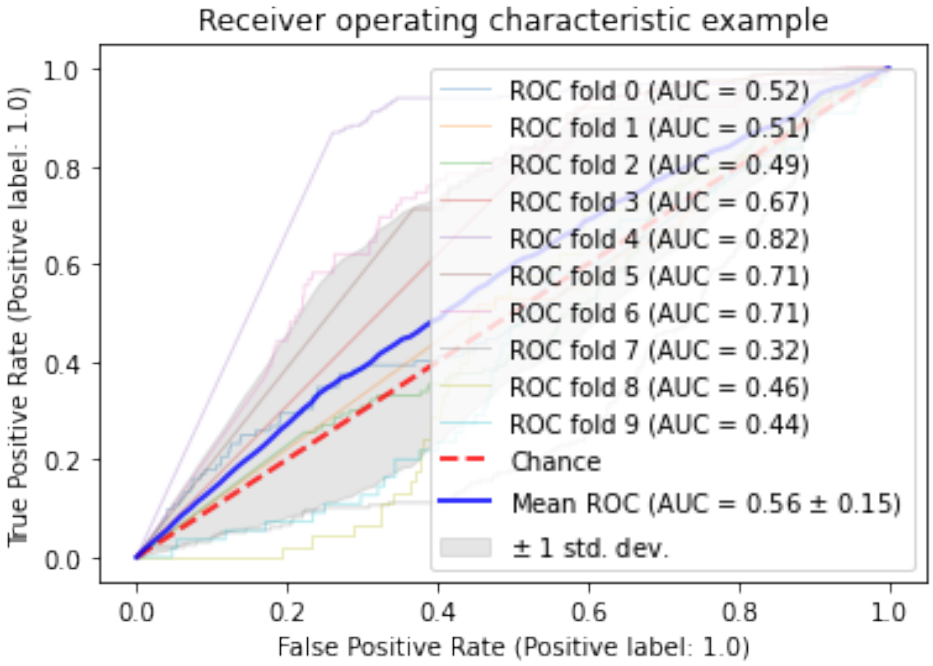

Figure 5.6: ROC False Positive Naive Bayes

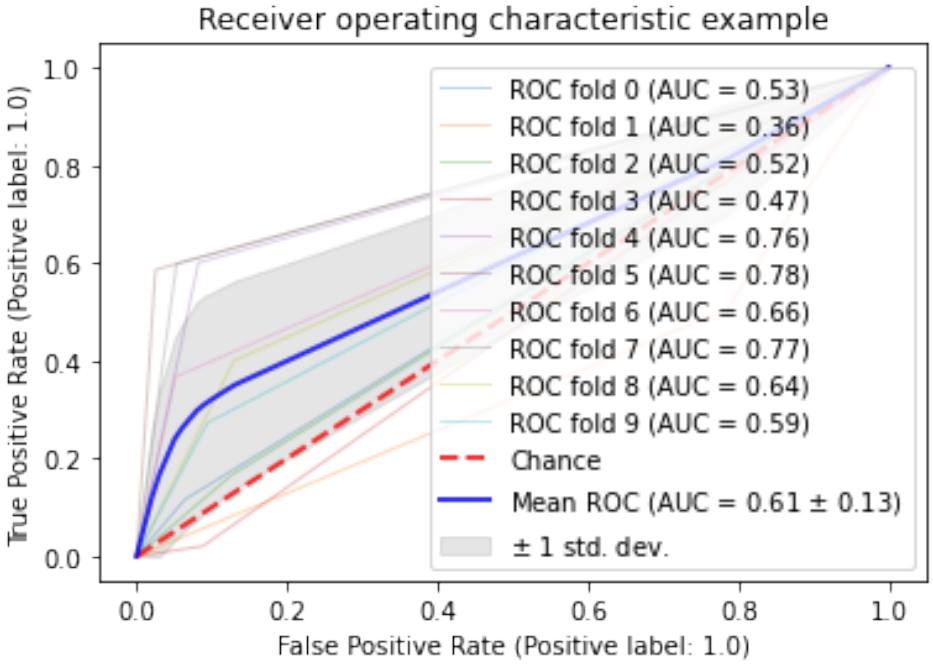

Figure 5.7: ROC False Positive Decision Tree

5.4. Conclusion

In this chapter, we have explored the dataset with real SOC tickets and used classifiers on predicting two separate labels. The first label is the number of false positives in Section 5.3.1. The second label is the merged tickets in Section 5.3.2. The machine learning models did not show any significant results for predicting the false positives in a SOC and for predicting the merged tickets. The most remarkable result to emerge from the data analysis is that the number of merged tickets is very high. Section 5.1.2 showed that are more merged tickets than new alerts are coming from new suspicious events. The findings of this analysis would suggest

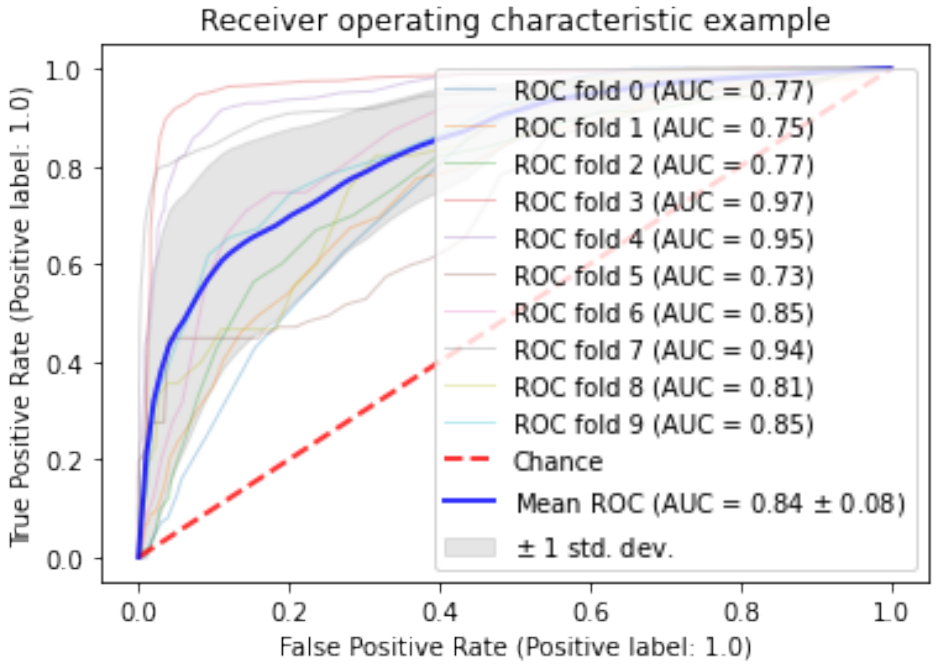

Figure 5.8: ROC False Positive Random Forest

Table 5.12: Merged or closed classifiers comparison

	NaiveBayes	**Decision Tree**	**Random Forest**
Accuracy	0.50	0.69	0.78
Precision	0.82	0.82	0.80
Recall	0.42	0.75	0.95
AUC	0.60	0.62	0.81

that the workload of the security analysts is not increased by the merged tickets. However, we have also discovered that the ticket duration of a merged ticket is quite low. The mean ticket duration of a merged ticket was less than one minute. Constantly merging tickets to other tickets is no challenge for security analysts and this will cause repetition. We can address alert fatigue by removing this number of merged tickets.
This forms the opportunity to research how we can reduce these tickets and cluster them such that the analyst only receives the original ticket. Although it is useful to remove these merged tickets, we have seen that removing these tickets does not remove the time that security analysts spend on a ticket.
We have tried to predict the false positives and the merged tickets using machine learning classifiers. The results of predicting using machine learning on both labels were poor. Predicting the false positives turned out to be more difficult than predicting the merged tickets. In Table 5.11 the results of predicting false positive alerts are shown. This shows that the Random Forest Classifier is best at predicting.
Table 5.12 shows that the recall is fairly good for the Random Forest Classifier. This means that the merged tickets are almost all predicted as merged tickets. On the other hand, we see that the precision is lower and this means that closed tickets are incorrectly predicted as merged tickets.
Using these classifiers increase the chance of incorrectly classifying a true positive or a closed ticket. For the algorithm to be working in a SOC this must not happen. We must be aware of such alerts. Implementing this model in the SOC is not advised because the SOC will miss such alerts. Using this information we move on to a state-of-the-art neural network and compare the performance of the classifiers with the neural network.

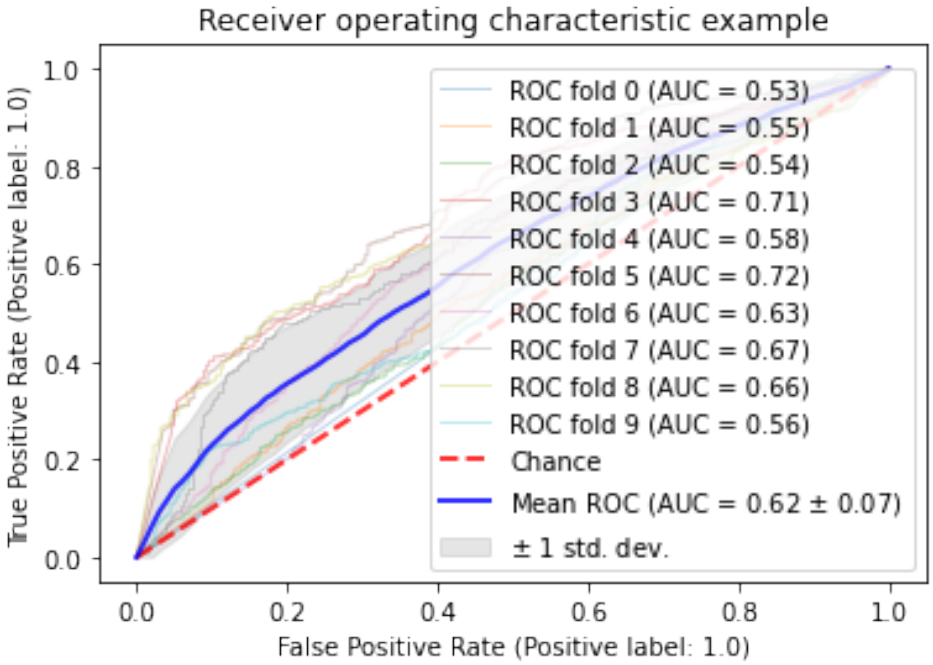

Figure 5.9: ROC State Naive Bayes

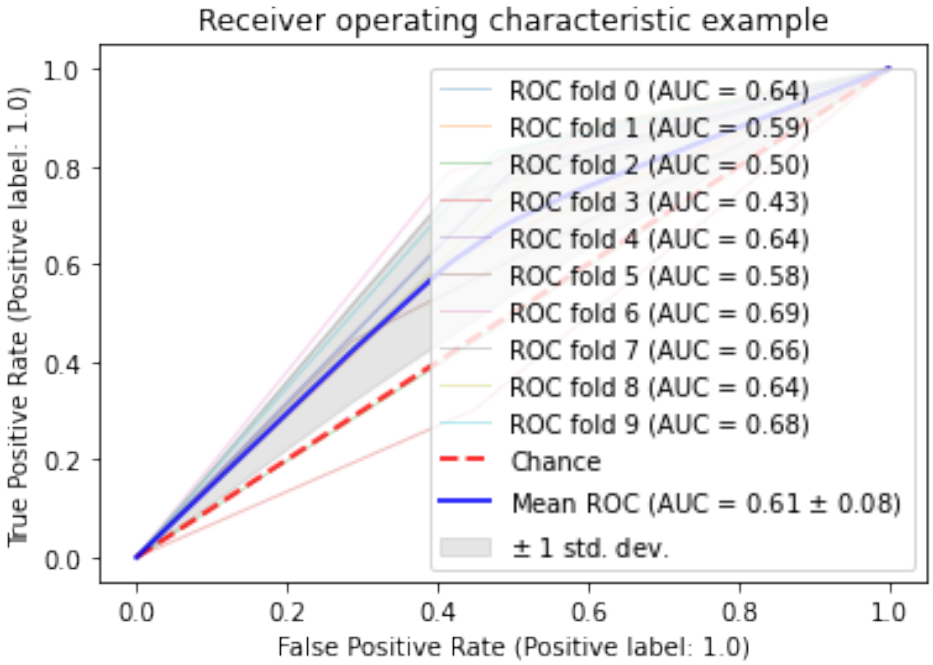

Figure 5.10: ROC State Decision Tree

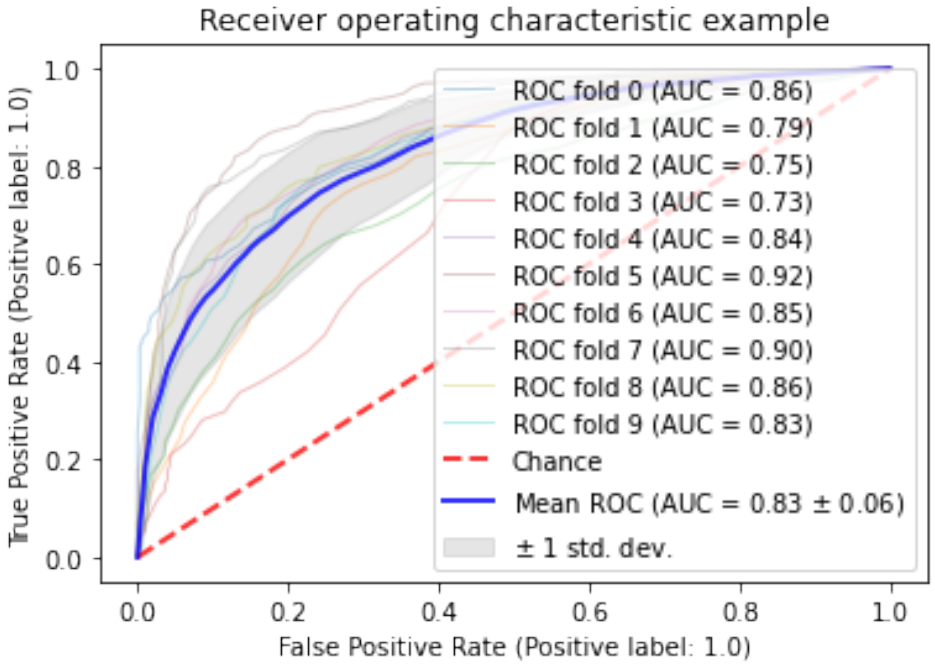

Figure 5.11: ROC State Random Forest

6

Exploring DeepCASE model

This chapter describes the DeepCASE model that implements a state-of-the-art neural network for reducing the number of security events in a SOC. The aim here is to evaluate the performance of the tool on our dataset. We first explain the preprocessing of the data (Section 6.1). Then we define the metrics that we use (Section 6.2). We outline the results (Section 6.3) and we end the chapter with a conclusion (Section 6.4)

6.1. Preprocessing of data

To evaluate the performance of the implementation of DeepCASE in a SOC, it needs to be assessed with a dataset of real SOC tickets created by the SIEM. As explained in the background in Chapter 2 the SIEM will correlate events and send them to the SOC as one ticket. Via the ticketing system, these tickets are stored and the security analysts will access these tickets and analyse the tickets. After this analysis, the security analyst will close the ticket. The time of the ticket created and the ticket closed will be calculated and stored as the processing time. It is useful to pick this ticketing system dataset for our evaluation of the DeepCASE model because we can calculate how much time the model reduces while clustering the events.
This dataset is similar to the dataset that is used in the data analysis in Section 5. Now it needs to be altered in some way for it to work on the DeepCASE model. In the next section, we will explain what needs to be modified before using DeepCASE on the dataset. To make the explanation more clear, Table 6.1 shows an example of the KPN dataset.

Table 6.1: Data from KPN

Title	**Created**	**Dienst**
Windows Defender ATP - microsoft_atp: Outbound connection to IP with a history of unauthorized access attempts	12-9-2020 19:20	Windows Defender ATP
[KPN intern] IDS - sourcefire - SQL generic sql with comments injection attempt - GET parameter [1:16431] - Prio: High	24-12-2020 12:20	IDS

DeepCASE solely uses the features *Event*, *Machine*, *Timestamp*. The feature *Event* is where the information about the alert is stored. In Table 6.1 can be seen that the column table is the feature where the information is stored. The *Machine* feature is the device for which the event is created, for example *nl-zl2-idps-ss01* and *nl-zl2-idps-ss02*. The *Service* column is the column where the relation is most similar to this explanation. The *Timestamp* is the time of the event that is alerted and is the column name *Created* in Table 6.1.
The Title will remove '[KPN intern] IDS - sourcefire -' and 'Prio: High' because not in every dataset this is used consistently. We will also remove [1:16431]. The modified version of the table is shown in Table 6.2.
Van Ede et al. use the Hadoop Filesystem (HDFS) dataset to verify the model with the LastLINE dataset [26]. The HDFS dataset is an open-source dataset used in the security log analysis tool DeepLog [43]. Because the HDFS dataset is open-source, thus available for everyone to use, we will test our model with the HDFS dataset. The dataset consists of log entries generated by over 200 Amazon EC2 nodes. We will use the training dataset, which consists of 44915 events. We use the training dataset because the number of events is comparable to the number of events in our ticketing system dataset. The dataset was labelled by experts as normal and anomalous events. It does not contain context-explaining information, exclusively numbers. Therefore, we

Table 6.2: Modified Data from KPN

Title	Created	Dienst
Windows Defender ATP - microsoft_atp: Outbound connection to IP with a history of unauthorized access attempts	12-9-2020 19:20	Windows Defender ATP
SQL generic sql with comments injection attempt - GET parameter	24-12-2020 12:20	IDS

can only use the dataset to compare the model with our dataset and verify that the model works. We will use the training dataset, as this dataset is roughly the same size, which is good for comparing the datasets.

6.2. Metrics

Different metrics are used by van Ede et al. such as the percentage of the tickets removed by the model [26]. This section is explained what metrics we use to verify and validate the results of the model in a ticketing system.
We expect that the tickets that will be removed are easier to recognise than the tickets that are not clustered by the model. The easy tickets are expected to cost less time than the hard ones. Therefore, we expect to see a lower ticket processing time in the clustered ticket than in the not-clustered tickets.

In the Context Builder the model uses a metric that defines how well the model predicts the next events. This is expressed by the Kullback-Leibler divergence as the loss function, which is explained in Section 2.4.5. We will use these metrics as well to compare the HDFS dataset with our dataset of the SOC.

The Interpreter will use different evaluation metrics. One of which is the number of clusters that the Interpreter can create from the events.
Another metric that flows directly from the number of clusters is the percentage of tickets that are removed. During the semi-automatic analysis, there are 10 samples from every cluster taken. 10 alerts are shown to the security analyst as one cluster. In their research, these samples will be shown to the security analyst and these analysts will define if the clusters are malicious or benign. There will be 10 events per cluster including their context per cluster shown to the analyst.

The *total_workload_reduction* is measured as the number of alerts sent to the security analyst by the tool and the *uncovered_events* divided by the *total_events*. The number of alerts that are sent to the security analyst is based on 10 sequences per cluster. The uncovered events are the alerts that are not able to cluster in a specific cluster.

$$total\_workload\_reduction = 1 - \frac{\#samples + \#uncovered\_events}{\#total\_events} \tag{6.1}$$

The percentage of tickets that will be reduced is the number of clusters times 10 divided by the total number of tickets. The number of samples is the number of clusters times 10.

$$samples = 10 * clusters \tag{6.2}$$

We take another metric that is useful to evaluate alert fatigue in the SOC. The SOC measures how much time an analyst has to spend on each ticket and is noted in our dataset. We can use this information to find out how much the merged tickets takes time. Do merged tickets take less time than uncovered tickets?

6.3. Results

The results will now be evaluated. We first compare the results of the ticketing system dataset with the HDFS dataset. Then we show more results of the ticketing system dataset. Afterwards, we give an in-depth analysis of the clusters. We end this section with a conclusion.

6.3.1. Comparing the results with the HDFS dataset and ticketing system dataset

The HDFS dataset is compared to the ticketing system dataset. The metrics explained earlier are compared to see which dataset performs better. Since we do not have information about the false positives in the HDFS

dataset, we only use the HDFS dataset for evaluation in terms of workload reduction and not in accuracy or time reduction. Thus, we only use the HDFS dataset to verify the model.

During the training of the ContextBuilder, the model will train to predict the next event that follows the context events. The loss of predicting the next event per epoch of both datasets can be seen in Figure 6.1. Of both datasets the loss decreases per epoch, which is what is expected because while training the model, the model will learn to predict the correct event. The performance of the model will increase and thus the loss will decrease per epoch.

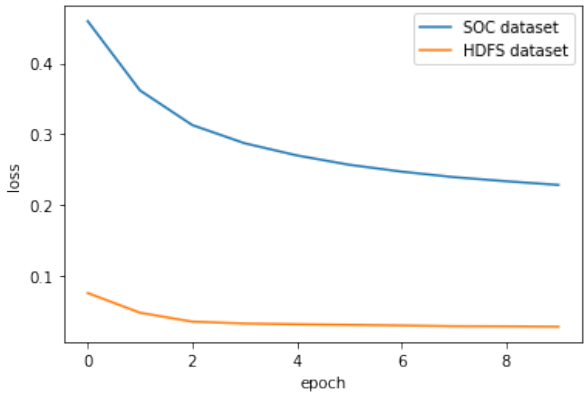

Figure 6.1: Loss comparison

Overall the loss of the HDFS is lower than the loss of the ticketing system dataset. We can see that after 10 epochs the loss of the ticketing system dataset is around 0.2 and 0.3. And the loss of the HDFS dataset is almost 0. This means that the ticketing system dataset is more difficult for predicting the next event than the HDFS dataset. This has to be taken into account when clustering because the clustering relies on the trained ContextBuilder.
We now compare the results obtained from the Interpreter and use multiple metrics to compare them. After clustering the similar attention vectors we evaluate the number of clusters and how many of the tickets will be removed when we do the manual analysis.

We also evaluate the percentage of tickets that are removed. Combining these metrics, we can calculate the *total_workload_reduction*. In Table 6.3 we see the results of the comparison between the SOC and HDFS datasets.

Table 6.3: Comparison of HDFS dataset and ticketing system dataset

	Number of events	**Min loss**	**Number of clusters**	***total_workload_reduction***
HDFS	95,125	0.0267	320	92.0%
Ticketing system dataset	20,744	0.2923	129	47.4%

We can see that the HDFS dataset has more events. The HDFS dataset is also performing better than the ticketing system dataset. In the paper of van Ede et al., the workload reduction was 92.26%. The dataset that is used in this research is working as expected but it is not able to cluster as many events as the HDFS dataset.

6.3.2. Results of ticketing system dataset

In Table 6.4 we see the results of the clustering. The implementation can identify 154 clusters. The number of events that the implementation can identify within a cluster is 9427. From all events, 11,347 cannot be clustered. When we calculate the workload reduction of the SOC using DeepCASE we can see in Equation 6.1 that the *total_workload_reduction* is 47.4%.
We see that there is a significant difference in the ticket duration between the clustered tickets and the uncovered tickets. This validates our hypothesis that the tickets that can be clustered are easy. This means that the *total_workload_reduction* is not saying that the workload is reduced. The metric does not take into account the time of security analysts and we have seen that time does play a role in workload reduction.

Table 6.4: Results ticketing system dataset

	Number of tickets	Ticket duration (in min)
Clustered	9427	142.81
Not clustered	11,347	385.83

6.3.3. Analysis of clusters

The analysis of the clusters is done by going through all clusters one by one and looking for interesting connections between the events and the context of the events. We found for example the cluster which outputs mining activity. The results are shown in Figure 6.2, where the events are only the number 849. The explanation of how the results are displayed is shown in Appendix C. We see that before the event the context consists of repeating 848 and 849. The contexts are similar, which show why they have been clustered.

```
Cluster: 30
Context: tensor([[849, 848, 849, 848, 849, 848, 848, 849, 848, 849],
        [848, 849, 849, 848, 848, 849, 848, 849, 848, 849],
        [849, 848, 849, 848, 849, 848, 848, 849, 848, 849],
        [848, 849, 848, 840, 849, 848, 848, 849, 848, 849],
        [848, 849, 848, 849, 849, 848, 848, 849, 848, 849]])
Events: tensor([849, 849, 849, 849, 849])
848 IDS - sourcefire: PUA-OTHER Cryptocurrency Miner outbound connection attempt [1:46237]
849 IDS - sourcefire: PUA-OTHER XMRig cryptocurrency mining pool connection attempt [1:45549]
```

Figure 6.2: Miner cluster

There exist other events that are clustered because in the context the number that is in the event is repeated before the actual event. For example, in Figure 6.3 we can see that most events of the contexts are 853. This is exactly why this cluster has been created because the events are repeating.

```
Cluster: 16
Context: tensor([[839, 838, 865, 839, 840, 853, 853, 853, 853, 853],
        [838, 865, 839, 840, 853, 853, 853, 853, 853, 853],
        [865, 839, 840, 853, 853, 853, 853, 853, 853, 853],
        [839, 840, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [840, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853],
        [853, 853, 853, 853, 853, 853, 853, 853, 853, 853]])
Events: tensor([853, 853, 853, 853, 853, 853, 853, 853, 853, 853, 853, 853, 853, 853])
853 IDS - sourcefire: SERVER-APACHE Apache Struts remote code execution attempt [1:49376]
```

Figure 6.3: Cluster Remote Code Execution Attempt

We have found a lot of repeating events in the context as explained above. Unfortunately, we have not found interesting attack patterns in the clusters. The goal of DeepCASE was to find these patterns, as the context is included in the analysis. We need more research on the way these patterns can be found by the model.

6.4. Conclusion

There is a significant difference in the results when the ticketing system dataset is used and when the HDFS dataset is used on the DeepCASE model. In Table 6.3 the ticketing system dataset is performing not as well as the HDFS dataset.
Moreover, we hypothesised that the alerts that have been clustered are quite easy. We expected to see that the average ticket duration of the clustered tickets is lower than the tickets that are not clustered. The results in Table 6.4 show that this hypothesis is correct. We have seen that the number of tickets removed is 47.4% and the time that is reduced is on average 142.81 minutes. This is less than the average ticket duration of the events that were not clustered, which was 385.83 minutes. The average ticket duration of the clustered events is less than the rest of the events that were not clustered. We have seen that the metric *total_workload_reduction* does not say much about the actual workload of the security analyst that has been reduced.
From the analysis of the clusters, we have gained insight into what kind of events are clustered and therefore removed for the security analyst. There are a lot of clusters created and the algorithm was able to reduce the number of tickets. Considering that the DeepCASE model mostly clusters the events that are only the simple events that have been clustered, for example only repeating events are clustered, one could ask why such an algorithm is needed to cluster the events to reduce the workload. Perhaps there could be easier ways to improve on this model and create an easier model, for example counting the number of events that are repeated and clustering them and giving them to the security analyst.
Thereby, we need more research on what the difficult events and the sophisticated attack patterns are and why these can not be clustered by DeepCASE. We need more research on how difficult clusters can be created. In Chapter 7 we try to find an answer to this problem.

7

Creating new algorithms for reducing the workload

In this chapter, we build upon the DeepCASE model explained in Chapter 2 and the results of Chapter 6. The possibilities of DeepCASE are further explored. First, we explain the datasets and algorithms that are used in this chapter (Section 7.1). Then, we explore the results (Section 7.2) and we end the chapter with a conclusion (Section 7.3).

7.1. Explanation of the datasets and algorithms

In this section, we explain different datasets and explain what algorithms are used for our experiments. The goal is to remove duplicates from the dataset and to discover more correlations than can be found with duplicates.

7.1.1. Datasets

An overview of the datasets that are used here can be seen in Table 4.8. First, DeepCASE is performed on a dataset that is the largest: the IDS dataset before the SIEM. The SIEM already introduces correlation between events and clusters them before showing it to the security analysts. It is useful to compare these results with the other dataset to examine what the SIEM does with the alerts.
The second dataset is used is extracted from the ticketing system dataset. This dataset was used in the previous chapter. In this chapter, we only use the IDS tickets of this dataset. The SIEM processes the incoming IDS events and creates a dataset that is already enormously smaller than the dataset before the SIEM. This dataset is also our starting point for the rest of the algorithms. We try to reduce this dataset to a point where no duplicates exist.

7.1.2. Algorithms

In this section, we create a new model that uses a heuristic to remove the merged tickets from the dataset. By this means, a new dataset is created and this dataset is used in the enhanced DeepCASE model. Below we explain both algorithms.

Remove duplicates Given is the next example in Table 7.1. The events are removed if the events are similar to an earlier alerted malicious event. The idea behind this process is to remove the repetitive alerts in the dataset, such that the analyst does not have to check similar alerts twice.
As an example, the two events in Table 7.1 are produced by the same attacker. This attacker is attempting to inject a SQL statement into a text field. The second event comes in 3 seconds after the first event. The source IP, destination IP and title are the same. Possibly the attacker tries to attack the target twice using the same method. We assume that when two events have a similar source IP, destination IP and alert title and happen within a given time frame, the two events originate from the same attack. The next paragraph describes how these duplicates are deleted.
To remove the duplicates we use a simple algorithm that is shown below. First, the data is looped over once and then the data runs again over their next tickets until the tickets are no longer within the given time range.

Table 7.1: Example similar events

Event	Timestamp	Source IP	Destination IP
SQL 1 = 1 - possible sql injection attempt	2021-05-28 02:50:15	x.x.x.1	x.x.x.2
SQL 1 = 1 - possible sql injection attempt	2021-05-28 02:50:12	x.x.x.1	x.x.x.2

If there is a ticket in the second loop that has the same source IP, destination IP and title then this event is removed from the list.

```
duplicateslist = dataframe
for i in 0:dataframe:
    for j in i+1:dataframe:
        if created time[i] – created time[j] < timeframe    #events within the given timeframe
        and created time[i] – created time[j] is not 0
        and source ip[i] is source ip[j]
        and destination ip[i] is destination ip[j]
        and title[i] is title[j]):
            if event exists in duplicates list:
                remove from duplicates list
        else: break
```

The example in the previous paragraph is extended in Table 7.2. This table shows a ticket that is removed from the dataset because it is a duplicate and it shows a ticket that is not removed. The events in bold are duplicates from the first event because they match the given criteria. The third event is not similar because it has a different event title and Destination IP. The last event looks similar as well but has the property that it has been alerted the next day.

Table 7.2: Example similar events - removed, bold is removed

Event	Timestamp	Source IP	Destination IP
IDS - sourcefire - SQL 1 = 1 - possible sql injection attempt	2021-05-28 02:50:12	x.x.x.1	x.x.x.2
IDS - sourcefire - SQL 1 = 1 - possible sql injection attempt	**2021-05-28 02:50:15**	**x.x.x.1**	**x.x.x.2**
IDS - sourcefire - remote execution attempt	2021-05-28 02:50:18	x.x.x.1	x.x.x.3
IDS - sourcefire - SQL 1 = 1 - possible sql injection attempt	**2021-05-28 02:50:20**	**x.x.x.1**	**x.x.x.2**
IDS - sourcefire - SQL 1 = 1 - possible sql injection attempt	2021-05-29 02:50:15	x.x.x.1	x.x.x.2

Using the duplication heuristic it is helpful to find out which time frame is best to use in the rest of our experiments. When the timestamp is small the algorithm is not able to find all duplicates, because an attack may take longer. When the algorithm has a bigger time frame, the chance is that tickets are incorrectly classified as duplicates. Running multiple experiments will define the best outcome for the time frame. Beginning small with 2700 seconds (0.75 hours) and we double that until we hit 32 days.

Compression algorithm After removing the duplicates we use the created dataset, without duplicates, to see if we find more sequences of events that can be merged. How we do this is explained next.
We have already seen in Chapter 2 in Section 2.5.2 that when multiple context events are important, DeepCASE has difficulty dividing the attention to all important events. What DeepCASE usually does is that it assigns attention to one event instead of all important events. With this knowledge, we aim to enhance DeepCASE, such that it can assign attention to the important context events in the sequence as well.
We use the same method we used in the toy example in Section 2.5.2 by removing the event with the most attention. The way we do this is to run DeepCASE on the dataset that has removed the duplicates. Then, the context of the clustered events is examined by the attention query. The goal is to find the event with the highest attention. Hereby we find what event has the most impact on the predicted event. After finding that context event this event is removed from the dataset. Using this technique we hope to find more correlations in the dataset after removing the event with the highest attention value.
In the example below, the context event 40 has the most attention, because it is available in all the tensors. The context events that has the number 40 is be removed from the dataset.

```
Events: tensor([40, 40, 40])
Context: tensor([[163, 163, 163, 163, 163, 163, 163, 163,  40,  40],
        [163, 163, 163, 163, 163, 163, 163, 163,  40,  40],
        [163, 163, 163, 163, 163, 163, 163,   9,  40,  40]])


tensor([[0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.1229,
         0.8586],
        [0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.0023, 0.1229,
         0.8586],
        [0.0022, 0.0022, 0.0022, 0.0022, 0.0022, 0.0022, 0.0022, 0.0011, 0.1165,
         0.8667]])
```

Figure 7.1: Events

We repeat this sequence until we do not find any more clusters. In Figure 7.2 is an overview shown of the algorithms.

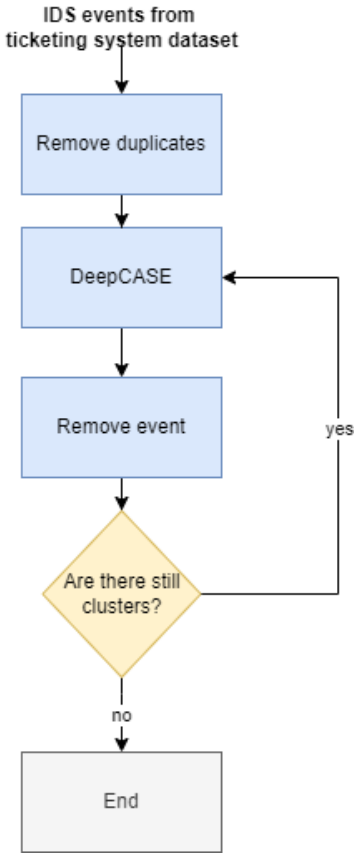

Figure 7.2: Extended model

7.2. Results

7.2.1. Duplicate algorithm

As can be seen in Figure 7.3, the longer the time frame is, the more tickets can be found. Most duplicates can be found within the first 24 hours of the attack but there are still duplicates that can be found after 1 week. When the time frame is too short the time frame is not able to see all duplicates. The first part of the graph is growing exponentially but at some point, this changes into a linear line.

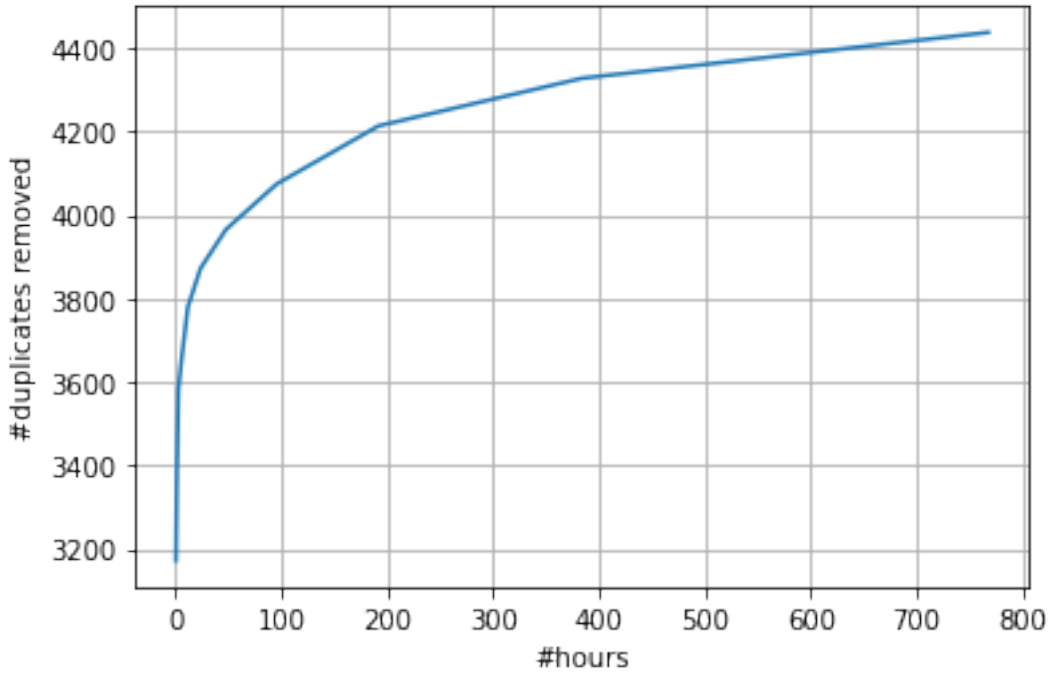

Figure 7.3: Time frame comparison

This is interesting because this does not support the expectation that at a certain point no duplicates can be found. These findings have led to the conclusion that attackers target the same victims with similar means. It is not desired that events are being falsely classified as duplicates. Therefore, we assume that one attack ends after 1 day. We then still have enough duplicates, but we do not risk events being falsely classified as duplicates.
The algorithm can reduce 3872 events of the IDS events in the ticketing system dataset. This is 55.19% of all IDS events in the ticketing system dataset. When we compare that with the number of events that DeepCASE can remove this is more than DeepCASE does with 3573 number of events for removing from the IDS dataset. This is 50.93% using the same dataset.

7.2.2. Comparison of different datasets using DeepCASE

In Table 7.3 can be seen that when the number of events goes down, the loss increases. When there is a high correlation between different alerts, the model can predict the next event in the sequence better. The dataset before the SIEM has the lowest loss ánd has the largest dataset. This is because before the SIEM the events are not filtered. Thereby, the events are strongly correlated. After the filtering of the SIEM, the dataset has fewer events. Now the events are less correlated. This can be seen in the results of the loss. Duplicates have a strong correlation with each other. When the duplicates are removed from the dataset, the loss increases further.

Table 7.3: Results datasets

	Loss	**Number of events**	**Number of clusters**	**Number of tickets clustered**	**Workload reduction**
IDS before SIEM	0.0383	5,000,000	1590	4,856,616	96.81%
IDS after SIEM	0.2296	7015	45	3573	55.85%
After remove duplicates	0.3350	2993	1	88	2.94%
After compression algorithm	0.3415	2918	0	0	0%

7.2.3. Take the loss as a measure

To measure how well the dataset is correlating with the other events, we can use the metric used to measure the correctness of the prediction of the next event in the dataset. The loss that DeepCASE uses is the Kullback-Leibler divergence. This indicates how a probability distribution of the predicted event is different from the actual event. What is not yet known is how much correlation still exists in the dataset.

7.2.4. Compression algorithm

Now we zoom in to the compression algorithm. This algorithm was made to find more clusters than initially found before. The intuition here is that when we remove the context events that have the highest attention, we can find more events that also have a strong correlation with the event. The compression algorithm does not work as expected. The number of times the algorithm runs is exactly one.

7.2.5. Analysis security analyst - compression algorithm

The analysis of the created clusters can be enhanced by the review of a security analyst at KPN. Security analysts see alerts daily and know best when an alert is a duplicate, a false positive or true positive. We give the security analyst two datasets and a set of questions. The questions are:

- Are there events that are clustered that should not be clustered?
- Events that can yet be clustered?
- Where in the SOC could the DeepCASE model be deployed?

The first dataset is before removing the duplicates and running the DeepCASE model. The second dataset contains only the events that have not been removed by the heuristic or the DeepCASE model. The analyst compares both datasets using the questions above.

Events that are clustered that should not be clustered The first question we had was if the events that are removed are legitimately removed. Or are there events removed that must not be removed? According to the security analyst, this is not the case. He can see that all source and destination IPs are there, but the duplicates are removed. So no source and destination IPs are missed.

Events not clustered that can yet be clustered The second question was if there were events that could still be clustered to other events but are missed by the model. This was the case when different alerts from the same attack were created. This mostly happened when a vulnerability scan in the network appeared. Such a scan searches for different vulnerabilities on different devices, thereby creating distinct alerts. An example of a scan like this is shown in Table 7.4. The scans appear fairly often in the dataset but could not be found by the models. For example:

Table 7.4: Example vulnerability scan

Event	Timestamp	Machine	Source IP	Destination IP	Destination Port
Apache Struts remote code execution attempt - GET parameter [1:21072]	3-6-2021 09:05	Machine1	IP1	IP2	80 (http) / tcp
Apache Struts remote code execution attempt - DebuggingInterceptor [1:21075]	3-6-2021 09:05	Machine1	IP1	IP2	80 (http) / tcp
Apache Struts remote code execution attempt [1:39191]	3-6-2021 09:05	Machine1	IP1	IP2	80 (http) / tcp
Apache Struts allowStaticMethodAccess invocation attempt [1:21073]	3-6-2021 09:05	Machine1	IP1	IP2	80 (http) / tcp
Apache Struts remote code execution attempt [1:39190]	3-6-2021 09:05	Machine1	IP1	IP2	80 (http) / tcp

Where to deploy DeepCASE in the SOC The third question was if the DeepCASE model could be installed in the SOC and where in the SOC could this then be placed. The answer to this question is two-fold.

First, where in the SOC can DeepCASE be installed. The answer was to be before the SIEM because DeepCASE turns out to be fairly suitable for the processing of raw unfiltered network events, as was shown in Table 7.3. He explained that he would make clusters out of the IDS data and instead of sending all IDS events to the SIEM, he can send a sample of clusters to the security analyst.
He added that the use of the heuristic is more interesting because it is easy to install and understand. The duplicates can be removed fairly good. As the analysts are security experts but no machine learning experts this would also be a great option to install in the SOC.

7.2.6. Confidence threshold

The number of clusters depends on the confidence output of the attention query. The clustering algorithm has a certain confidence threshold and the attention query must pass this threshold to create a cluster. DeepCASE works as such that if the confidence threshold is not reached, the sequence is passed to the security analyst without a cluster.
Using the dataset after the removal of the duplicates we use different confidence thresholds when the value is too low or too high. The first confidence threshold is 0.2. The next example is clustered by the algorithm. We see that the confidence threshold is above 0.2 and can therefore be clustered. In appendix C the context and events for the cluster found here are displayed.

```
[0.2607, 0.2599, 0.2601, 0.2603, 0.2587, 0.2590, 0.2590, 0.2611, 0.2600,
0.2600, 0.2600, 0.2587, 0.2605, 0.2581, 0.2585, 0.2600, 0.2582, 0.2555,
0.2602, 0.2598, 0.2579, 0.2610, 0.2591, 0.2602, 0.2584, 0.2604, 0.2599,
0.2559, 0.2604, 0.2594, 0.2602, 0.2603, 0.2561, 0.2590, 0.2603, 0.2595,
0.2546, 0.2606, 0.2593, 0.2534, 0.2561, 0.2584, 0.2559, 0.2568, 0.2587,
0.2594, 0.2591, 0.2598, 0.2578, 0.2608, 0.2598, 0.2558, 0.2595, 0.2587,
0.2580, 0.2605, 0.2605, 0.2605, 0.2602, 0.2592, 0.2600, 0.2601, 0.2590,
0.2602, 0.2602, 0.2608, 0.2591, 0.2590, 0.2604, 0.2586, 0.2584, 0.2596,
0.2588, 0.2603, 0.2573, 0.2581, 0.2590, 0.2607, 0.2579, 0.2598, 0.2590,
0.2598, 0.2590, 0.2582, 0.2604, 0.2562, 0.2603, 0.2595]
```

Figure 7.4: Confidence of cluster that is found by DeepCASE

Now we lower the confidence threshold to 0.1. The number of events that are left is 2835, which is quite a difference compared to the confidence threshold of 0.2. The number of events that are clustered that can be found here is 196 and the number of clusters is 4. Below is an example of an attention vector that has a confidence level between 0.1 and 0.2. What is important to note is that we can find context events that are different from the next event.
At confidence threshold 0.07 the algorithm started to work. DeepCASE was able here to cluster tickets after running the model for the second time. In Table 7.5 we show how much time the model has run at the confidence threshold.
When the threshold is set at 0.05, it can be seen in Figure 7.6 that context events are no duplicates of the events.
Though there are signs of sequences that do not belong together. For example in Figure 7.7. As can be seen, the context events do not have any relationship with the next event, because 163 is no event, this is the padding event. In this regard, these events should not be merged into one cluster. Instead, these sequences should be forwarded to a security analyst individually. The analyst can examine if the clusters are correctly clustered together.

7.2.7. Analysis security analyst - confidence threshold

For the analysis of the different clusters, we showed samples of the clusters to the security analyst. The analyst can use his knowledge about the alerts to accurately see which clusters should be together and which should not. We show the clusters to the analyst using the context event with the highest attention and the event itself. Especially the lower confidence thresholds, there were many clusters, therefore we selected clusters that were interesting and unique. A summary of the sample clusters is shown in Appendix D. Every cluster has its name we use in the analysis. In the appendix, this is explained as well. For the clusters with confidence thresholds 0.09 and 0.1, the events showed all logical correlations in the clusters. In some clusters, the context events

```
Event:
[42, 42, 42, 42, 42]
Context event:
[163, 163, 163, 163, 163, 163, 163, 163, 163, 39]
[163, 163, 163, 163, 163, 163, 163, 163, 163, 39]
[163, 163, 163, 163, 163, 163, 163, 163, 163, 39]
[163, 163, 163, 163, 163, 163, 163, 107,  45, 39]
[163, 163, 163, 163, 163, 163, 163, 163, 107, 39]

Confidence threshold: [0.1750, 0.1750, 0.1750, 0.1762, 0.1758]

Attention vector:
[0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.9841]
[0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.9841]
[0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.0018, 0.9841]
[0.0012, 0.0012, 0.0012, 0.0012, 0.0012, 0.0012, 0.0012, 0.0029, 0.0122, 0.9764]
[0.0015, 0.0015, 0.0015, 0.0015, 0.0015, 0.0015, 0.0015, 0.0015, 0.0051, 0.9833]
```

Figure 7.5: Cluster with different context event

```
Event: tensor([63, 63, 63, 63, 63])
Context events: tensor([[163, 163, 163, 163, 163, 163, 163, 163,  91, 103],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 103],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 103],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 103],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 103]])
```

Figure 7.6: Context events and events

Table 7.5: Comparison of confidence threshold

Confidence threshold	**Number of clusters**	**Number of times DeepCASE is executed**	**Number of events left**	**Loss**
0.2	2	1	3053	0.3356
0.1	4	1	2851	0.3417
0.09	9	1	2818	0.3453
0.08	10	1	2789	0.3464
0.07	21	2	2612	0.3524
0.06	32	4	2644	0.3513
0.05	68	10	2471	0.3563

and events were all the same, for example, *0.09_cluster_9_number_1*. Some clusters showed some interesting vulnerability scans, for example, *0.09_cluster_8_number_1*.
When we set the confidence threshold to 0.08, there were some clusters that were found to be logical:

- *0.05_cluster_14_number_1*
- *0.05_cluster_24_number_1*
- *0.07_cluster_8_number_1*
- *0.08_cluster_5_number_1*
- *0.08_cluster_15_number_1*

The rest of the clusters showed some anomalies in the clusters. In *0.08_cluster_13_number_1* the clusters there were two different events in the cluster. The first event is a Citrix arbitrary code execution attempt

```
[68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68, 68,
        68, 68, 68, 68])
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163],
        [163, 163, 163, 163, 163, 163, 163, 163, 163, 163]
```

Figure 7.7: Context event no relationship with the next event

and the second event is an Apache Tomcat FileStore directory traversal attempt. DeepCASE clustered these events incorrectly, according to the security analyst, the two services should not be clustered together. In *0.06_cluster_3_number_7* there was one event that showed an Apache Struts access attempt. The other events in this cluster were SQL injection attempts. These should not belong together.

7.3. Conclusion

In this chapter, we aimed at enhancing the state-of-the-art neural network to improve the results obtained previously. We first removed duplicates by using a simple heuristic. Then we implemented an algorithm that removed the context event with the highest attention value from the list of events. Then we ran DeepCASE again until no other clusters were found.

Using the heuristic the number of tickets can be decreased. Comparing this to the number of tickets DeepCASE can remove we see that the simple algorithm is good as well. Also, the analyst that was asked to give an analysis of the clusters was especially interested in the method of the heuristic because of the ease of use of this method.

At a confidence threshold of 0.07 or lower, the algorithm ran more than 1 time which means that more events can be found when the context event with the highest attention value is removed. We have evaluated when the confidence threshold was too low, to see if this.

We have evaluated the clusters by an analyst and found that at the confidence threshold of 0.08 or lower, the clusters began to show anomalies. We can conclude that the confidence threshold is best set at 0.09 because this has no anomalies in the clusters and the number of clusters that are made is better than 0.1. This, unfortunately, was not at the level that the enhanced algorithm started to work.

We see an increase in the number of clusters when the confidence threshold is lower. What is not yet known is how the confidence threshold affects the number of false negatives. We have seen that when you set the threshold too low the sequences being clustered do not belong together.

We have seen that the loss goes up when duplicate events are removed and the number of events is less. One way of using this measure is to use it as a measure for the correlation between events, this could be a good option to use in a SOC.

8

Conclusion and discussion

In this research, we have evaluated different metrics and methods for reducing the workload of security analysts in the SOC. In Section 8.1 we discuss several limitations. In Section 8.2 we conclude explain the answers to the research questions and we suggest the possibilities for future work in Section 8.3.

8.1. Discussion

We begin with a discussion of several limitations in our work.

8.1.1. False positive check

In the data analysis in Chapter 5, one major challenge was that the label *False Positive* in the dataset of the ticketing system was not always accurate. Whenever an alert occurred in the system that turned out to be a false positive, the analysts needed to mark it as a false positive themselves. In the ticketing system, there is a check box that the analyst needs to check when the alert is a false positive. The problem is that analysts are not always consistent in checking this box. Due to human error, the analysts can forget to check the box or accidentally check the box when it was not a false positive.
It is also important to note that when the parent ticket was a false positive, its merged tickets were not checked as false positive. We needed to modify the false positive merged ticket to match with the parent ticket. By this means we have verified that analysts are not always consistent in using the check box for a false positive. This will influence the prediction of the machine learning models because the models learn when an alert is a false positive. When a true positive is a false positive, the model will learn the wrong label.

8.1.2. *Service* as *Machine* in DeepCASE

In Chapter 6, we used the feature *Service* as the feature *Machine* that needed to be used in the DeepCASE model. The feature *Machine* in the DeepCASE model has a slightly different definition than the feature *Service* in the ticketing system dataset. *Service* is the way the SOC describes the type of device it is using, such as IDS, Firewall etc. *Machine* is the particular device in the network, such as *nl-zl2-idps-ss01* and *nl-zl2-idps-ss02*. *Service* is less specific because there are multiple Intrusion Detection Systems in the network.

8.1.3. Improvements in the SOC

Security analysts are constantly improving the way they operate. At the time of the data collection, several considerable changes were implemented to improve the information in the tickets. At the beginning of the dataset, merged tickets had multiple empty columns, such as *Service* and *False Positive*, as explained in Section 4.2. After a while, the columns were filled and we could use these as well. We modified the columns manually by using the information of the parent ticket. When this parent ticket does not exist or was too far in the past, the merged ticket was simply removed from the dataset. This should be taken into consideration because this could have influenced the results of predicting labels as mistakes can be made when modifying the dataset manually.

8.1.4. Loss interpretation

The results in Section 7 showed that the loss could be a good metric to define the amount of correlation in a dataset. We could show this metric to the security analysts, who can use this metric to see if there are many tickets in the ticketing system that should be merged to other tickets.
The analyst could have difficulty interpreting the loss and when the loss is good. It is not known yet what ticket is indeed merged. Therefore, the security analysts will not know what parts of the data are unique and what are duplicates. Moreover, we saw an increase in loss when the amount of correlation went down but this research did not connect the exact amount of correlation of the dataset to the loss.

8.2. Conclusion

In this study, the goal was to explore the possibility of implementing machine learning models in a SOC. We first explored different metrics to use as a metric for machine learning models. Then, classical machine learning methods tried to predict false positives in the ticket dataset. Next, a state-of-the-art neural network is used to examine this technique on our ticketing system dataset. We also built a simple heuristic that is compared with the rest of the models. At last, a new approach for evaluating the dataset is introduced. The sub-questions as defined in Chapter 1 are answered below:

8.2.1. What are existing metrics for measuring the performance of security analysts in a SOC?

To answer this question, a literature study has been done to find the metrics for measuring the performance of machine learning models in a SOC. Multiple metrics are explained in the literature study. Often they were defined differently but they have been combined into consistent metrics. An overview of these metrics can be found in Table 3.1. We include multiple challenges of these metrics that some of the authors have raised.

One problem of the metric 'number of tickets created' was that the metric often does not take into account the number of false positives. The first problem is that when an analyst takes a long time to analyse an event that turns out to be a false positive, no ticket will be created. When this metric is used, the number of false positives is of negative influence for analysts who research many false positives. We have seen that false positives are not taking more time than true positives but they take on average around 60 minutes (Section 5.1.1).

Moreover, measuring the performance of analysts by using the metric the 'time taken to respond' will encourage the analysts to be as fast as possible to analyse tickets. The analysts will automatically spend less time understanding the root cause of the incident. It will learn less about why the event is happening.

Some of the metrics do not take into account the complexity of the ticket. Especially the number of alerts analysed, the number of tickets closed per day and the average time spent on each ticket. Some tickets are more difficult to examine and the complexity of the ticket decreases the number of tickets that an analyst examines and the average ticket duration.

8.2.2. How do existing machine learning methods perform in a SOC?

We have evaluated different models to examine if the alerts in a SOC can be reduced. First, we tried to classify the labels *False Positive* and *State* with the traditional machine learning models Naive Bayes Classifier, Decision Tree Classifier and Random Forest Classifier. The most important aspect for classifying the label *False Positive* is that a true positive must not be predicted as a false positive. When this happens, the ticket will be discarded by the model. An ideal situation for predicting the label *False Positive* is that the model does not predict that a true positive is a false positive. We use the metric precision to define the model's ability to predict if the true positive is a true positive. Similar to the label *False Positive*, the most important aspect for predicting the label *State* is that a closed ticket (unique ticket) must not be predicted as a merged ticket. In this case, the ticket will be discarded by the model. The precision of these labels shows that the models had difficulty predicting the true positives and the closed tickets correctly.

In Section 5.1.2 we have seen that when a ticket has been merged, the duration of the ticket being evaluated is less than when this is not a merged ticket. We used the model DeepCASE to find out if we can cluster the tickets by using the history of the dataset. We have seen in Table 6.4 that the tickets that are clustered take less time than the tickets that were not clustered. These results support the idea that a machine learning model

clusters tickets that are easy and require little time.

It should be noted that it remains useful to remove the merged tickets because the repetitiveness of the duplicate tickets is causing alert fatigue for the security analysts. Therefore, it is still necessary to improve on these machine learning models and to find good ways to filter the duplicate tickets in the SOC.

8.2.3. How do state-of-the-art neural networks perform in a SOC compared to existing machine learning methods?

To answer this question, we compare DeepCASE with the machine learning methods and various things are important here. The machine learning methods are performing poorly on the ticketing system dataset. The best algorithm was the Random Forest Classifier and it still had a lot of false positives and false negatives. The Random Forest classifier had too many true positives and closed tickets classified as false positives and merged tickets. Implementing these predictors in the SOC will lead to the false investigation of tickets. This means that it is not yet suitable to implement in the SOC.

DeepCASE is better at recognising the next event and recognising similar events. Therefore, it is better at clustering events. However, DeepCASE is a difficult method and many security analysts would have difficulty understanding the inner workings of this method. The model is said to be explainable, but the model remains a black-box model.

What is interesting is that the loss outputted by the ContextBuilder in DeepCASE fits to be a metric for evaluating the correlation of a ticketing system dataset. When a dataset correlates well with other events, DeepCASE is better at predicting the next event. This is directly dependent on the loss that will be lower. Conversely, the loss in a SOC must be as high as possible because a SOC wishes the tickets to be as unique as possible. This means that every attack should be alerted only once.
Interestingly, the simple heuristic that removes the duplicates, is performing quite well. A high number of duplicates were found by the heuristic. The heuristic is even performing better than DeepCASE does. The same dataset gave a better result for the heuristic than DeepCASE. Thereby, the security analysts that were asked to give an analysis were especially interested in the heuristic because of the ease of use of this method.

8.3. Future work

We discovered some improvements in the analysis and the models. We list them below.

8.3.1. Explore the use of multiple metrics in the SOC

This work has shown that the metrics for measuring machine learning models in a SOC are important to focus on in this field of research. We have established that previous work mostly evaluate metrics by conducting interviews and little work has been done on the use of metrics for machine learning in a SOC. These findings show the need for quantifying the value of various metrics for machine learning. One possible approach of this evaluation is to implement machine learning models, then use different metrics to compare the performance. Then, using the models as a comparison we can compare different metrics to explore the possibilities of these metrics.

8.3.2. Use more years of data

Our research was limited to investigating one year of SOC tickets. Especially in Chapter 7, the number of events of the IDS alerts from the ticketing system were 7015, which is not a lot. More years of tickets in the dataset can increase the performance of the models. Future work can add more years of tickets to the dataset to use a larger dataset for the models to train on.

8.3.3. Stream events

Next to the metrics found in the literature review, a new metric has been found. This metric is the loss in DeepCASE used to measure the correctness of the prediction of the next event. The loss turned out to measure the correlation of a dataset very well.
It is useful to investigate whether security analysts would benefit from having information on the correlation of the SOC. At the moment we can only see that the loss is higher when there are fewer duplicates in a dataset.

Future work needs to define a perspective on when the data has more duplicates.
Future work could offer the security analysts the loss produced by DeepCASE to review this metric. Security analysts could be offered two types of methods. The first one is a normal situation where the analysts do their job as normal. The second one would be the one where DeepCASE is implemented and the loss is shown. We could use this information to evaluate when the correlation of the dataset is correct and every alert is one incident.

8.3.4. Detect vulnerability scans

We have discovered that DeepCASE finds it difficult to detect vulnerability scans in the network. We have analysed an enhanced version of DeepCASE to find out if this is possible. We have seen some good results that show signs that this could be possible. We have seen several vulnerability scans, but certainly not all of them. Future work could enhance the detection of vulnerability scans in the network by improving DeepCASE even further.

Bibliography

[1] Steffen Haas and Mathias Fischer. “GAC: graph-based alert correlation for the detection of distributed multi-step attacks”. In: *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 2018, pp. 979–988.

[2] Carson Zimmerman. “Cybersecurity Operations Center”. In: *The MITRE Corporation* (2014), pp. 34–35.

[3] Critical Start. *Why security alert fatigue matters and how to address it*. URL: https://www.criticalstart.com/resources/why-security-alert-fatigue-matters-and-how-to-address-it/.

[4] Jon Oltsik. *Dealing with overwhelming volumes of security alerts*. URL: https://www.esg-global.com/blog/dealing-with-overwhelming-volume-of-security-alerts.

[5] Imperva. *Alert fatigue*. 2021. URL: https://www.imperva.com/learn/data-security/alert-fatigue/.

[6] Diana Kelley and Ron Moritz. “Best practices for building a security operations center”. In: *Inf. Secur. J. A Glob. Perspect.* 14.6 (2006), pp. 27–32.

[7] Faris Bugra Kokulu et al. “Matched and mismatched socs: A qualitative study on security operations center issues”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 1955–1970.

[8] Sandeep Bhatt, Pratyusa K Manadhata, and Loai Zomlot. “The operational role of security information and event management systems”. In: *IEEE security & Privacy* 12.5 (2014), pp. 35–41.

[9] Oskars Podzins and Andrejs Romanovs. “Why siem is irreplaceable in a secure it environment?” In: *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*. IEEE. 2019, pp. 1–5.

[10] I Putu Elba Duta Nugraha. “A Review on the Role of Modern SOC in Cybersecurity Operations”. In: *International Journal of Current Science Research and Review* 4.05 (2021), pp. 408–414.

[11] Ansam Khraisat et al. “Survey of intrusion detection systems: techniques, datasets and challenges”. In: *Cybersecurity* 2.1 (2019), pp. 1–22.

[12] Robbie Allen and Alistair Lowe-Norris. *Active directory*. " OŔeilly Media, Inc.", 2003.

[13] Sathya Chandran Sundaramurthy et al. “A Tale of Three Security Operation Centers”. In: *Proceedings of the 2014 ACM Workshop on Security Information Workers*. SIW Í14. Scottsdale, Arizona, USA: Association for Computing Machinery, 2014, pp. 43–50. ISBN: 9781450331524. DOI: 10.1145/2663887.2663904. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/2663887.2663904.

[14] Issam El Naqa and Martin J Murphy. “What is machine learning?” In: *machine learning in radiation oncology*. Springer, 2015, pp. 3–11.

[15] David D Lewis. “Naive (Bayes) at forty: The independence assumption in information retrieval”. In: *European conference on machine learning*. Springer. 1998, pp. 4–15.

[16] Philip H Swain and Hans Hauska. “The decision tree classifier: Design and potential”. In: *IEEE Transactions on Geoscience Electronics* 15.3 (1977), pp. 142–147.

[17] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[18] Charles E Metz. “Basic principles of ROC analysis”. In: *Seminars in nuclear medicine*. Vol. 8. 4. Elsevier. 1978, pp. 283–298.

[19] Mohamed Bekkar, Hassiba Kheliouane Djemaa, and Taklit Akrouf Alitouche. “Evaluation measures for models assessment over imbalanced data sets”. In: *J Inf Eng Appl* 3.10 (2013).

[20] David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[21] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation Learning: A Review and New Perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[22] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[24] Kyunghyun Cho et al. “Learning phrase representations using RNN encoder-decoder for statistical machine translation”. In: *arXiv preprint arXiv:1406.1078* (2014).

[25] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[26] Thijs van Ede et al. “DeepCASE: Semi-Supervised Contextual Analysis of Security Events”. In: *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE. 2022.

[27] Paul E Black, Karen Scarfone, and Murugiah Souppaya. “Cyber security metrics and measures”. In: *Wiley Handbook of Science and Technology for Homeland Security* (2008), pp. 1–15.

[28] Enoch Agyepong et al. “Challenges and performance metrics for security operations center analysts: a systematic review”. In: *Journal of Cyber Security Technology* 4.3 (2020), pp. 125–152.

[29] Enoch Agyepong et al. “Towards a framework for measuring the performance of a security operations center analyst”. In: *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE. 2020, pp. 1–8.

[30] Sathya Chandran Sundaramurthy et al. “A human capital model for mitigating security analyst burnout”. In: *Eleventh Symposium On Usable Privacy and Security ({SOUPS} 2015)*. 2015, pp. 347–359.

[31] Martin Rosso et al. “SAIBERSOC: Synthetic Attack Injection to Benchmark and Evaluate the Performance of Security Operation Centers”. In: *Annual Computer Security Applications Conference*. 2020, pp. 141–153.

[32] Sean Oesch et al. “An Assessment of the Usability of Machine Learning Based Tools for the Security Operations Center”. In: *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. IEEE. 2020, pp. 634–641.

[33] Pooya Jaferian et al. “Heuristics for evaluating IT security management tools”. In: *Proceedings of the Seventh Symposium on Usable Privacy and Security*. 2011, pp. 1–20.

[34] Martin Husák and Jaroslav Kašpar. “AIDA framework: real-time correlation and prediction of intrusion detection alerts”. In: *Proceedings of the 14th international conference on availability, reliability and security*. 2019, pp. 1–8.

[35] Wajih Ul Hassan et al. “Nodoze: Combatting threat alert fatigue with automated provenance triage”. In: *Network and Distributed Systems Security Symposium*. 2019.

[36] Yuxin Meng et al. “Adaptive false alarm filter using machine learning in intrusion detection”. In: *Practical applications of intelligent systems*. Springer, 2011, pp. 573–584.

[37] Khloud Al Jallad, Mohamad Aljnidi, and Mohammad Said Desouki. “Anomaly detection optimization using big data and deep learning to reduce false-positive”. In: *Journal of Big Data* 7.1 (2020), pp. 1–12.

[38] Muhamad Erza Aminanto et al. “Combating threat-alert Fatigue with online anomaly detection using isolation forest”. In: *International Conference on Neural Information Processing*. Springer. 2019, pp. 756–765.

[39] Dinesh Yadav. *Categorical encoding using Label-Encoding and One-Hot-Encoder*. 2019.

[40] David Cournapeau. *A set of python modules for machine learning and data mining*. URL: `https://pypi.org/project/sklearn/`.

[41] Payam Refaeilzadeh, Lei Tang, and Huan Liu. “Cross-validation.” In: *Encyclopedia of database systems* 5 (2009), pp. 532–538.

[42] Christoph Bergmeir and José M Benıtez. “On the use of cross-validation for time series predictor evaluation”. In: *Information Sciences* 191 (2012), pp. 192–213.

[43] Wei Xu et al. “Detecting large-scale system problems by mining console logs”. In: *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. 2009, pp. 117–132.

A

Heatmap

Different features in the heatmap on the next page shows correlations with other features of the ticketing system dataset. First, *Jaar*, *Kwartaal*, *Week* and *Dag* correlate because they are always dependent on each other. It is also obvious that different devices, such as IDS alert different alerts with different title names. This can be derived from the fact that *Dienst* and *Title* are strongly correlating. More interesting things can be seen. For example, the feature *State* is correlated with the *Doorlooptijd (in min)*. A closed ticket in the system is taking more time than the tickets that are merged.
It is also interesting to see that *False Positive* correlates with *Kwartaal*, *Week*, *Dag*. This could mean that at a certain time in the year, multiple false positives were coming in.
What is interesting is to see that different services have a higher priority than other services. This could be derived from the fact that *Priority* and *Dienst* correlate with each other.
We see that *False Positive* depends a lot on *Title*, *Agent/Owner*, *Priority* and *Dienst* and the time of year. For *State*, this is different because few features do not correlate with the label *State*. Only the *Doorlooptijd (in min)* is important for the feature.

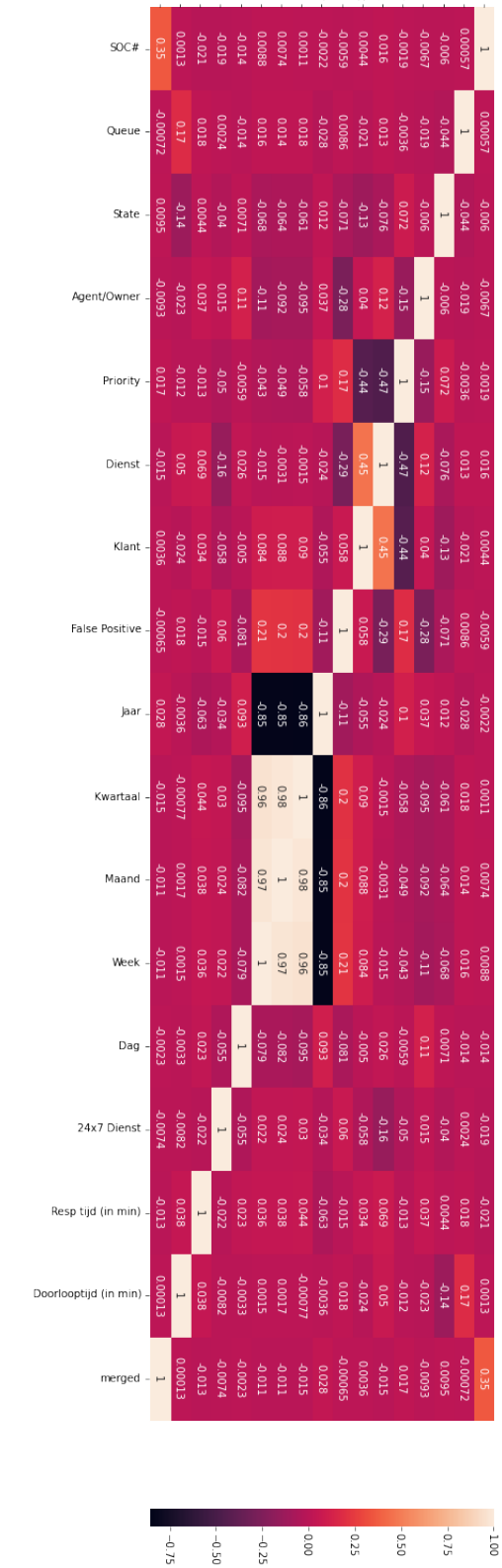

Figure A.1: Heatmap of correlating features

B

Sliding windows and cross-validation parameters

We use several folds and sliding windows and predict the label *State* and the label *False Positive*. The folds that we use are 3,5,10,15. The sliding windows we use are 1,2,3. We can see that a sliding window of 1 is performing slightly better than larger sliding windows. This is the case for both features and all different folds. The number of folds that we will use is 10 because they have a better prediction when we look at the results overall.

B.1. State

Table B.1: Parameter tuning folds and sliding windows - label *State*

Folds	Sliding windows	Accuracy	AUC	Precision	Recall
3	1	0.81	0.83	0.82	0.96
3	2	0.80	0.82	0.81	0.96
3	3	0.77	0.8	0.79	0.95
5	1	0.80	0.82	0.80	0.97
5	2	0.79	0.83	0.79	0.97
5	3	0.78	0.80	0.78	0.98
10	**1**	**0.82**	**0.84**	**0.83**	**0.95**
10	2	0.79	0.82	0.81	0.94
10	3	0.79	0.81	0.80	0.96
15	1	0.81	0.84	0.83	0.94
15	2	0.80	0.83	0.82	0.95
15	3	0.79	0.82	0.81	0.95

B.2. False Positive

Table B.2: Parameter tuning folds and sliding windows - label \textit{False Positive}

Folds	Sliding windows	Accuracy	AUC	Precision	Recall
3	1	0.88	0.88	0.00	0.00
3	2	0.89	0.89	0.00	0.00
3	3	0.88	0.88	0.00	0.00
5	1	0.89	0.83	0.13	0.01
5	2	0.89	0.84	0.02	0.00
5	3	0.89	0.83	0.16	0.00
10	**1**	**0.90**	**0.83**	**0.28**	**0.04**
10	2	0.89	0.83	0.18	0.05
10	3	0.90	0.83	0.2	0.04
15	1	0.90	0.75	0.16	0.03
15	2	0.90	0.79	0.08	0.03
15	3	0.89	0.78	0.12	0.03

C

Events and context example

C.1. Explanation of context vector and event

For an explanation of the DeepCASE results, we use two context events and two events. The array next to the event is the event of the incident that needs investigation. The context events are the context of the actual event and have happened before the event. The results can be read in such a way that the first event in the event array matches the first event in the context event array. Then the second event in the event array matches the second event in the context event array.

```
Event: [40,40]

Context event: ([[163, 163, 163, 163, 163, 103,  39,  46,  38,  40],
[163, 163, 163, 163, 163,  70,  62,  42,  42,  40],
```

C.2. Result DeepCASE - confidence threshold 0.2

Below we see the cluster of the DeepCASE model that is used at a confidence threshold of 0.2. The events are all the same and for the context, the events are the same as well. For the context event holds here that '40' is in every context event the even with the highest attention. We can derive this because '40' is the event that is present in every context event.

```
Event:
[40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,
40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,
40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,
40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,
40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40]
Context event:
[163, 163, 163, 163, 163, 103,  39,  46,  38,  40]
[163, 163, 163, 163, 163,  70,  62,  42,  42,  40]
[163, 163, 163, 163, 163, 163, 163,  91,  37,  40]
[163, 163, 163, 163, 163, 163, 163,  89,  88,  40]
[ 64,  64, 106,  86,  38,  42,  41,  56,  40,  38]
[163,  47,  42,  39, 103,  91,  41,  38,  40,  42]
[163, 163, 163, 163, 163, 163, 163, 163, 163,  40]
[163,  64,  83,  83, 103,  43,  49, 106,  80,  40]
[ 89,  88,  40,  42,  62,  89,  88,  40,  40,  40]
[163,  89,  88,  40,  42,  62,  89,  88,  40,  40]
[163, 163,  89,  88,  40,  42,  62,  89,  88,  40]
[163, 163, 163,  89,  88,  40,  42,  62,  89,  88]
[ 80,  91,  83,  83,  68,  68,  76,  93,  42,  40]
[163, 163,  76,  42,  64,  40,  94,  41,  38,  79]
```

```
[163, 163, 163, 163, 163, 163,  58,  41,  40,  38]
[163, 163, 163, 163, 163, 103,  42,  62,  42,  40]
[163, 163, 163, 163, 163,  50,  64,  40,  42,  62]
[163, 163, 163, 163,  50,  42,  40,  40,  40,  37]
[163, 163, 163, 163, 163, 163,  50,  42,  40,  40]
[163, 163, 163, 163, 163, 163, 163,  50,  42,  40]
[163, 163, 163, 163,  43,  95,  42,  40,  42,  62]
[163, 163, 163,  64,  63,  43,  83,  83, 103,  40]
[ 45,  39,  42,  47, 103,  64,  91,  40,  38,  42]
[ 42,  41,  79,  38,  41,  64,  80,  71,  71,  40]
[ 40,  42,  41,  79,  38,  41,  64,  80,  71,  71]
[163, 163, 163, 163, 163, 163,  79,  43,  50,  40]
[ 40,  42,  62,  42, 106,  76,  79,  43,  37,  40]
[ 40,  40,  42,  62,  42, 106,  76,  79,  43,  37]
[ 64,  76,  71,  93,  76,  42,  38, 103,  41,  40]
[ 94,  49,  39,  45,  42,  46,  39,  47,  40,  38]
[163, 163, 163, 163, 163, 163, 163,  71,  43,  40]
[163,  16,  11,  76,  70,  47,  42,  39,  42,  40]
[ 94,  94,  46,  68,  62,  42,  40,  42,  40,  75]
[ 45,  71,  94,  94,  46,  68,  62,  42,  40,  42]
[ 46,  42,  39,  93,  76,  71,  94,  42,  38,  40]
[ 42,  63,  41,  42,  40,  38,  40,  91,  64,  40]
[ 40,  42,  63,  41,  42,  40,  38,  40,  91,  64]
[ 42,  83,  83,  45, 106,  71,  93,  68,  95,  40]
[ 40,  42,  83,  83,  45, 106,  71,  93,  68,  95]
[ 40,  42,  62,  40,  42,  63,  41,  42,  40,  38]
[106,  76,  40,  42,  62,  40,  42,  63,  41,  42]
[163,  39,  42,  45,  43, 106,  76,  40,  42,  62]
[ 40,  39,  40,  46,  42, 103,  43,  40,  42,  41]
[163, 163, 163,  40,  39,  40,  46,  42, 103,  43]
[163, 163, 163, 163, 163,  71,  71,  41,  40,  38]
[163, 163, 163, 163, 163, 163, 163, 163,  40,  39]
[163, 108, 104, 107,  39,  71,  42,  41,  40,  38]
[ 42,  40,  42,  62,  81,  95,  91,  62,  38,  40]
[ 63,  42,  40,  42,  62,  81,  95,  91,  62,  38]
[163, 163, 163, 163, 163,  83,  83,  80,  93,  40]
[163,  40,  42,  40,  62,  42,  50,  93,  75,  40]
[163, 163,  40,  42,  40,  62,  42,  50,  93,  75]
[163, 163, 163, 163, 163, 163, 163, 163,  37,  40]
[163, 163, 163, 163, 163, 163, 163, 163,  40,  42]
[163, 163, 163, 163, 163, 163,  70,  38,  40,  41]
[163, 163, 163,  79,  51,  83,  83,  41,  42,  40]
[163, 163,  71,  71,  38,  95,  93,  41,  42,  40]
[163, 163, 163, 107,  39,  42,  45,  89,  88,  40]
[103,  30, 103,  80,  87,  75,  42,  62,  42,  40]
[163, 163, 163, 163,  75,  66, 104,  40,  30,  62]
[163, 163, 163, 163, 163, 163,  42,  38,  68,  40]
[163, 163, 163, 163, 163, 163, 163,  91,  91,  40]
[ 87,  50,  50,  50,  50,  50,  62,  42,  40,  42]
[163, 163, 163, 163, 163, 163, 163,  50, 106,  40]
[163, 163, 163, 163, 163, 163, 106,  50,  42,  40]
[ 75,  68,  68,  89,  88,  94,  50,  49, 106,  40]
[163, 163, 163, 163, 163, 163,  68,  68,  40,  42]
[163, 163, 163, 163, 163, 163, 163, 163, 163,  40]
[163, 163, 163, 163, 163, 163, 108, 104,  62,  40]
[ 40,  62, 104, 108, 104,  80, 108,  62,  40,  39]
```

```
[ 39,  40,  40,  62, 104, 108, 104,  80, 108,  62]
[163, 163, 163, 163, 163, 163, 163, 163,  39,  40]
[163, 163, 163, 163, 163,  62,  40,  42, 103,  42]
[163, 163,  62,  40, 103,  40,  75,  31,  62,  40]
[163, 163, 163,  62,  40, 103,  40,  75,  31,  62]
[163, 163, 163, 163, 163, 163, 163,  62,  40, 103]
[163, 163, 163, 163, 163,  26,  39, 104,  40,  62]
[163, 163, 163, 163, 163, 163, 103, 104,  39,  40]
[163, 163, 163, 163, 163, 163,  43,  40,  74,  16]
[163, 163, 163, 163, 163, 163, 163, 163,  40,  40]
[163, 163, 163, 163, 163, 163, 163, 163, 163,  40]
[163, 163, 163, 163, 163, 163, 163, 163,  40,  40]
[163, 163, 163, 163, 163, 163, 163, 163, 163,  40]
[  9,  40,  40,  40,  62,  37,  40,  40, 108, 108]
[163, 163, 163,   9,  40,  40,  40,  62,  37,  40]
[163, 163, 163, 163,   9,  40,  40,  40,  62,  37]
[163, 163, 163, 163, 163, 163, 163,   9,  40,  40]
[163, 163, 163, 163, 163, 163, 163, 163,   9,  40]
```

D

Analysis of the clusters

In Section 7.2.7, we show the analyst examples of the clusters. In this Appendix we explain the clusters that are sent to the security analyst. We take the example *0.05_cluster_14_number_1* and show how the name is built up. 0.05 is the value of the confidence threshold. Then we have the number of the cluster, which is 14. Then we have the times that the algorithm has run which is in this case 1. Do note that the cluster number is the number in the run, not in the whole run. So, there could exist a cluster that is *0.05_cluster_14_number_2*. We show the unique titles of the clusters and show what the analysts had to say about these clusters. He analysed the clusters together with the Source IP and Destination IP. The clusters had two types of events, the context event and the event itself. We did not have information on what the context event is and what the event is but we can still analyse if the titles should be together.

D.1. 0.05_cluster_14_number_1

These clusters were correctly classified, as in the data was seen that the Source IPs and Destination IPs were all similar. These come all from the same scan.

- Cisco ASA directory traversal attempt
- Citrix ADC and Gateway arbitrary code execution attempt

D.2. 0.05_cluster_24_number_1

These were scans as well, coming from the same Source IP and going to the same Destination IP.

- JBoss JMXInvokerServlet access attempt
- SQL 1 = 1 - possible sql injection attempt

D.3. 0.06_cluster_3_number_7

The Apache Struts class access attempt was not correlating with the SQL injection attempt. This was not a logical cluster.

- SQL 1 = 1 - possible sql injection attempt
- Apache Struts java.lang.ProcessBuilder class access attempt

D.4. 0.07_cluster_8_number_1

This was a logical cluster.

- Apache Struts remote code execution attempt
- Apache Struts OGNL getRuntime.exec static method access attempt

D.5. 0.08_cluster_5_number_1

This was a logical cluster.

- Apache Struts remote code execution attempt
- Apache Struts2 blacklisted method redirect

D.6. 0.08_cluster_13_number_1

The Source IPs were different and did not match as well as the alerts did not seem to correlate. The analyst did not know why these were clustered.

- Citrix ADC and Gateway arbitrary code execution attempt
- Apache Tomcat FileStore directory traversal attempt

D.7. 0.08_cluster_15_number_1

This was a logical cluster.

- Citrix ADC and Gateway arbitrary code execution attempt
- JBoss JMXInvokerServlet access attempt