# Automated data exfiltration detection using netflow meta-data

Takang Kajikaw Etta Tabe

TU Delft

# Automated data exfiltration detection using netflow metadata

by

**Takang Kajikaw Etta Tabe**

to obtain the degree of Master of Science
Data Science  Technology track
Cyber Security specialization

at the Delft University of Technology,
Faculty of Electrical Engineering, Mathematics and Computer Science,
to be defended publicly on Thursday September 19, 2019 at 1:30 PM.

| | | |
|---|---|---|
| Student number: | 4373758 | |
| Project duration: | January 3, 2019 — September 3, 2019 | |
| Thesis committee: | Dr. ir. J. C. A. van der Lubbe, | EEMCS, TU Delft |
| | Dr. Ir. S. E. Verwer, | TU Delft, Supervisor |
| | Peter Cooper, | Adyen, Supervisor |
| | Dr. A. Panichella, | SERG, TU Delft |

TU Delft

# Abstract

*The volume and sophistication of data exfiltration attacks over networks have significantly increased in the last decade. This has resulted in the need for defense mechanisms, to effectively detect both known and unknown data exfiltration scenarios over the network. While methods such as DPI (Deep Packet Inspection) are commonly used to detect data exfiltrations, this mechanism requires a thorough inspection of every payload or packet going out of the network, making it unsuitable for use in some environments, as it is quite resource intensive and can lead to severe data privacy implications. In our work, we use lightweight netflows which are non-privacy invasive to detect data exfiltrations at connection-level granularity. The key intuition behind our proposed solution is that connections involved in data exfiltration tend to differentiate themselves from normal network connections based on certain feature values. The result of this research shows that features extracted from netflows such as the duration of a netflow, the source bytes, the source bytes sent per second, the source bytes sent per packet and the producer-consumer ratio can be used to effectively detect data exfiltration. Subsequently, connections are grouped using k-means, and the robust Z-score of their distances from their respective cluster centroid is used as a statistical and distance-based technique to detect connections involved in a data exfiltration. While this method detects some data exfiltration scenarios, it results in a significant number of false positives. Combining this with the results from the LOF (local outlier factor) and the LoOP (local outlier probability), which are density-based techniques, leads to a more robust model, as it significantly reduces the number of false positives and false negatives. Also, we show that using the smallest clusters formed from k-means for analysis leads to similar detection results as the entire datasets, with a significant reduction in computation time.*

# Preface

Being born and growing up as a kid in one of the poorest, but happiest places in the world, I had never, even in the wildest of my dreams, ever thought that I would one day be embarking on such a glorious journey; a journey of more than a thousand miles, that started with a little less than one step. Initially, I set on this journey without a clear plan. It was a maze with interesting and humbling lessons and experiences at every corner or bend. Like a new born baby learning to walk, I started by crawling, then I took the leap of faith and made my first staggering steps, of course with the help and guidance of God, my family, friends and colleagues, to whom I owe a debt of thanks.

I would like to start by enormously thanking my parents, especially my ever loving, sweet, adorable, supportive and caring mother, who worked multiple odd jobs just to make ends meet and stood by my side throughout this journey. Without her, I would definitely not be where I am today. Thanks a lot momma bear (Madam Sule). To my dad, thank you for everything and the manly advice pops. Also, thank you Wlodek for the moral support throughout this crazy journey. You saw it start and I am glad that you are witnessing the finish. To "tante Jacqueline" and E. Schijff, thank you. To my little brothers Jason and John, thank you guys for giving me so much responsibility and for encouraging me on the days when I was down.

I would like to thank my supervisors: Peter Cooper for persistently getting me out of the rabbit hole in several occasions, for the academic, moral and intellectual guidance and for teaching me how to critically analyze problems. The critical feedback and constant support were of tremendous help. Also, thank you for giving me an awesome spot on such a cool team and coming up with such a cool project. I am and will be forever grateful. Thank you to Sicco Verwer, for his academic guidance and constant support throughout the duration of this project. The weekly meetings, the constructive feedback were the molding blocks in the development of this product.

Special thanks to Huub van der Voort, dr. Julienka Mollee, Suze Derkse, Ignacio Jimenez Pi, Paul Moreno, Alejandro Ferrer Delgado, Albert Tresens, Thibaut Zonca, David Meijers and Otto Heinen for giving me a space on such a cool team and for your listening ears to my daily cries. Thank you guys for your support, the daily feedback, your complete engagement, the challenging questions and for giving this project the value it has. Big thanks for making time out of your busy schedules to attend my numerous, poorly structured and sometimes boring presentations. To Huub van der Voort (again), thank you immensely for the time you spent not only on datasets, but also on the review of the entire report, your feedback on every aspect of this project, the constant check-ins and for the time you spent on my sometimes irrelevant questions.

It is commonly said, that the way to a man's heart is through his stomach. With this, I send out immense thanks from the bottom of my heart, to the awesome ladies and gentlemen of the Adyen catering team for the beyond great, delicious lunch and for all the love and care.

To dr.ir. J.C.A van der Lubbe, dr. Annibale Panichella and everyone else who helped me on this journey, I say thank you so much. You are truely appreciated.

*Takang Kajikaw Etta Tabe*
*Delft, September 2019*

# Contents

# List of Figures

# List of Tables

# 1 Introduction

The increasing amount of sensitive information collected and stored by companies, combined with an increase in the value of stolen information [97] has led to a sharp rise in their data exfiltration attack surface over the last decade[86]. Data exfiltration, also referred to as data leakage, data loss or a data breach in this report, involves the unauthorized movement of sensitive information from an internal company or organizational network scope to an external scope. According to the IBM security and Ponemon Institute, not only has there been a 21% increase in the number of data breaches in the past six years, but the average total cost of a data breach in the last two years for companies worldwide has risen from $3.6 million to almost $4 million[86]. The possibility of a leak of sensitive or confidential information ranks as one of the biggest fears to organizations, their employees and customers, requiring an immediate call to action[66]. This is because very often the consequences are dire, ranging from reputation loss to monetary fines, legal sanctions and ramifications including lawsuits with enormous impact on business continuity.

Data exfiltrations within an organization can be carried out by both insiders and outsiders. Insiders usually have legitimate network and system access, and can make use of this legitimate access to inadvertently breach security and exfiltrate information. Outsiders, on the other hand, have no legitimately authorized network or system access and need to make use of a compromised internal system in order to exfiltrate information. Irrespective of the actor, the data to be exfiltrated will always be placed in an external outbound payload, providing us with the capability of detecting data exfiltrations carried out by both internal and external actors. This narrows down our research scope to external outbound connections exclusively.

Major data breaches which occurred in 2018 demonstrate the inevitability and the possibility of a leak of sensitive, confidential and private information from organizations in different business sectors. Some of these data breaches obtained from PandaLabs [76] include:

- The Marriott hotel chain data breach which leaked the private and personal data of about 500 million customers.
- The Aadhaar (India's national ID database) breach potentially exposing the information and biometric identification data of 1.1 billion Indian citizens.
- The Exactis data breach of June 2018 which leaked information containing names, home addresses, phone numbers, mortgage information and religion of about 340 million individuals on a publicly available server.
- The MyFitnessPal data leakage of March 2018 in which data of about 150 million users had been leaked due to a compromise of the app and website.
- The Panera bread breach in April 2018 which leaked up to about 37 million customer records which included names, email and home addresses.
- The October 2018 data breach just a few weeks before the November elections in the US which left about 35 million voter records up for sale on a hacking website.

The fact that data exfiltrations have enormous impact to organizations, combined with the increasing frequency of occurrence, makes the detection of this attack important for organizations, irrespective of the sector in which they operate.

## 1.1. Industry Partner

Adyen is a PSP (payment service provider) that focuses on the processing of payments for companies all over the globe, while also providing other products and services, such as fraud detection. Adyen's payment landscape can mainly be divided into three channels: M-commerce (Mobile commerce), E-commerce (Electronic commerce) and POS (Point of Sale).

Adyen collects, stores and processes substantial flows of transaction data in order to process payments for merchants. A part of this payment infrastructure contains highly sensitive, confidential and protected merchant and shopper information, handled and sometimes analyzed by Adyen employees. This makes them potential targets for external attackers looking for systems to steal valuable data from. Furthermore, disgruntled or bribed employees with sufficient infrastructural knowledge or excessive access privileges with an incentive of stealing or aiding in a data exfiltration is an imminent risk. Similarly, internal employees are also susceptible to phishing attacks [32], resulting to a data exfiltration malware being just one mouse click away. The aforementioned problems pose a serious threat to companies, their employees and customers. This in turn leads to the need for mechanisms to protect against data exfiltration. The chief outcome of such a mechanism is a system which provides a clear overview of the movement of data in and out of the organization's network.

In order to have some control of the network traffic coming in and going out of the network, Adyen collects and logs real-time network traffic metadata from its internal network. This meta data is then stored in an elasticsearch cluster. The Elasticsearch cluster provides easy extraction of data for analyses and is an important component of the network traffic logging and routing process as shown in Figure 1.1.

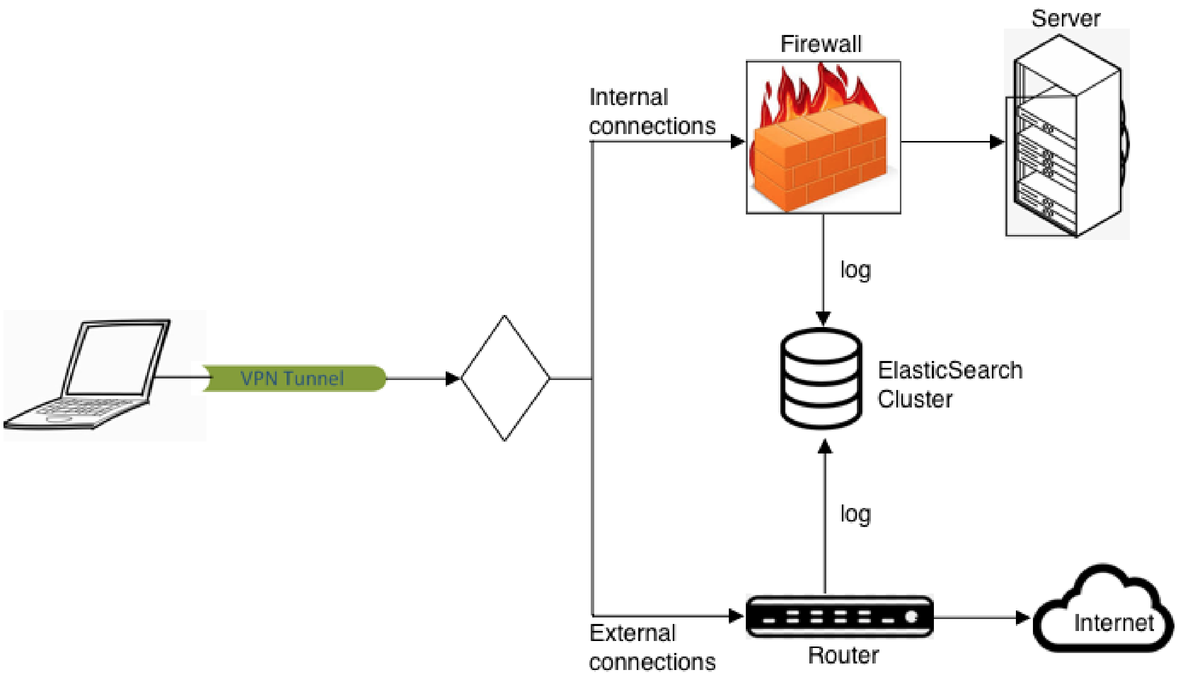


Figure 1.1: Simple graphical overview of network traffic routing and logging done at Adyen

Figure 1.1 shows the simplified network routing and logging process at Adyen. From the figure, we see that all traffic from an Adyen device is routed through a VPN tunnel. From here onwards, a routing decision is made. All internal connections go through firewalls before an access is made to the Adyen servers. Outbound external connections, on the other hand, are routed through proxy servers to the internet. A key event common to both internal as external connections is the logging of network traffic metadata to the elasticsearch cluster. This occurs for outbound internal connections at the firewall level and at the router level for outbound external connections. An advantage of this is that it gives valuable high level information about the connections going in and out of the network.

## 1.2. Problem Description

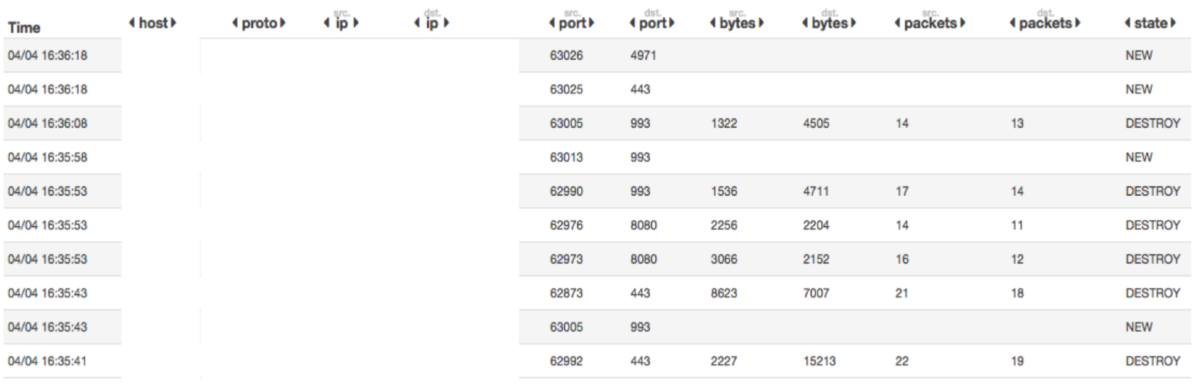
In order to detect and prevent data exfiltration over networks, deep packet inspection techniques are commonly used. They require the inspection of every packet going out of the network, making them very resource intensive and sometimes breaching privacy laws, depending on the type of data being sent out. We, therefore, research on the use of network metadata, which are non privacy-invasive to detect data exfiltration.

The main goal of this research is to build a system that effectively detects data exfiltration using network metadata. However, it is important to mention that this system has to work in a large-scale environment like Adyen and handle the large influx of new data entering the network every second. Also, important to note is that the detection of data exfiltrations over networks comes with some challenges. Firstly, the seldom occurrence of data exfiltrations, combined with the possibility of these events going on for long periods of time could lead to an increase in the number of false positive and false negative events [11]. Additionally, the fact that data exfiltrations could be hidden in a small single network connection among other large network connections can make them difficult to detect [71, 92]. Also, more limitations are faced when trying to detect network anomalies from real-time systems [35] rather than from publicly available datasets as used in most research. One of these limitations is the computational cost, especially for distance-based outlier or anomaly detection mechanisms, where the need for the calculation of distances for new datapoints can be overwhelming for high volume data [35].

We are, therefore, faced with the problem of **finding an automated data exfiltration detection solution based on non-privacy invasive lightweight network metadata**. Note that the term *lightweight* is used. This originates from the fact that no in-depth overview of the communication is provided but just basic information of the network connections made.

## 1.3. Dataset

For the sake of network traffic monitoring, Adyen collects and stores real-time NetFlow metadata. Netflow was originally a packet switching system designed by Cisco in 1996 [52]. It is designed to collect IP traffic information, providing a clear overview of the source and destinations of the network traffic connections, and how much traffic is being generated between these hosts. A sample of NetFlow data captured at Adyen is shown in Figure 1.2. In this research, we further refer to singular NetFlow instances as netflow and multiple instances as netflows.



| Time | host | proto | src. ip | dst. ip | src. port | dst. port | src. bytes | dst. bytes | src. packets | dst. packets | state |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 04/04 16:36:18 | | | | | 63026 | 4971 | | | | | NEW |
| 04/04 16:36:18 | | | | | 63025 | 443 | | | | | NEW |
| 04/04 16:36:08 | | | | | 63005 | 993 | 1322 | 4505 | 14 | 13 | DESTROY |
| 04/04 16:35:58 | | | | | 63013 | 993 | | | | | NEW |
| 04/04 16:35:53 | | | | | 62990 | 993 | 1536 | 4711 | 17 | 14 | DESTROY |
| 04/04 16:35:53 | | | | | 62976 | 8080 | 2256 | 2204 | 14 | 11 | DESTROY |
| 04/04 16:35:53 | | | | | 62973 | 8080 | 3066 | 2152 | 16 | 12 | DESTROY |
| 04/04 16:35:43 | | | | | 62873 | 443 | 8623 | 7007 | 21 | 18 | DESTROY |
| 04/04 16:35:43 | | | | | 63005 | 993 | | | | | NEW |
| 04/04 16:35:41 | | | | | 62992 | 443 | 2227 | 15213 | 22 | 19 | DESTROY |

Figure 1.2: Sample of Netflow data kept and stored by Adyen. The **Time** field contains the date and time when that specific netflow took place. The **host** field pertains to the firewall where the flow was collected. Followed is the **src.ip** field which is the source IP, corresponding to the IP address of the host initiating a communication. The **dst.ip** corresponding to the destination IP is the IP address of the destination host of that particular connection. The **proto** is the protocol used, which could either be TCP, UDP or ICMP. The **src.port / dst.port** tuple corresponds to the ports used at the source and destination of the communication. The **src.bytes / dst.bytes** tuple correspond to the total number of bytes sent from the source and destination respectively. Furthermore, the **src.packets / dst.packets** tuples correspond to the number of packets used to send the information from the source to destination and destination back to source respectively. The field **state** corresponds to the state of the communication. Ongoing communications are labeled with state NEW while terminated communications are labeled with state DESTROY. The **host**, **proto**, **src.ip** and **dst.ip** fields are redacted due to the sensitivity and confidentiality of the information.

## 1.4. Proposed Solution and System Components

In order to achieve our aforementioned goal, a proof of concept is designed and implemented to detect data exfiltration in outbound external connections using netflows. The methodology is divided into five main components as shown in Figure 1.3. In the first component, the extracted netflows are pre-processed and prepared for processing. The second component extracts useful features from the netflow metadata which can be used to detect data exfiltration. We then normalize these feature values in the third component, after which, we group our connections based on the similarity of their feature values in the fourth component. The fifth component allocates anomaly scores to connections. To grasp a better comprehension of these phases, we first of all explain our original dataset.

The key intuition behind our proposed solution is that data exfiltrations over the network tend to differentiate themselves from other normal network connections based on certain feature values. Also, if real-time network traffic changes significantly at different times, we have to adopt a solution which in its entirety is also not static and takes this network change into account. Figure 1.3 shows a high level overview of the components in our system. The role of each system component is explained below.

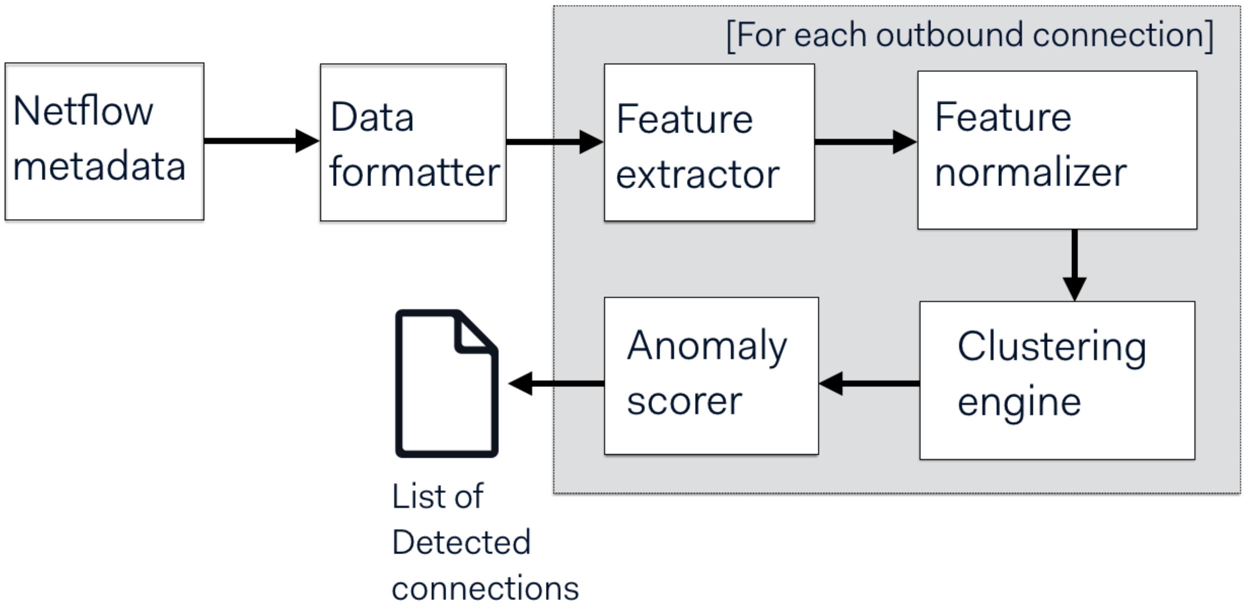


Figure 1.3: System decomposition flow chart of our proposed solution.

### 1.4.1. Netflow metadata

This component represents the elasticsearch clusters on which network traffic metadata are collected and stored.

### 1.4.2. Data formatter

This component extracts the metadata from the netflow collector, prepares and formats as pre-processing. In this component, we select fields from the overall netflow information which are useful in the feature extraction component. Any fields not needed in the feature extraction component are, therefore, discarded. The output of this component are all connections in the network and their corresponding relevant information.

### 1.4.3. Feature extractor

This component is responsible for the extraction of a set of features (selected from recent literature) which we need to detect data exfiltration on a connection-level granularity. The output from the data formatter is used as input in this component. The output of this component is a dataframe which contains the analyzed connections and their corresponding feature values. An overview of the chosen set of features and their motivations is discussed in Chapter 3. Not all features from the netflow collector are used in this research. The discarded features and the motivations why they are discarded are provided in Section 3.3.

### 1.4.4. Feature normalizer

The features extracted in order to detect data exfiltration are observed to all have a different heavy tail distribution. As a result of this, normalization of these feature values is needed in order to do any comparisons between connections. This is explained and further discussed in more detail in Section 3.4. This component returns another dataframe containing normalized feature values for all connections.

### 1.4.5. Clustering engine

In this component, we process the dataframe from the previous component and create clusters based on our extracted feature set per connection. Based on certain criteria provided in Chapter 5, connections are clustered using k-means clustering algorithm. This component returns a dataframe containing connections clustered into different groups with the calculated euclidean distance from their cluster centroid. This component is presented in Chapter 5.

### 1.4.6. Anomaly Scorer

This component allocates anomaly scores to connections. It uses a distance, statistic and density-based approach to allocate these anomaly scores to connections. The distance and statistic based scores are allocated per connection based on the robust Z-score of the distance from its cluster centroid as explained in Section 6.1. The density based approach uses the LOF (local outlier factor) explained in Section 6.2 and LoOP (local outlier probability) explained in Section 6.3 to score connections. Each connection gets three main anomaly scores allocated to it, which are combined together by averaging to give a total of seven anomaly scores per connection. These seven anomaly scores are individually normalized within the range [0,1] and scaled by 100 such that they are within the range of [0, 100] with 100 being a connection likely to be an exfiltration and 0 being a normal connection. More details about this component are provided in Chapter 6. Based on the anomaly scores from the anomaly scoring engine, a threshold is chosen above which connections are considered to be anomalous and below which they are considered to be normal. The output is then a list of detected connections with an anomaly score greater than the threshold.

## 1.5. Research Question

This thesis presents an exploratory study on the use of netflow metadata for the detection of data exfiltrations over networks. In order to achieve this, a classical unsupervised anomaly detection method is used which starts with the extraction of important features, with features relating to specific host traffic content [14, 30]. Once features have been selected, a clustering algorithm is applied to group network connections based on the selected features. A distance, statistical and density based approach is then used to allocate anomaly scores to connections, which are then used to detect connections involved in data exfiltrations. With this said, we can now formulate the research question that we answer in this paper.

**Research Question: Is the use of NetFlow metadata effective in detecting data exfiltration over networks in a large-scale industrial context?**

In order to answer the research question, we split the main research question into smaller sub research questions. The approach used to answer the research question clusters connections based on features which can be used to detect data exfiltration over networks, which leads us to our first sub research question.

**Sub-Question 1: What features extracted from the available NetFlow metadata can we use to detect data exfiltration?**

Once we have a selection of features which we deem necessary and appropriate for our use case, we then have to think about an appropriate clustering method which can be used to cluster connections such that data exfiltration over networks can be detected. It is worth mentioning that various clustering algorithms will perform differently on our data, which brings us at our next sub research question:

**Sub-Question 2: What clustering technique is appropriate to use for the detection of data exfiltration using NetFlow metadata?**

Depending on the clustering algorithm we choose to use, clusters will be formed differently. From the numerous clustering algorithms that exist in literature, it is important to find a suitable clustering algorithm which can be used to cluster connections established and logged on the netflow collector. The importance of making the right choice lies in the fact that we should be able to understand how various connections are clustered and also explain the characteristics of the different clusters. Due to the large influx of the input data that needs to be processed and clustered, it is of uttermost importance that the clustering method be consistent, scalable, enabling it to handle large data inputs. The clustering algorithm should also produce clusters which are self explanatory to domain experts. Also, it is important that the clustering algorithm has minimal tweakable parameters, so we can exhaustively explore the parameter space for optimal solutions. Testing all available clustering algorithms is unfeasible, so we have to choose a small subset of clustering algorithms from the larger set, keeping our solution requirements in mind.

**Sub-Question 3: What technique can we use to detect netflow traffic connections involved in data exfiltration based on the results from the clustering algorithm used in sub-question 2?**

Once our netflow connections are clustered, we need to find an appropriate distance-based technique to detect connections which are involved in data exfiltration. In this case, a score in the range of [0, 100] pertaining to how anomalous a connection is should be allocated with scores closer to 0 meaning the connection is more likely to be benign and scores closer to 100 meaning a connection is more likely to be involved in an exfiltration. Using multiple techniques other than a distance-based technique to detect data exfiltration might get rid of the biases that come into play when using just a single technique, leading us to our next sub research question.

**Sub-Question 4: What other techniques independent from the technique in sub-question 3 can we use to allocate anomaly scores to netflow connections and how do we combine them?**

When other techniques are found and combined with the distance-based approach, connections end up with seven anomaly scores.

**Sub-Question 5: What anomaly scoring metric performs best in the detection of simulated data exfiltration scenarios and at what threshold does this perform best?**
Combining the different techniques above leads to seven anomaly scoring metric. We then identify which of the anomaly scoring metrics performs best in terms of recall, precision, fpr and fnr. We also determine the threshold at which it performs best.

## 1.6. Research scope

In this research, netflow metadata is used to automatically detect data exfiltrations over networks. We, therefore, assume that the netflows have already been collected. As such, netflow collection and storage is out of the scope of this research. Also, this research focuses on the detection of data exfiltrations over networks. Data exfiltrations not going over the network are, therefore, out of the scope of this project.

In addition, the scope of this project covers only connection-level granularity. Although detection is done on a connection-level granularity, our proposed solution can easily be adapted to host-level granularity. The appropriate feature-set for detection at a host-level have been implemented and are presented in Section 3.2. The complete implementation on a host-level granularity has been left as future work.

## 1.7. Major research contributions

Most techniques used to detect data exfiltrations going over networks increasing rely on Deep Packet Inspection. The nature of this technique which requires a thorough inspection of every payload going out of the network, makes it unsuitable for use in several environments. This is predominantly because of its privacy implications, combined with the fact that analyzing every single payload going out of a network is quite resource-intensive. Our proposed solution is novel as it makes use of lightweight netflow meta data, commonly collected by most companies and which provides only the high-level characteristics of every connection, making it non privacy-invasive. The work from this research led to some major contributions which we list below:

- **Feature-set selection:** we identify the right feature-set from netflow metadata which can be used to detect data exfiltrations over networks at connection-level granularity.
- **Clustering algorithm:** We identify what clustering algorithm is suitable for grouping network connections with the main aim of detecting connections involved in data exfiltration.
- **Distance and statistical based anomaly detection technique:** We identify an appropriate distance and statistical based technique which can be used to detect connections involved in data exfiltration based on the distance from their respective cluster centroids.
- **Density-based anomaly detection technique:** We identify an appropriate density-based technique for the detection of connections involved in data exfiltration based on the relative density of the area in which connections find themselves in the feature space.
- **Combination of distance, statistical and density based approaches:** Recent works in the field of network anomaly detection either make use of a distance, density or statistical based approach to detect anomalies. While these methods seem to work well individually, they have their biases. In order to eliminate the bias of a single approach, we combine the three approaches for better detection results in terms of recall, precision. FPR (False Positive Rate) and FNR (False Negative Rate).
- **Computational time speed up:** Initially, our proposed solution took hours to run on the entire test datasets used in this research. In order to speed up its computational time performance, we perform our analysis on the smallest cluster of every dataset. This reduced the time to run from hours to seconds with the very same detection abilities.

## 1.8. Results summary

The results of this research shows us that simple netflow features such as the duration of the netflow, the source bytes, the source bytes per packet, the source bytes per second and the ratio of bytes sent to bytes received can be used to detect data exfiltrations. Also, we found that combining a distance, statistical and density based anomaly detection approach leads to better detection results. The statistical and distance-based approach at low thresholds leads to high recall value as it detects a very high proportion of true positive events but also results in a very low precision value due to the high number of false positives. The density-based approach on the other hand leads to high precision scores due to its very small number of false positives but to low recall due to a few false negatives. Linearly combining these approaches together leads to better detection results as it leads to a lower number of false positives and false negatives, compared to when these approaches are used single-handedly.

## 1.9. Report outline

This report documents an exploratory study on the use of netflow metadata for the automated detection of data exfiltrations over the network. Chapter 2 provides elaborate background knowledge on the mathematical concepts used in this report and also provides some recent literature on network traffic-based anomaly and outlier detection. Each component of our proposed solution is then presented in the chapters that follow. Chapter 3 provides and motivates the choice of features used in this research to detect data exfiltration and also the set of discarded features and the motivations why they are discarded. This chapter provides an answer to **sub research question 1**. We also further explore the training dataset for trends and correlations on the features selected in Chapter 3. In Chapter 4, we explain the data collection stage of the various datasets used in this research, explore the various test datasets and explain the simulated data exfiltration scenarios they contain. Chapter 5 explains the criteria used to select the clustering algorithm used in this research, provides and explains the results obtained from the clustering algorithm on our different test datasets. This chapter answers **sub research questions 2 and 3** as it also explains how anomalous connections can be detected from the clustered set. We propose three main metrics used to allocate anomaly scores to the connections in our various test sets, which are then combined together by averaging as explained in Chapter 6.

This chapter answers **sub research questions 3 and 4**. The results obtained from our built proof of concept on the various test sets and the validation set are presented and discussed in Chapter 7. This chapter answers **sub research question 5**. The limitations of our work and areas open for further future work are discussed in Chapter 8. We finally conclude in Chapter 9.

# 2 State of the Art

This chapter provides background knowledge required to understand the rest of the report. Additionally, it presents some relevant and recent literature studies in the field of network traffic anomaly detection.

## 2.1. Background

The upcoming chapters will contain a few terms and concepts which will need to be explained in order to better understand this research work. An attempt to explain these terms and concepts will be made in this section.

### 2.1.1. Distance Metrics

First and foremost, it is quite important for us to define or describe what we mean by distance. Distance is a measure of the length of space between two or more points or objects. The distance between two points is the length of a straight line segment that connects them together and the distance of a point from a line is the length of the shortest line segment from the point to the line, perpendicular to it. In the field of machine learning, the distance metric helps algorithms to identify and recognize similarities and dissimilarities between objects. While there are several distance metrics, in this research we will cover 2 different distance metrics namely Euclidean distance, Mahalanobis distance. Even though the Minkowski and cosine metrics would not be used, we will mention them so others could experiment with them if deemed necessary.

#### Mahalanobis Distance

The Mahalanobis distance is a distance metric which computes the distance between two points from the same dataset containing a multivariate space. In other words, it measures the distances between a point Q and a distribution D. It, therefore, measures how many standard deviations point Q is away from the mean of the distribution D. In the regular case of a Euclidean, variables for example a, b, c, can be shown as 3 axes drawn at right angles to each other, allowing for the measurement of the distance between two points with a simple ruler. In contrast to Euclidean, for uncorrelated variables, the Euclidean distance is equal to the Mahalanobis distance. In the case of correlated variables, the right angularity of the axes is canceled, making measurement by a ruler impossible.

With the presence of more than 3 variables, plotting this on a 3D space becomes impossible. Since we will be dealing with a data set containing more than 3 variables, this is of importance. This problem is solved by Mahalanobis since it measures the distances between points (correlated points included) for multiple variables.

The formula for the Mahalanobis distance is provided below[95].

$$Mahalanobis\,Distance(r,s) = \sqrt{(r-s)M^{-1}(r-s)^T}$$

where: r and s are two points for which the Mahalanobis distance is to be computed
M is the inverse of the sample covariance matrix of the entire dataset.

**Example**

In order to have a representation of how the Mahalanobis distance is calculated, let's assume we have a dataset with n = 5 objects, with each object having a Length (A), Width (B), and Height (C) as shown in table 2.1. Since our dataset has 3 variables, it is 3-dimensional. Let us then calculate the distance between another object, o = (22.0, 20.0, 18.0) and our dataset of 5 objects.

| A | B | C |
|---|---|---|
| Length | Width | Height |
| 10.0 | 5.0 | 10.0 |
| 20.0 | 9.0 | 13.0 |
| 30.0 | 13.0 | 15.0 |
| 33.0 | 20.0 | 19.0 |
| 49.0 | 34.0 | 23.0 |

Table 2.1: Dataset of 5 objects with Length, Width and Height parameters

From the table above, we can calculate the Means per column or variable using the formula:

$$Mean(\mu) = \frac{1}{n} \cdot \sum_{i=1}^{n} X_i$$

where X is a value in the column and n is the total number of objects.

With the above formula, we get:

$$\begin{aligned} Mean(A) &= \frac{10.0 + 20.0 + 30.0 + 33.0 + 49.0}{5} = 28.4 \\ Mean(B) &= \frac{5.0 + 9.0 + 13.0 + 20.0 + 34.0}{5} = 16.2 \\ Mean(C) &= \frac{10.0 + 13.0 + 15.0 + 19.0 + 23.0}{5} = 16.0 \end{aligned}$$

The mean of the dataset is (28.4, 16.2, 16.0). In order to find out how far another object, o = (22.0, 20.0, 18.0), is from our dataset, we need to calculate the Mahalanobis distance.

From intuition, we could look at the distance between object o = (22.0, 20.0, 18.0) and the mean of our dataset (28.4, 16.2, 16.0). The Mahalanobis distance also takes into account how far the Length, Width and Height values are from each other.

To begin with, we calculate the difference between the object o and the mean of the dataset to get, k, :

$$k = (22.0, 20.0, 18.0) - (28.4, 16.2, 16.0) = (-6.4, -3.8, 2.0)$$

Using the formula provided above for the Mahalanobis distance, we need the inverse of the covariance matrix. In order to get this, we need to formulate the covariance matrix of our dataset. In the covariance matrix, the variances of the variables appear in the diagonals while the covariance appear in the off-diagonal sections as shown below for variables A, B and C from our example table above:

$$\begin{bmatrix} Var(A) & Cov(B,A) & Cov(C,A) \\ Cov(A,B) & Var(B) & Cov(C,B) \\ Cov(A,C) & Cov(B,C) & Var(C) \end{bmatrix}$$

The matrix above tells us that we need to calculate the variance of each variable in our dataset and the covariance between the objects from 2 different variables. The variance and covariance will be calculated using the formula:

$$Var(X) = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (X_i - \mu_x)^2$$

$$Cov(X,Y) = \frac{1}{n-1} \cdot \sum_{i=1}^{n-1} (X_i - \mu_x) \cdot (Y_i - \mu_y)$$

Notice that the formulas above uses (n-1) as denominator instead of n. This is because we make use of the sample variance for this calculation.

Calculating the variances gives us:

$$Var(A) = \frac{(10-28.4)^2 + (20-28.4)^2 + (30-28.4)^2 + (33-28.4)^2 + (49-28.4)^2}{5-1} = 214.3$$
$$Var(B) = \frac{(5-16.2)^2 + (9-16.2)^2 + (13-16.2)^2 + (20-16.2)^2 + (34-16.2)^2}{5-1} = 129.7$$
$$Var(C) = \frac{(10-16)^2 + (13-16)^2 + (15-16)^2 + (19-16)^2 + (231-16)^2}{5-1} = 26$$

Calculating the covariances gives us:

$$Cov(A,B) = 161.4$$
$$Cov(A,C) = 73$$
$$Cov(B,C) = 57$$

Due to the symmetric property of the covariance, we get:

$$Cov(X,Y) = Cov(Y,X)$$

With the above calculated values for Variance and covariance, we get the following covariance matrix:

$$\begin{bmatrix} 214.3 & 161.4 & 73 \\ 161.4 & 129.7 & 57 \\ 73 & 57 & 26 \end{bmatrix}$$

The inverse of the above covariance matrix is:

$$\begin{bmatrix} 0.11 & -0.03 & -0.24 \\ -0.03 & 0.22 & -0.39 \\ -0.24 & -0.39 & 1.58 \end{bmatrix}$$

Now, multiplying the 1x3 matrix, k = (-6.4, -3.8, 2.0 ), with the above inverse covariance matrix, we get the temporary 1x3 matrix:

p = (-1.07, -1.42, 6.18)

To get the squared Mahalanobis distance, we need to multiply p with the transpose of k, giving:

$$MD^2 = 24.604$$

with MD being the Mahalanobis Distance.

The last step entails taking the square root, resulting to a Mahalanobis distance of 4.96

Despite thefact that the Mahalanobis distance seems to be a very good distance feature, it requires the calculation of the inverse of the correlation matrix. This becomes infeasible If the variables are highly correlated with each other [95].

## Euclidean Distance

The Euclidean is one of the most commonly used distance metrics in the field of mathematics and geometry [40]. It is often considered the *ordinary straight line* distance. It is a 1, 2, 3 dimensional distance metric in which the distance between two points p and q in that dimensional space corresponds to the length of the straight line segment connecting both points.

For a 1,2,3 dimensional space, the Euclidean distance between points a and b in that space is calculated as follows:

$$Euclidean\ Distance = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

where n is the number of dimensions.

In order to understand how the euclidean distance is calculated, we will show and explain a simple graphical example, shown below.

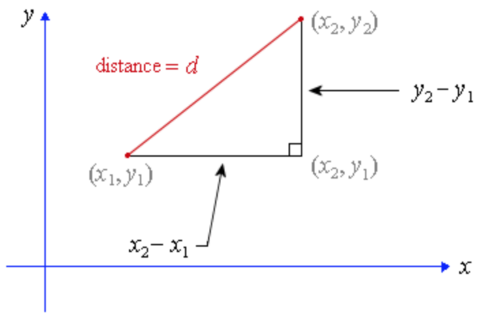


Figure 2.1: Graphical representation of the euclidean distance, d, between 2 points with coordinates $(x_1,y_1)$ and $(x_2,y_2)$

Fig 2.1 explains the concept of the Euclidean distance. From the figure we see two points in a 2 dimensional (x, y) space with coordinates $(x_1,y_1)$ and $(x_2,y_2)$. We can observe clearly that the euclidean distance is similar to the theory of pythagoras, with the euclidean distance, d, being equal to the hypothemus of a right angle triangle formed between our two points. For the above case, since we are in a 2D space, our n = 2 and the euclidean distance is calculated as using the following formula:

$$\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}$$

**Example**

In order to show how this is calculated, we will use the 5 objects presented in table 2.1. We will as an example then calculate the euclidean distance between another object, o = (22.0, 20.0, 18.0) and objects 1 = (10.0, 5.0, 10.0), 3 = (30.0, 13.0, 15.0) and 5 = (49.0, 34.0, 23.0) from our table as follows:

$$\begin{aligned} E.D(o,1) &= \sqrt{(22-10)^2 + (20-5)^2 + (18-10)^2} = \sqrt{433} = 20.8 \\ E.D(o,3) &= \sqrt{(22-30)^2 + (20-13)^2 + (18-15)^2} = \sqrt{122} = 11.04 \\ E.D(o,5) &= \sqrt{(22-49)^2 + (20-34)^2 + (18-23)^2} = \sqrt{950} = 30.82 \end{aligned}$$

While the euclidean distance is easy to calculate and can come in quite handy for easy distance calculations between 2 points in 2D and 3D spaces, it, however, has a huge disadvantage in spaces of much higher dimensionality. For such spaces, the Euclidean distance becomes extremely unreliable [3] as it loses its powers.

## Manhattan Distance

The Manhattan distance, also referred to as the *city block distance*, is another distance metrics which is quite similar to the Euclidean distance. This being because, for 2 points, p and q, in a 2D space, it just like the Euclidean creates a right angle triangle between p and q. Nonetheless, the way the right angle triangle is used to calculate the distance between these two points is quite different. While the Euclidean distance is equivalent to the hypothemus of the right angled triangle, the Manhattan distance on the other hand is calculated by adding the length of the two adjacent sides of the right angle triangle together.

The formula for calculating the Manhattan distance between points p and q in an n-dimensional space is provided below:

$$Manhattan\ Distance(p,q) = \sum_{i=1}^{n}(|p_i - q_i|)$$

Using the example and points from Fig 2.1, calculating the Manhattan between the 2 points with coordinates $(x_1,y_1)$ and $(x_2,y_2)$ gives us the formula:

$$\sum_{i=1}^{2}(|x_i - y_i|) = |x_1 - x_2| + |y_1 - y_2|$$

From above, we see that n is 2 since we are in a 2-dimensional x, y space.

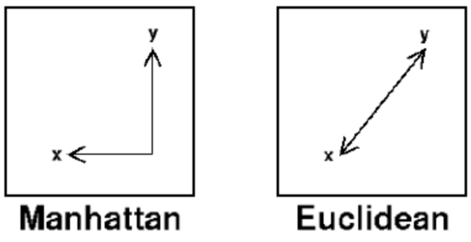


Figure 2.2: Graphical overview of how the Euclidean and the Manhattan distances are calculated between 2 points x and y

Figure 2.2 graphically shows how the euclidean and manhattan distances are calculated for 2 points in a 2D space. We observe for the Manhattan distance that it is equal to the length of the two adjacent sides of the right angle triangle formed between the two points, while the Euclidean is equal to the hypothemus of the right angle triangle formed.

### Minkowski Distance

The Minkowski distance is considered the generalized distance metric [88]. This is because tuning one of its parameters results in the Minkowski distance being equal to some other distance metric [88]. It is predominantly the generalization of the Euclidean and the Manhattan distances. The formula for the Minkowski distance between 2 datapoints X and Y where X = $(x_1, \cdots, x_n)$ and Y = $(y_1, \cdots, y_n) \in \mathbb{R}^n$ is as follows:

$$Minkowski\ Distance(X,Y) = \left(\sum_{i=1}^{n}(|x_i - y_i|)^{\lambda}\right)^{\frac{1}{\lambda}}$$

which if broken down, boils down to:

$$Minkowski\ Distance(X,Y) = \left(|x_1 - y_1|^{\lambda} + \cdots + |x_n - y_n|^{\lambda}\right)^{\frac{1}{\lambda}}$$

From the formula above, it is easy to note that when $\lambda = 1$, then the distance becomes the Manhattan distance. Furthermore, for a $\lambda = 2$, the distance becomes the Euclidean distance. For $\lambda =$ , we obtain the Chebychev distance which is also known as the *maximum value* distance. Thisstems from the fact that it is computed by taking the absolute magnitude of the differences between the coordinates of an object pair [88]. As a result of this, the Minkowski distance is considered the generalized metric distance.

### 2.1.2. Clustering Algorithms

In the field of data mining, clustering is a technique used to discover or find groups or patterns, while identifying interesting distributions and correlations in a dataset [41, 42]. Datapoints in a dataset are clustered or partitioned based on a specific similarity metric such that datapoints clustered together are similar to each other and dissimilar from datapoints from other clusters [42]. Clustering can, therefore, be seen as a natural way to distinguish between what datapoints are similar to each other and which are different, thereby achieving some sort of division.

In the field of machine learning, the search for clusters is considered unsupervised learning, with the clusters found conforming to some hidden or unknown pattern, insinuating that clustering is the unsupervised learning of hidden patterns in datasets [13]. Current literature portrays that clustering algorithms are neither classified in a canonical or straightforward manner. According to [36], clustering algorithms are divided into two main categories: **partitioning** and **hierarchical clustering methods**. In [45], these two main groups are further extended with three additional categories: **density-based, model-based** and **grid-based clustering**

**methods**. From the main clustering methods mentioned above, hierarchical, partitioning and density-based clustering methods are considered traditional clustering techniques [13, 44, 82].

In this research, we make use of three clustering algorithms which fall under the partitioning, hierarchical and the density based clustering methods which are described in sections 2.1.2 , 2.1.2 and 2.1.2 respectively. The other clustering methods are out of scope for this research.

## Partitioning clustering methods

Partitioning clustering is a clustering method where a set of datapoints is assigned to a user predefined number of clusters. If we have a dataset with n datapoints, the goal of the partitioning clustering algorithms is to partition these n datapoints into k clusters, such that each data point belongs to only one cluster and the number of clusters, k, has to be predefined by the user. It does this by moving datapoints from one cluster to the other, starting from an initial partition. In order to achieve global optimality, all possible partitions need to be enumerated, which is quite exhaustive [83]. As a result of the infeasibility of the solution, an iterative approach is adopted, which iteratively relocates datapoints between the predefined k clusters until some stop criterion is met.

One of the methods used for partitioning is the **error minimization method** which makes use of the **error minimization algorithm**. These algorithms are the most frequently used partitioning method and find clusters by minimizing a certain error criterion. The most extensively used and well-known criterion is the **Sum of Squared Error (SSE)**. This criterion which works well with isolated and compact clusters calculates the total of the squared Euclidean distances between datapoints and a certain representative value. For a clustering T with a dataset M of k clusters, the SSE is calculated as follows:

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{n_j} ||x_i^j - C_j||^2$$

where $c_j$ is the cluster of the j-th cluster, $n_j$ is the number of datapoints in cluster j and $x_i^j$ is the i-th element of the j-th cluster.

A well used clustering algorithm which uses this criterion is **K-means**. This algorithm allocates M datapoints in N dimensions into k clusters all having a cluster center such that the intra-cluster sum of squares error is minimized [46, 69]. The cluster centers, either chosen at random or by some other heuristics are used to assign datapoints into clusters at each iteration. Datapoints nearest (according to the Euclidean distance) to a certain cluster center are allocated to that cluster. The cluster centers are then recalculated and the datapoints allocated to their respective clusters until the convergence criterion is met. The complexity of P iterations of the algorithm with K cluster centers carried out on a sample of size n with R attributes is **O(P * K * n * R)** [87]. The linear complexity of the algorithm's run time makes it computationally attractive and conducive for substantially large datasets and gives it an upper hand over other clustering methods such as hierarchical clustering methods which have a non-linear complexity [83]. One disadvantage with k-means is that it may converge improperly if the initial clusters are improperly chosen.

Other partitioning methods include graph-theoretic clustering method which produce clusters using graphs. This method utilizes the idea of minimal spanning trees (MST) [102]. It creates an MST of the data and deletes the longest MST edges in order to create clusters. An example of how clusters can be formed from an MST of nine two-dimensional datapoints is shown in Figure 2.3 [49]. If the predefined k by the user was 2, we end up with 2 clusters if we delete the longest edge of the graph of length 6 between datapoints C and D , with cluster 1 containing datapoints A, B, C and cluster 2 containing D, E, F, G, H, I. If the user-predefined k is 3, then the next longest edge of length 4.5 will be deleted between nodes E and F such that we now end up with three clusters with datapoints A,B,C, D,E and F, G,H,I.

One of the main problems with partitioning clustering methods is the number of clusters which has to be predefined by the user. On the other hand, they have a huge advantage when used for applications with large datasets, for which the computation of dendograms is computationally expensive.

## Hierarchical clustering methods

Hierarchical clustering on the hand produces a set of nested clusters which are all organized in the form of a hierarchical tree. Clusters are formed by partitioning datapoints either in a top-down or bottom-up manner, where smaller clusters end up either being merged into large clusters or large clusters end up been split into smaller ones. The end result of this clustering algorithm is a dendogram which gives a high overview of the nested grouping of datapoints, the similarity between the clusters formed and the similarity levels at which

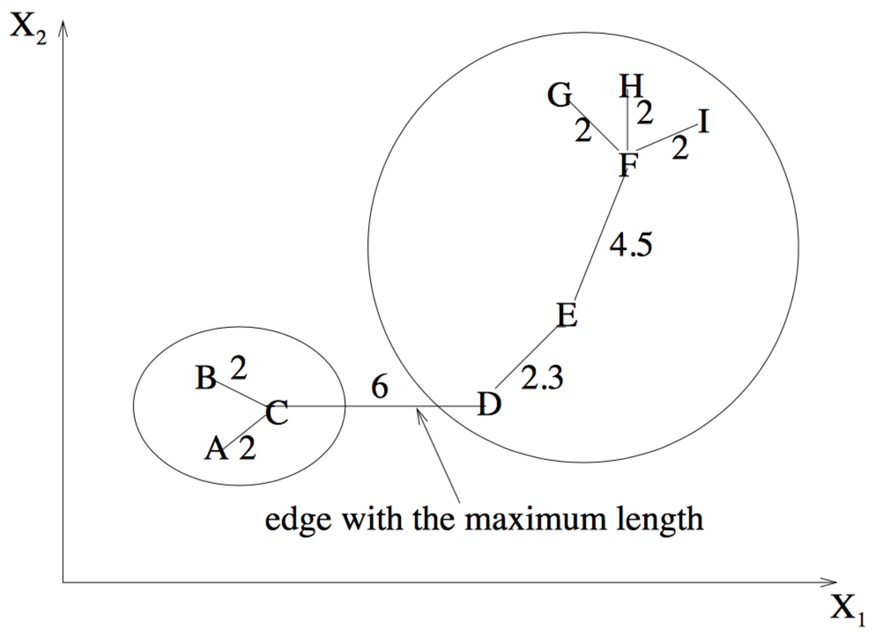


Figure 2.3: Partitioning clustering example in which clusters are formed from minimal spanning trees as shown in [49]

these clusters change. A graphical overview is shown of a clustering with six datapoints numbered in black grouped into five nested clusters numbered in red and their corresponding dendogram in figure 2.4. In this example, datapoints 3 and 6 are in cluster 1, datapoints 2 and 5 are found in cluster 2. Datapoints 3, 4, 6 are all in cluster 3 and datapoints 1, 2, 4 are located in cluster 4, with all these datapoints located in a huge cluster 5. The corresponding dendogram shows the datapoints on the x-axis and the similarity level, in this case, the distance between the points and clusters on the y axis. It shows the sequences of splits or merges. A clustering is obtained by cutting the dendogram at the appropriate similarity level. These clustering methods can further be divided into two subgroups:

- **Agglomerative Clustering**: In this case, each datapoint is in the beginning considered as a single cluster. The closest pair of clusters are then iteratively merged until some stop criteria is met or the desired cluster structure is achieved.
- **Divisive Clustering**: In this variant,all datapoints initially belong to one big cluster. This big cluster is then split into other smaller sub-clusters, which are in-turn also split into sub-clusters. This goes on until either a stop criterion or a desired cluster structure is met.

Based on the way the similarity measure is calculated, we can further divide hierarchical clustering methods into:

- **Single-link clustering**: This method considers the distance between two clusters to be the smallest distance between any two datapoints from both clusters. In the case of similarities, the similarity between two clusters is equal to the greatest similarity between any two members from both clusters [90]. This method is also called the nearest neighbor method, minimum method or the connectedness [83].
- **Complete-link clustering**: This method on the other hand, considers the distance between two clusters to be the longest distance between any two datapoints from both clusters[54]. In the case of similarities, the similarity between two clusters is equal to the least similarity between any two members from both clusters. This method is also known as the maximum method, the diameter or the furthest neighbor method[83].

- **average-link clustering**: This method considers the distance between two clusters to be the average distance between any two datapoints from both clusters. This method is also called the minimum variance method [83] and presented in [100].

The advantage of hierarchical clustering methods is that they work well on datasets containing non-isotropic clusters and they have the ability to create not a single but multiple partitions, giving the user the ability to choose the best partition based on the similarity levels. Additionally, specification of a predefined number clusters is not needed as any desired number of clusters can be obtained by cutting the dendogram. One of the major disadvantages of this method is that it has a non-linear time complexity of the order $O(n^2)$. This means that hierarchical methods have a scalabitlity issue for large datasets. Another huge disadvantage of hierarchical methods is that they do not provide back-tracking abilities as what was previously done can never be undone [83].

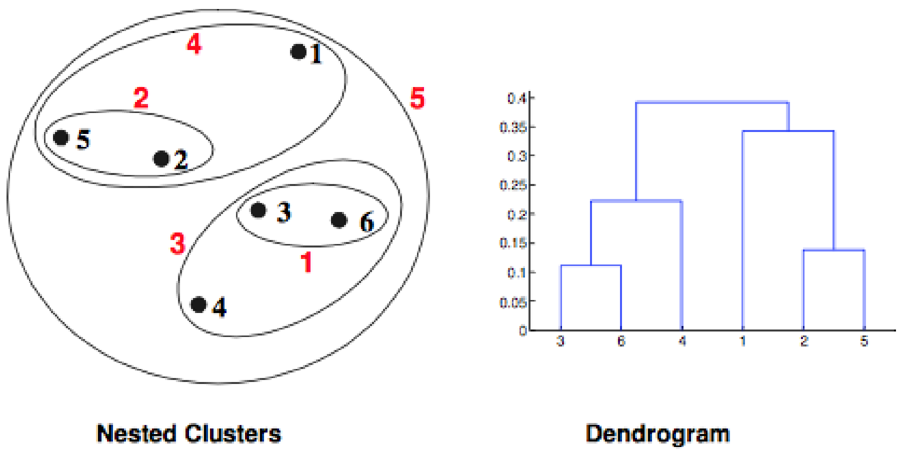


Figure 2.4: Hierarchical clustering example for a set of nested clusters numbered in red, the datapoints in the clusters numbered in black and with the representative dendogram of the nested clusters.

## Density-based clustering methods

This clustering method is based on the notion of density. Its main aim is the identification of clusters of arbitrary shapes [83]. For this clustering method, the assumption made is that datapoints are generated using multiple probability distributions and takes the probability density into consideration when coming up with clusters. Clusters are identified as common regions of high density which are separated by low density regions, where the notion of density refers to the number of datapoints within a certainly defined radius, r. The algorithm marks datapoints situated in low density regions as outliers. One of the most used density-based clustering methods is the density-based spatial clustering of applications with noise (DBSCAN) proposed in [34].

DBSCAN identifies clusters by identifying regions of high densities which are disconnected from each other by regions of low density. The high density regions are then considered clusters. The outliers in the dataset are identified as noise. The idea of density has to do with the number of datapoints close to each other. In order to cluster using DBSCAN, two parameters are required and need to be predefined: $\epsilon$(Eps) and the **minimum points**(minPts) where $\epsilon$ defines the radius of the neighborhood around a datapoint, x, called the $\epsilon$-neighborhood of x and the minPts is the number of neighbors within the $\epsilon$ radius. DBSCAN uses these parameters to identify core points, border points and outliers. The $\epsilon$-neighborhood of x in a dataset, D is represented as:

$$N_\epsilon(x): \; y \in D : distance(x, y) \leq \epsilon$$

The main or key idea here is that the density of the neighborhood around a datapoint has to exceed some threshold for it to form a cluster. The shapes of the neighborhoods found depend on the *distance* function. DBSCAN uses the aforementioned parameters to categorize datapoints into three main groups:

- **Core Points**: These are datapoints containing more than a certain number of datapoints (MinPts) within their $\epsilon$-neighborhood. The MinPts then acts as the threshold above which datapoints are considered core points. ’

- **Border Points**: These are datapoints containing less than minPts datapoints within their $\epsilon$-neighborhood but which are in the $\epsilon$-neighborhood of a core point.
- **Outliers**: These are datapoints that are neither core nor border datapoints. These are also considered as noise.

In order to understand how DBSCAN algorithm works, we need to define three terms:

- **Directly density reachable**: A datapoint, x, is **directly density-reachable** from another datapoint, y, if:
  1. $\mathbf{x} \in N_\epsilon(y)$ , which indirectly says that x is in the $\epsilon$-neighborhood of y.
  2. $|N_\epsilon(y)| \geq MinPts$ , meaning y is a core point.
- **Density reachable**: A datapoint, x, is **density reachable** from another datapoint, y, if there are a set of core points leading from y to x.
- **Density connected**: A datapoint, x, is density connected from another datapoint, y, if there exists some core datapoint, z, such that both x and y are **density reachable** from z.

With that being said, let us now explain how DBSCAN clustering works in a few steps[34]:

1. Randomly select a datapoint, x, and determine all datapoints **density reachable** from x w.r.t Eps and MinPts.
2. If x is a core point, then a new cluster is formed ifthis is not yet in a cluster and all datapoints density connected to x are placed in the same cluster as x.
3. If x is a border point, then go to the next datapoint. This means that there are no **density-reachable** datapoints from x.
4. iterate until all datapoints have visited and processed.
5. If a datapoint does not belong to any cluster, it is considered as noise or an outlier.

With DBSCAN clustering, the number of clusters to be created does not need to be predefined, giving it an advantage over partitioning clustering methods. Also, DBSCAN can identify non-linearly separable clusters of various shapes and also has the concept of noise, making it robust to outliers. DBSCAN clustering also has a number of disadvantages. The $\epsilon$ and minPts parameters have to be predefined, which could be quite difficult if the data is not well understood. Furthermore, the difficulty in choosing an appropriate minPts-$\epsilon$ combination for datasets with large variations in densities makes it impossible to cluster these datasets. DBSCAN also does not work well with high-dimensional data.

## 2.2. Related work

In the field of statistics and data mining, we notice that outlier and anomaly detections have been extensively researched and studied, eventhough with different perspectives, aspirations and motives [6, 22, 39]. In anomaly based detection techniques , ongoing network or host patterns or behaviors are compared to established patterns or behaviors of normality. If large enough deviations are observed between the new ongoing patterns or behaviors and the established patterns of normality, then these large deviations are termed anomalies[16]. Based on the type of anomaly, detection techniques can be categorized into network-based, host-based and a combination of both host and network-based anomaly detection [93].

Network based anomaly detection can be done in three different settings: supervised setting, semi-supervised setting and unsupervised setting [16]. In the supervised setting, a training dataset is made available with explicit labels for both the benign and anomalous data instances. In the semi-supervised setting, the training dataset only contains labels for the benign instances. The unsupervised setting on the other hand, no labels are available for the data instances. In the latter setting, it is assumed that normal instances occur far more frequently compared to the anomalous instances in the test dataset [16]. This research is done in an unsupervised setting.

In this research, while our main focus is on network traffic-based anomaly detection, we also have a host-based anomaly detection component meaning we have a combination of both network and host-based anomaly detection. Due to the fact that our focus in this research is the detection of anomalies (data exfiltration) in netFlow in an unsupervised environment, this section first presents some work related to Network traffic-based anomaly and outlier detection in general and also further presents other works specific to the detection of data exfiltration from netflow traffic.

### 2.2.1. Network traffic-Based Anomaly detection

Mathew [70] worked on a network traffic anomaly detection system based on packet bytes. He described an anomaly detection system which identified suspicious network traffic based on two stages; the first being a packet filter stage in which traffic is filtered to pass only the most interesting packets and the second being the flagging of the events based on their bytes values per modeled common protocols (tcp, ftp, http, smtp). Events with byte values that have not been observed over a long period of time are flagged [70]. However,due to a change in traffic characteristics in the last decade, new byte values which have not been observed before will be considered anomalous while they could actually be benign, leading to a high rate of false positives.

A histogram-based approach to anomaly detection in network traffic is presented in [53]. This approach utilizes network traffic feature (source and destination IP addresses, source and destination ports, TCP flags, protocol number, packet size and flow duration) values from the training data to create histogram baselines of every feature. These per feature histogram baselines are then used during network monitoring to raise alarms in cases where deviations above a certain threshold from the baselines are observed. A similar histogram-based approach is adopted in [18] where histogram-based methods are used to identify flows which seem suspicious and with the assumption that anomalous flows typically possess similar characteristics, association rules are employed to detect anomalous flows. Other histogram or sketch-based network traffic anomaly detection methods are presented in [48, 63, 91].

In [99], an entropy-based anomaly detection technique is proposed in which alarms are raised when the entropy contents of certain network traffic feature change. This technique uses data compression algorithms to estimate the entropy and detects worm outbreaks and extensive scanning activity in almost real time. Netflow records are represented as a compressed sequences in binary form and size of the compressed records are used as the entropy value of the contents of the sequence. It was found out that hosts involved in scanning tend to have traffic flows with IP address fields containing a smaller entropy per destination address they try to connect to compared to normal traffic. This technique uses the source and destination IP addresses and ports as features for their compression algorithms in order to obtain their entropy values. Other entropy-based anomaly detection methods in network traffic are presented in [17, 51, 74].

[59] proposes a network anomaly detection technique which makes use of the distribution of the network traffic features in order to detect a wide range of anomalies. For this analysis, the four features used were source and destination Ip addresses and the source and destination ports of every network traffic connection. Entropy values of these features are then used to show that these method catches more anomalies compared to standard volume-based anomaly detection techniques. Clustering is then used to detect more unknown anomalies based on the entropy values of the features.

Clustering is also one of the methodologies used in the detection of anomalies using network traffic data. Clustering is an unsupervised learning method which places unlabeled datapoints into different groups based on a particular similarity metric such that datapoints in the same group tend to be more similar to each other than and more dissimilar to those in a different group. Recent works using clustering to detect or identify anomalies make use of the following assumptions:

- **Assumption** If clusters are created of only normal data, any new datapoints that do not fit well with these predetermined clusters are considered anomalous [6, 34].
- **Assumption 2** For clusters containing both benign and anomalous datapoints, anomalous datapoints can be detected or identified based on their distances from their cluster centroid as benign datapoints are found to be located very close to the cluster centroid and anomalous ones further away [4, 6].
- **Assumption 3** In the case of clusters of varying sizes, the bigger and thicker clusters can be considered benign while the smaller and sparser clusters can be considered anomalous [6].

In [81], a clustering based algorithm on network flow and packet information is proposed which is based on a certain predefined and hard-coded width of the formed clusters. By assuming that the training dataset has an accurate reflection of the range and standard deviation of the entire distribution, all new datapoints are then transformed to a standardized space. The transformed standardized space is then use to hard-code the width of the cluster. It is then further assumed that a large proportion of the entire dataset consists of normal datapoints. Based on this assumption, a certain percentage of the clusters are then considered normal and the remaining ones considered anomalous. Data instances, transformed into the standardized space are then classified based on the label of the cluster they find themselves in. Datapoints in the anomalous cluster are classified as anomalous and those in the clusters labeled as normal are considered normal. [61] also propose a grid-based and density-based clustering algorithm which is suitable for unsupervised anomaly detection based on the assumptions that the majority of the network connections are benign traffic, with only a certain X % being malicious [81] and also that attack traffic differs statistically from normal traffic.

A clustering based semantic summarization algorithm is proposed in [5], where a combination of k-medoid and x-means are used as an alternative to k-means. Their experimental analysis showed that their technique performed significantly better than that of the k-means based summarization method. This techniques could be further used to detect collective anomalies in network.

[35] proposes a time-based NetFlow anomaly detection mechanism by extending the Micro Clustering Outlier Detection (MCOD) to Micro Clustering Outlier Detection in Time-series. Once the extended MCOD algorithm is achieved for a few time windows on Netflow data, polynomial regression is then used as a means to generalize the cluster densities. Anomalies are then detected based on the change in cluster densities over time. In order to do this, they considered 6 network traffic-based attributes (start time, source and destination ports, source and destination bytes and protocolD) to be considered for the detection of anomalies.

In [58], kmeans clustering is used to detect network flow-based anomalies. This is achieved by separating the training data with unlabeled flow records into normal and anomalous clusters and using these clusters to detect anomalies in new data by calculating the distance from their corresponding cluster centroids. For this purpose, they utilize the total number of packets sent, the total number of bytes sent and the number of distinct source-destination pairs within a given time interval as their feature values. Time series analysis of network streams have also been shown to work well in [9, 85].

The combination of cluster parameters and internal cluster evaluation schemes used to evaluate clustering is proposed as an anomaly detection technique for network traffic in [80]. The cluster centroid diameter used to measure the compactness of the cluster is used as cluster parameter and the Davies-Bouldin index used for maximum cluster separation is used as internal cluster evaluation scheme. This index aims to maximize the inter cluster distance and minimize the intra cluster distance. By assuming that attacks have similar or identical properties and using the cluster compactness combined with the cluster separation, they distinguish between normal and anomalous clusters. As an example, very compact clusters having a Davies-Bouldin index of 0 or close to 0 can be an indicator of a mass attack. While this method seems to be very efficient, it is used to identify intrusion detection attacks which are quite different from data exfiltrations in characteristics and also, their approach is reported to work better in a heavily attacked environment with massive attacks while data exfiltrations seldom occur and would, therefore, not be detected with this approach.

[73] presents a network data mining technique to detect anomalous network flows by making use of the k-means clustering algorithm. In order to achieve this, flows are first classified according to their port numbers and transport protocols as a means to distinguish between different services. K-means clustering is then separately applied to the different services obtained per (protocol, port) pair using total number of bytes, total number of packets and the unique source-destination pairs as features for the clustering. An initial k value of 2 is used based on the assumption that normal and anomalous traffic will both fall in different clusters and the euclidean distances from the cluster centroids are then used as a distance-based anomaly detection technique for new test data. Instance closer to the anomalous cluster centroid and those farther away from the normal cluster centroid than a predefined threshold are then considered anomalous. While a k-value of 2 is used here, no scientific method is used to back the use of this number up and the features used for clustering would not be good enough to detect a smart data exfiltrator who decides to act normal by sending small number of bytes in small packets but then over a very long period of time. Kmeans clustering is also used in [89] to analyze and visualize network flow data with the main aim of trying to detect anomalies.

A graph-based anomaly based detection mechanism for communities is presented in [101]. Information across various institutions considered as communities is collected and communication graphs for IP addresses within a community are drawn with the nodes representing different organizations and the edges connecting that IP address to the different organizations that particular IP address has communicated with. The width of the edges quantifies the importance of that particular communication. Suspect behavior within organizations can be detected by comparing communication behavior across member organizations in that community. In this manner, targeted attacks on communities of interest such as financial institutions, e-commerce sites can be detected.

[92] proposes a different approach which uses big data to identify anomalies in network traffic. They came up with a new unsupervised detection approach based on an Apache spark cluster in Azure HDInsight to identify anomalies caused by UDP flood attacks from particular IP addresses. Their method delivered a 96% accuracy.

Mohammed et al propose a clustering algorithm to detect outliers [8]. This is done by identifying suspicious clusters of unusual sizes, which are far away from the other clusters and performing an analysis on these clusters.

In order to detect network traffic anomalies in private organizational networks, [94] first of all studies the patterns of typical network usage within a corporate environment. Based on the observed or studied patterns, two novel approaches are presented to detect network traffic anomalies. The first approach monitors for legitimate client to service communications and based on that, detects abnormal TCP and UDP flows. The second approach on the other hand employs clustering as a means to detect clients with a sudden change in traffic behavior.

## 2.3. summary

In this chapter, we provide necessary background knowledge required to fully understand the rest of the report. We initially start out by explaining four commonly used distance metrics (mahalanobis, euclidean, manhattan, minkowski) and how they are calculated. Furthermore, we present and explain three clustering methods extensively used (partitional, hierarchical and density-based clustering methods). We conclude the chapter by presenting relevant and recent literature in the field of network traffic anomaly and outlier detection.

# 3
# Feature-Set Exploration

In this chapter, we present and provide motivations for the feature sets that are used to detect possible data exfiltrations at both the connection-level and host-level granularity. Additionally, we further discuss how each of the features in the different feature sets are normalized.

In order to come up with the feature sets for the detection of data exfiltration, we generally consider the following:

1. A naive attacker exfiltrates data by sending out files at once in a very short period of time [27].
2. An advanced attacker may exfiltrate data by splitting a whole file into smaller fixed size chunks and sending these smaller fixed size chunks out to the target at different times [27].
3. A more advanced attacker may exfiltrate data by not only splitting a whole file into smaller chunks but also by sending this fixed size chunks at regular or scheduled time intervals [28]

With the aforementioned in mind, we now propose a set of features chosen from literature aimed in detecting data exfiltration at connection-level and host-level granularities, presented in Sections 3.1 and 3.2.

## 3.1. Features To Detect Data Exfiltration On Connection-level Granularity

The feature set in this section is calculated on a connection-level granularity. As earlier explained, this research focuses on outbound external connections only, which are connections from an internal Adyen network to an external network. The features used and their motivations are presented below.

### 3.1.1. Duration

The firs feature is the duration of a connection. From the hypothesis we considered above, we see the importance of the notion of duration. When a connection is initiated between two hosts, this session is logged in the netflow collector with several parameters as explained in Section 1.3. One of the parameters or fields which are logged for every connection is the **state** field which takes values **NEW** at the commencement of a communication channel between two hosts and **DESTROY** once the communication between these two hosts is terminated. For each of these events (commencement and termination of a communication between two hosts), the **timestamp** values are also recorded alongside other parameters. Note that millions of other connections could be established between the start and end of a particular session between two hosts. In order to calculate the duration of the connection between two hosts, we need to find the matching tuple [src.ip, dst.ip, src.port, dst.port, proto]for the initiation and termination of these connections and then convert their timestamps which are in Iso 8601 notation to datetimes. The datetime for the termination of the connection is then subtracted from that of initiation of the same connection to get the duration.

This allows us to monitor how long a session between two hosts takes and also identify deviations in the duration of communications between these hosts. This feature on its own does not give a lot of insights, but when combined with other features such as **source bytes**, could help identify connections that may correspond to data exfiltration as proposed in [77]. As an example, if a connection that normally sends about 200 bytes of data in 5 mins all of a sudden sends a thousandfold increase in source bytes in less than 10 seconds, it may be involved in data exfiltration. In this research, the duration of a connection is represented in seconds.

### 3.1.2. Source bytes

Per connection, we extract the amount of information uploaded by the initiating host of the connection. These values are directly extracted from the netflow collector without any modifications.

This feature permits us to monitor deviations in the number of uploaded bytes by hosts initiating a connection, as these may also correspond to data exfiltration. This feature alone can be used to detect very naive data exfiltrations where very large source bytes are uploaded in a single connection, but can not be used to detect more advance data exfiltrations in which the exfiltrator uploads small bytes but within a long period of time, for example. If a host suddenly exhibits a hundredfold increase in the number of uploaded bytes, it may correspond to data exfiltration [77]. In this research, the uploaded amount of information by a host is represented in bytes.

### 3.1.3. Source bytes per packet

This feature calculates the amount of information uploaded per packet by the initiating host of a connection.

This feature gives us an estimate of the amount of bytes uploaded per uploaded data packet to a particular destination IP . The motivation for this feature is to monitor the uploaded bytes in a single uploaded packet. Also, a ten or hundredfold increase in uploaded bytes per packet means a host is sending a lot more information in every sent packet than usual and this could be an indicator of a data exfiltration. This value is represented in bytes per packet.

### 3.1.4. Source Bytes Per Second

This feature records the amount of information uploaded by the initiating host per second. The **source bytes** values are gotten from the netflow collector while we calculate the **duration** values.

This feature allows us to view the number of uploaded bytes relative to the amount of time used to upload the data. This helps us observe deviations in the number of uploaded bytes per second for a connection compared to its peers. A flow in which huge amount of bytes are uploaded per second compared to its peers might be an indicator of data exfiltration. For the purpose of this research, this feature is measured in bytes per second.

### 3.1.5. Producer-Consumer Ratio

The producer-Consumer ratio was introduced by Carter Bullard and John Gert [21] as a metric for the detection of data exfiltration. This can be considered for clarity purposes as the sender-receiver ratio.

The main idea behind this feature is that every node in a network is either a sender or receiver of information. It gives a good indication of the pattern in a particular connection while ignoring more complicated information about that connection. It is can be calculated as follows:

$$PCR = \frac{(sourcebytes - destinatonbytes)}{(sourcebytes + destinationbytes)}$$

The PCR value is always a number within the range[-1,1] and can be interpreted as shown in Table 3.1 [21]. Connections with a high PCR value ($\geq 0.7$) are considered pure pushers because they send out significantly more information than they receive. Connections with a very low PCR value ($\leq -0.7$), on the other hand, are regarded as pure pullers because they receive significantly more information than they send out. A connection doing a pure push could be involved in a potential data exfiltration

| PCR | Connection role |
|---|---|
| 1.0 | Pure push - FTP upload, multicast, beaconing |
| 0.4 | 70:30 export - Sending Email |
| 0.0 | Balanced Exchange - NTP, ARP probe |
| -0.5 | 3:1 import - HTTP Browsing |
| -1.0 | Pure pull - HTTP Download |

Table 3.1: PCR values and their corresponding interpretations from [21]

## 3.2. Features To Detect Data Exfiltration On Host-level Granularity

In this section, we present the features for host-level granularity. Most of the features chosen for host profiling are selected from previous research [30, 50, 62].

### 3.2.1. Number of Peers

For every internal host, the number of peers refers to the number of unique or distinct destination IPs it engages in a communication with. These are destination IPs to which one or more than one packet is sent to during the communication.

The purpose of this feature is to monitor and identify anomalous behaviors correlated with changes in the number of distinct or unique destination IPs. When used or combined with other features such as **source bytes**, it can be a good indicator of a data exfiltration. Given a time window or time frame, if the number of uploaded or source bytes for an internal host increases drastically while the number of unique destination IPs communicated with by an internal host remains fixed or stable, this could correspond or be a case of data exfiltration. This feature can also be used to identify and distinguish different types of communications as well such as one-to-one communications which could signify a download or upload and one to many which could be network scans [30].

### 3.2.2. Number of Unique Source Ports

For every host, we calculate the total number of unique source ports it uses for communication with other services within the observed time frame.

This feature reflects the role of the host. Large values can be clear indicators of other attacks such as port scans. Although this may seem as an out of scope attack for this research, we still take it into account since hosts who carry out port scans might also be looking for potential ports to exfiltrate data.

### 3.2.3. Number of Unique Destination Ports

This feature represents the total number of unique destination ports used per host for communication with other services within an observed time frame.

This reflects the role of the host. A high number corresponds to scanning for open ports on a single or several IPs while an extremely low value may represent scanning for a single port on several IPs. This can be an indicator of a port scan for potential data exfiltration.

### 3.2.4. Source Ports per Peer Ratio

This feature is somewhat related to the feature presented in Section 3.2.2. In this case, we calculate the ratio of the number of unique source ports to the number of unique peers per host.

This feature also reflects the role of the host in the network and large values which indicate large numbers of initiated connections may betray port scans which could be an indicator of a search of port for data exfiltration. This arises from the fact that servers use a single fixed port for replying to classical protocols while clients on the other hand, open a new mostly random port for each connection to a server.

### 3.2.5. Destination Ports per Peer Ratio

This feature, related to that in Section 3.2.3 calculates the ratio of the number of unique destination ports to the number of unique peers per host.

This feature just like the previous three features also shows the role of the host in the network. This is a better feature compared to that in section 3.2.3 but for the sake of relativitiy, we calculate both values.

### 3.2.6. Mean Packets per flow

For every host, this feature calculates the average of the number of packets sent over all its flows.

This shows a clear difference between hosts containing predominantly large-sized flows and those containing small sized flows, which could also be an indication of hosts that are involved in data exfiltration. Note that this is calculated for emitted traffic and not incoming traffic.

### 3.2.7. Percentage of Small-Sized Packets

For this research, just like in [30], we calculate the percentage of flows per host that emit traffic with number of uploaded bytes $\leq 144$ bytes.

### 3.2.8. Percentage of Large-Sized Packets

This feature unlike that in 3.2.7 calculates the percentage of flows per host emitting traffic $\geq 1392$ bytes. This feature is also used in [30] to classify network hosts based on their connection behaviors.

Flows with large sized packets are involved in data exchange. Although the content of the exchange is not known, a scenario where a host who normally sends out small-sized packets suddenly starts sending out large-sized packets to a particular external IP node might be an indication of data exfiltration.

### 3.2.9. Total Number of Uploaded Bytes

For every internal host in our network, we calculate its total number of uploaded bytes within a particular timeframe.

This enables us to monitor for deviations in the number of uploaded bytes per host as this may also be an indicator of data exfiltration. If a host all of a sudden increases its amount or number of uploaded bytes by a certain huge or large number, this this could be correlated to a data exfiltration attack. For the purpose of this research, the total number of uploaded bytes per host is represented in bytes.

## 3.3. Discarded Features

Investigations on what features to use at both granularities led to the dropping of some features which are logged in the netflow collector. In this section, we present the features which were present in the netflow collector and were discarded. While some features are in their entirety not made use of, others are used to derive some of the features used.

### 3.3.1. Network traffic recorder

This feature contains the type of service from which the logs on the netflow collector was captured from, depending on the type of connection (internal or external). As shown in Figure 1.1, external connections go through the router before being sent out to the internet. These connections then have the **router** as parameter value for this feature. Internal connections on the other hand go through the **firewalls** and have this parameter as value for this feature.

Since we focus on only outbound external connections for this research which all go through the router, all our connections end up having the **router** as the value to this feature.

### 3.3.2. Port

This feature indicates the ports used by two communicating hosts to communicate with each other.

Attackers tend to use certain common ports to carry out attacks over the network such as data exfiltration [25]. Knowing that these ports are been monitored, some attackers adhere to using alternative ports to evade detection. This feature is discarded to avoid missing exfiltrations carried out using normal or alternative ports.

### 3.3.3. Protocol

This indicates the protocol used for communication between 2 hosts. This could be either tcp, udp or icmp as observed from the netflow logs.

Although certain protocols can be predominantly used to exfiltrate data such as common protocols used by command and control servers, attackers avoid detection by adhering to alternative protocols [26]. As a result, in order not to limit our detection mechanism to commonly used protocols, we discard this feature. This gives us the possibility to detect data exfiltrations that are carried out using other techniques or other alternative protocols.

### 3.3.4. State

Every connection initiated or terminated by an internal host either to an internal or external service gets allocated a state. This state either takes the parameters **New** for a newly initiated connection or the state **Destroy** for the termination of an already initiated connection between two hosts.

Although this feature is not used for analysis, it is used to derive other features used in this research. It is mainly used to calculate the duration of connections.

## 3.4. Feature Normalization

Once the features are calculated, our next step is normalization. It is a scaling technique, where we can create a new range from an already existing one, for the purpose of clustering and further analysis. While some clustering techniques do not require a normalized dataset to establish similarity or dissimilarity between datapoints, other techniques which utilize a distance measure as a similarity or dissimilarity metric do require this. The main aim of feature normalization is to scale each individual feature value between a value of 0 and 1, such that each feature becomes equally important. The normalization of connection-level features and host-level are presented below.

### 3.4.1. Connection-level feature normalization

Connection-level features are normalized using the **min-max normalization** technique. This technique uses the original range of data in order to provide a linear transformation to a range [0, 1] with the minimum value of a feature being transformed to 0, the maximum value of that feature to 1 and every other value gets transformed into a decimal between 0 and 1.

This is achieved using the formula for a feature, y,:

$$y_{normalized} = \frac{y - min(y)}{max(y) - min(y)}$$

The above formula gets the minimum and maximum values from the list of feature values and divides the difference between the current value and the minimum value by the difference between the maximum and the minimum values.

### 3.4.2. Host-level feature normalization

Host-level features are normalized using a non-linear transform to balance out the difference in scales or ranges between features as presented in [30]. The non-linear transform used to achieve this is provided below:

$$F_n : f_n = \left( (\frac{2}{\phi}) arctan(\frac{F_n}{R_n}) \right)$$

where $F_n$ is the normalized value for feature n, $R_n$ is a parameter specific to feature $f_n$ such that for feature number i, $R_i$ represents the average number of peers and feature $R_{ii}$ to $R_v$ and $R_{ix}$ are set to the not normalized number of peers. Since features 3.2.7 and 3.2.8 are percentages, we can just divide these values by 100 to get a normalized value. Feature 3.2.6 is scaled by using the value $R_{vi} = 100$.

## 3.5. Summary

In this chapter, we present the feature-set which according to us, is best to detect data exfiltration at connection-level and host-level granularities. On a connection-level granularity, we select 1) Duration of a connection, 2) Source bytes, 3) Source bytes per packet, 4) Source bytes per second, 5) Producer-consumer ratio as feature-set. For this granularity level, we discard the port and protocol as features. We additionally present the technique used to normalize the selected feature values, known as the min-max normalization technique.

# 4 Dataset Exploration

In this section, we explain how the training, testing and validation datasets are collected, and further present the characteristics of each the datasets. All datasets used in this research are collected at Adyen. For all the tests and validation datasets, the source IP addresses in this dataset are mapped to random IP addresses and the dates are changed in such a way that the days still remain the same, such that a Tuesday in the originally collected dataset remains a Tuesday in this dataset. Subsequently, we provide the evaluation criteria used in this research.

## 4.1. Dataset Collection

### 4.1.1. Training dataset

For our initial training dataset, also referred to as training set 1, training dataset 1, first training dataset or first training set in this report, we collected my personal netflows (originating from my laptop) at Adyen for a day from **Tuesday February 5th 2019 at 11:06 Am CET** until **Wednesday February 6th 2019 at 11:05 Am CET**. Note that this dataset contains flows for just a single host and the day for the collection is selected at random. A total of 8458 flows containing 4188 established connections are logged and collected during the collection period. This dataset contains only benign connections.

### 4.1.2. Test datasets

#### First test dataset

This dataset, also referred to as test set 1, test dataset 1, first test dataset or first test set was collected on the **14th of February 2019**. **No prior knowledge** of the data exfiltration detection anomaly scoring engine was provided. Partial knowledge was present about some of the features used by our model to detect data exfiltration. This depicts an internal attacker or a compromised system trying to exfiltrate data, knowing there is a data exfiltration model in place but with partial knowledge of the features used to detect data exfiltration, no knowledge of the clustering engine, the anomaly scoring engine and the threshold used for data exfiltration detection, not caring about been detected.

We collected my personal network flows at Adyen for about 30 minutes starting from **11:00 AM CET until 11:27 AM CET**. It is important to pay particular attention to the fact that this dataset contains flows for just a single host (me), and the exfiltrator just wanted to exfiltrate data at all cost without caring about the detection mechanisms in place or been undetected. A total of 276 flows containing 126 connections are established and logged during this 30 minutes period. While we expect all of these connections to be normal connections without any exfiltration, the dataset on the other hand contains 2 instances of a simulated data exfiltration, both to a single internal destination IP address. These are simulated with the sole purpose of testing by uploading 2 large files of sizes 10MB and about 50MB to an internal Adyen storage system (with an internal destination IP address) in 2 separate connections. Note that for this research, we focus only on **outbound or external connections** (connections from an internal Adyen IP address to an external IP address) in order to detect data exfiltration, but for the sake of this simulated exfiltration instances using my internal IP, we consider the internal IP address of the Adyen storage system to which the large files are sent as an external IP address. The main reason for considering only external or outbound connections in this research is because inbound connections (connections from an external IP address to an internal Adyen IP address) seldom occur and are already being carefully monitored at Adyen. Connections within internal IP addresses are also not considered because the data stays within the company's network boundary.

### Second test dataset

**No knowledge** of our data exfiltration detection model was available the simulation of the data exfiltration scenarios in this dataset. No knowledge was provided about the features, the clustering algorithm and the anomaly scoring algorithms been used for detection and their detection thresholds. This is considered to be an internal attacker or a compromised system trying to exfiltrate data, knowing there is a data exfiltration model in place but with no knowledge of what it uses for detection or how it detects data exfiltration.The exfiltrator tries to evade detection in some of the simulated exfiltration scenarios while at the same time not caring about been detected in others.

It contains a week of netflow for four randomly chosen hosts within the company. The dataset is made up of a total of 63220 flows with 31542 established connections from 4 different hosts, who will in this research be further referred to as Host A - D as shown in Table 4.1. The dataset contains atleast 1 malicious host and 1 or more data exfiltration scenarios. No previous knowledge of these hosts and the data exfiltration scenarios in place are known to us at the time of testing. We then evaluate the detection performance of our model on unknown data exfiltration scenarios.

This test dataset, also referred to as test set 2, test dataset 2, second test dataset or second test set in this report contains data exfiltration instances that we were unaware of. Two out of the four hosts in this dataset were involved in atleast one data exfiltration scenario, where one host exfiltrated 100+ MB of data over two different websites, in a very small period of time. The second host on the other hand, split the file of sensitive information into chunks of 100-300KB and uploaded it to an amazon webhost over a period of five hours .Explaining it more in depth,

- **Scenario 1:** Host A exfiltrates 3 files of sizes 50MB, 100MB and 200MB in a total of 20 different netflow connections to 5 different external hosts, to two different websites which we will refer to in this research as Website A and Website B. The exfiltration to Website A occurred in 9 of the 20 netflow connections, involving 2 different external hosts. However, the exfiltration to Website B occurred in 11 of the 20 exfil connections and involved 3 different external hosts.

- **Scenario 2:** Host D on the other hand uploaded a 155MB file via git to github.com, in a single netflow connection. In this exfiltration scenario, the exfiltrator uses a popularly and highly used service at Adyen (github.com) to seem exfiltrate data without been detected. This scenario is similar to the exfiltration scenarios in the first test dataset.

A total of 21 netflow connections were, therefore, involved in data exfiltration.

| **Host** | **Total number of netflows** | **% of total network traffic** | **Number of netflows involved in exfiltration** | **% of netflows involved in exfiltration** |
|---|---|---|---|---|
| Host A | 9442 | 29.9% | 20 | 0.2 % |
| Host B | 7625 | 24.3% | 0 | 0 % |
| Host C | 7617 | 24.1% | 0 | 0 % |
| Host D | 6858 | 21.7% | 1 | 0.014 % |
| **total** | **31542** | **100 %** | **21** | **0.066 %** |

Table 4.1: Hosts contribution to the network traffic in the second test dataset

Table 4.1 shows the number of connections made by each host and the percentage of the network traffic they occupy in the second test dataset. While Host A makes the highest number of connections and exfiltrates data in 5 of his total connections, thereby being responsible for almost 30% of the network traffic with 0.2% of his network traffic connections being exfiltration. Host D on the other hand makes the least number of connections and exfiltrates data in 1 of these connections, therefore, being responsible for 21.7% of the entire network traffic with 0.014% of his network traffic involved in exfiltration. Hosts B and C make almost the same number of connections, both contributing to about 24.2% and 24.1% respectively of the network traffic, and not being involved in any exfiltration at all. This confirms the fact that exfiltrations always happen in very small proportions of the network traffic, making its detection even more difficult as explained in the last two paragraphs of Section 1.2. In total, 0.066 % of all connections were involved in an exfiltration in this dataset.

### Third Test Dataset

**All knowledge** of our data exfiltration detection model was available during the simulation of the data exfiltrations contained in this dataset. Full knowledge of the features, the clustering algorithm, the anomaly scoring algorithms used for detection and their detection thresholds were provided. This scenario depicts an internal attacker or a compromised internal system trying to exfiltrate data, knowing there is a data exfiltration model in place, having full knowledge of what it uses for detection and how it operates. The attacker in this case is trying to exfiltrate data without been detected.

This dataset,also referred to as test set 3, test dataset 3, third test dataset or third test set in this report contains a week of netflow metadata for eleven different hosts within the company. This dataset contains 168,497 flows with 84,050 established connections from 11 different hosts who will in this research further be referred to as Host E - O as shown in Table 4.2. This dataset contains one malicious host and one or more data exfiltration connections. I sat down with Adyen's security team and explained, having full knowledge of the model, how data could be exfiltrated without it been detected. We then evaluate the detection performance of our model when the different anomaly scoring metrics presented in Section 6 are used.

The dataset contains 1 malicious host (host I on Table 4.2) who exfiltrates data by:

- **Scenario 3:** Splitting a large file into chunks of 10KB and uploading it to an amazon host over a period of 3 and a half hours with a random delay of about 1 to 3 minutes between each upload. This host exfiltrated these chunks of data in a total of 102 connections to a single external host, with the principal aim of **evading detection by using his full knowledge of the model.**

| **Host** | **Total number of netflows** | **% of total network traffic** | **Number of netflows involved in exfiltration** | **% of netflows involved in exfiltration** |
|---|---:|---:|---:|---:|
| Host E | 14838 | 18.7% | 0 | 0.0% |
| Host F | 14214 | 16.9% | 0 | 0.0% |
| Host G | 13456 | 16.0% | 0 | 0.0% |
| Host H | 12775 | 15.2% | 0 | 0.0% |
| Host I | 8781 | 10.4% | 102 | 1.16% |
| Host J | 4866 | 5.8% | 0 | 0.0% |
| Host K | 4541 | 5.5% | 0 | 0.0% |
| Host L | 4513 | 5.3% | 0 | 0.0% |
| Host M | 2585 | 3.1% | 0 | 0.0% |
| Host N | 2341 | 2.8% | 0 | 0.0% |
| Host O | 1140 | 1.3% | 0 | 0.0% |
| **total** | **84050** | **100%** | **102** | **0.12%** |

Table 4.2: Hosts contribution to the network traffic in the third test dataset

Table 4.2 shows the number of netflows made by each internal host and the percentage of the network traffic they occupy in the third test dataset. Hosts E - H and J - O do not exfiltrate any data, giving them a 0% on the percentage of exfiltration connections. While Host E makes the highest number of connections, being responsible for about 18% of the entire network traffic, host O on the other hand makes the least number of connections, being responsible for just 1.35% of the total network traffic in the third test dataset. For this dataset, the exfiltrating host does not contain the largest amount of the network traffic, but is noticed to exfiltrate in just 1.16 % of all its established connections. This confirms the fact that exfiltrations always happen in very small proportions of the network traffic, making its detection even more difficult as explained in the last two paragraphs of Section 1.2. In total, 0.12% of all netflows are involved in an exfiltration.

### 4.1.3. Validation Set

**Partial knowledge** of our data exfiltration detection model was available during the simulation of the data exfiltration scenarios in this dataset. Full knowledge of the features, the clustering algorithm and the anomaly scoring algorithms been used for detection were provided, but no knowledge about the detection thresholds. This is a scenario that depicts an internal attacker trying to exfiltrate data, knowing there is a data exfiltration model in place and having full knowledge of the system components but lacking knowledge of the thresholds used for detection.

The validation set contains a week of netflow metadata for 11 hosts within the company randomly selected and contains only outbound connections. The dataset consists of 149781 network flows with 74437 established connections. The random internal hosts present in this dataset will further be referred to as hosts P - Z as shown on Table 4.3. The dataset contains 1 malicious internal host involved in 5 data exfiltration scenarios (scenarios 4 - 8 shown below). The exfiltration scenarios in this set are:

- **Scenario 4:** Uploaded 155KB to gofile.io. This exfiltration was to a single external host or IP address in 4 netflows.
- **Scenario 5:** Uploaded 145MB to wetransfer.com and downloaded 133MB from wetransfer.com. The upload was to a single external host in 6 netflows. The download was also from a different host IP address and was done in 3 netflows. This exfiltration scenario, therefore, involved 1 external host Ip addresses and 6 netflows. In this research only external outbound connections are considered so the downloads in 3 netflows are not considered.
- **Scenario 6:** Uploaded 57MB to filebin.net in 7 netflows.
- **Scenario 7:** Uploaded 209MB to a popularly used file cloud system by employees within the company in 4 netflows.
- **Scenario 8:** Uploaded 630KB (small) to an amazon server in small chunks of about 4.3KB every 2.5 minutes. Also, the server responded on every request with 5 paragraphs of lorem ipsum (3KB), resulting in a total response of 375KB. Do note that there is also an ssh session (port 22) to the external host IP address. This scenario involved 131 netflows.

In total, these 5 scenarios involved 5 external host IP addresses and 152 netflows.

| **Host** | **Total number of netflows** | **% of total network traffic** | **Number of netflows involved in exfiltration** | **% of netflows involved in exfiltration** |
|---|---:|---:|---:|---:|
| Host P | 6737 | 9.1% | 0 | 0.0 % |
| Host Q | 2805 | 3.8% | 0 | 0.0 % |
| Host R | 7302 | 9.8% | 152 | 2.1 % |
| Host S | 12281 | 16.5% | 0 | 0.0 % |
| Host T | 5467 | 7.3% | 0 | 0.0 % |
| Host U | 15211 | 20.4% | 0 | 0.0 % |
| Host V | 9693 | 13.0% | 0 | 0.0 % |
| Host W | 4353 | 5.9% | 0 | 0.0 % |
| Host X | 3079 | 4.1% | 0 | 0.0 % |
| Host Y | 5811 | 7.8% | 0 | 0.0 % |
| Host Z | 1698 | 2.3% | 0 | 0.0 % |
| **total** | **74437** | **100 %** | **152** | **0.2 %** |

Table 4.3: Hosts contribution to the network traffic in the validation dataset

Table 4.3 shows per host the number of netflows made, the percentage of the total network traffic which they contributed, the number of netflows involved in an exfiltration and the percentage of netflows involved in an exfiltration. We observe that host U has the highest amount of netflows, contributing to 20.4% of the total network traffic, none of which are involved in an exfiltration. Host R on the other hand contributes to 9.8 % of the total network traffic, with 2.1 % of his netflows involved in an exfiltration. While the other hosts contribute differently to the network traffic, none of them are involved in any exfiltration. In total, 0.2 % of the netflows in this dataset are involved in an exfiltration.

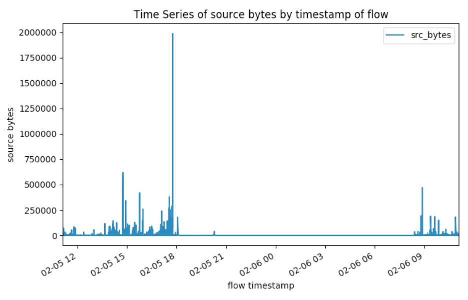
A brief overview of the characteristcis of each of the test sets used in this research is provided in Table 4.4. This table shows per test set the number of hosts involved in the data collection, the total number of connections or netflows in the dataset, the total duration for which the dataset was collected, the number of individual connections or netflows involved in an exfiltration and the total percentage of the connections in the dataset involved in an exfiltration. Note that these exfiltrations are simulated.

| Test Set | Number of hosts | Number of connections | Duration of collection | Number of exfil connections | Total exfil connections (%) |
|---|---|---|---|---|---|
| Test set 1 | 1 | 126 | 30 mins | 2 | 1.6 |
| Test set 2 | 4 | 31542 | 1 week | 21 | 0.066 |
| Test set 3 | 11 | 84050 | 1 week | 102 | 0.12 |
| Validation set | 11 | 74437 | 1 week | 152 | 0.2 |

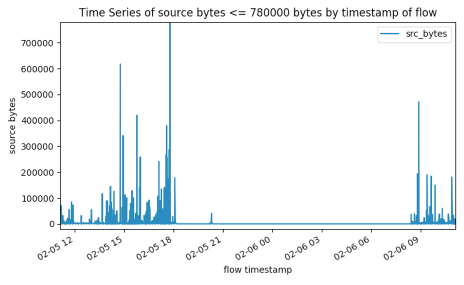
Table 4.4: Characteristics of the different test sets used in this research

## 4.2. Dataset Manipulation

First of all, since the netflow metadata contains the **timestamp** feature, we start by analyzing the time series pattern of the source bytes per connection in search of any obvious trends or abnormalities. The observed time series for the training and first test datasets are shown in Figures 4.1 and 4.2 respectively. Although we use several test datasets in this research, the time series for the first test datasets are considered to be representative of the other datasets.



(a) Complete training dataset source bytes time series plot

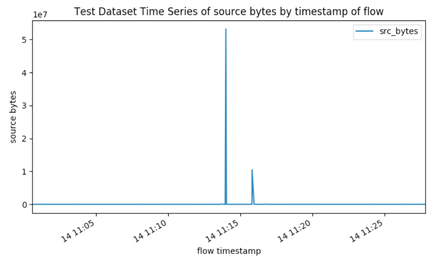


(b) Partial training dataset source bytes time series plot

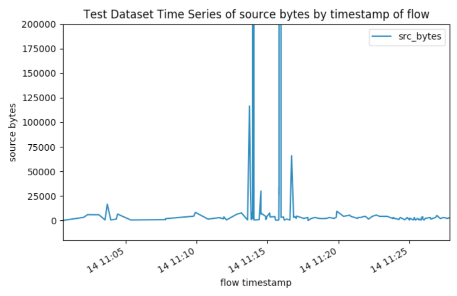
Figure 4.1: Time series of source bytes for all network connections in training set

Figure 4.1a shows the time series for the entire first training dataset with the timeframe (in the format day-month hour of day) on the x-axis and the source bytes on y-axis. From the time series, we notice no network activity between just after 6PM on Tuesday February 5th 2019 and around 8:30Am the day after, with very little activity around 8PM. In order to have a better view at the time series, we limit the scale of the y-axis to about 780000 bytes as shown in Figure 4.1b. Nonetheless, we learn that very little or no traffic is normally sent between certain times of the day. This can be used to detect possible exfiltrations as deviations from this trend might be abnormal and be a possible indicator of exfiltration. But this alone does not provide enough reasoning for a connection to be considered as an exfiltration, mainly because of the fact that Adyen has offices in different continents with different time zones with some employees working on weekends.

Figure 4.2 on the other hand shows the time series for the entire first test dataset used. As explained, this



(a) Complete Test set 1 source bytes time series plot



(b) Partial Test set 1 source bytes time series plot

Figure 4.2: Test dataset 1 time series of the source bytes for all network connections

test set contains 2 instances of a simulated data exfiltration to an internal storage system. While most of the flow seems consistently low, we notice two spikes which correspond to the 2 simulated exfiltration cases. In order to have a better view at the flow patterns for the non-exfiltration cases, we limit our source bytes axis to 200000bytes as shown in Figure 4.2b. Figure 4.2a shows the time series for the flow entries in test set 1 with their original x and y axes.

Although both the first training and the first test sets both contain spikes, it is worth mentioning that the magnitude of the different connections differs enormously. While the spike in the daily normal flow is in the magnitude of $10^6$ bytes (1986764 bytes), the spike in the abnormal flow in the test set is in the magnitude of $10^7$ bytes (53234114 and 10441353 bytes). In order to make a clear distinction between these flows, we look at their durations. We observe that the connection in the first training set that resulted in a spike had a duration of 1681 seconds which is almost 30 minutes while in the test test, the connection with 53234114 bytes took 80 seconds which is less than 2 minutes and that with 10441353 bytes took 167 seconds which is less than 3 minutes.

The above observation indicates that uploading a huge amount of data in a very short period of time can be a good indicator of a possible exfiltration. This emphasizes on the fact that **duration** and the amount of uploaded information per second (**source bytes per second**) maybe good indicators of a possible data exfiltration.

The observation of the spikes in the first test set shows that source bytes alone can be used as a feature on its own to detect data exfiltration scenarios where extremely large amounts of information is been uploaded to the internet. While this might be a false positive, it is still wise for a network administration to have a look at such connections. As a result of this, we consider source bytes as one of the features for data exfiltration detection.

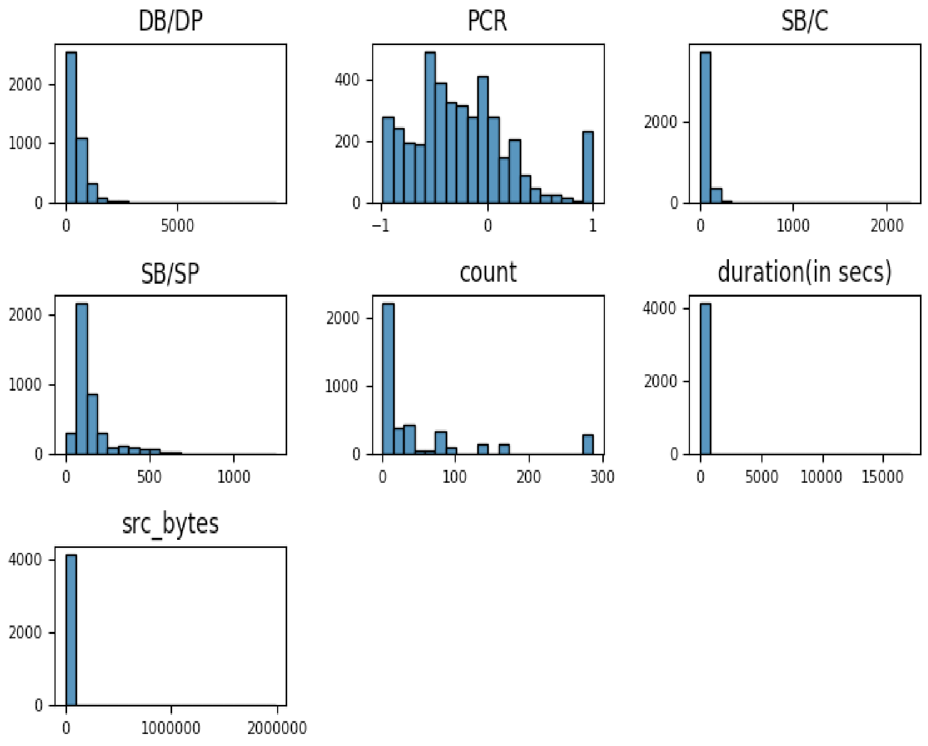


Figure 4.3: Feature histograms for connections in training set

With this in mind, we then go on to look at the values of the features selected for the connections in our training dataset. In order to have a visual overview of the distribution of the data per feature in our training dataset, we analyze the feature histograms of all the connections in the training set. The results of this are shown in Figure 4.3. It is immediately obvious that they are all skewed. While all the other features appear to be highly positively skewed, we see that the **PCR** and **SB/SP** features are lessly positively skewed. Also, it is quite interesting to notice how the $src\_bytes$ and the **duration(in secs)** all have a single bin because most of the values fall around the same value. The **DB/DP** shows that more than three quarters of the connections in our first training dataset have a destination byte to destination packet ratio value between 0 and 1500 with a few connections having higher values of about 2500. Also, more than three quarters of the connections in our first training dataset have a PCR value ≤ 0.6, with about 300 connections having a PCR value ≥ 0.7. The PCR feature and the meaning of various values is explained in pcr and a clear table with these explanations is provided in Table 3.1. This means that very few connections actually do pure pushes or uploads to the internet. Connections doing pure pushes should, therefore, also be investigated as these are potential data exfiltration connections. Quite interesting to note is that all connections in our first training set uploaded less than 500 bytes per second. Deviations from this would mean a host is trying to send out more information at a faster rate which might also be an interesting connection to look at, especially if the amount of bytes sent out per second are extremely large. On average, more than three quarters of the connections sent out less than 500 bytes of information in a single packet. Connections sending larger amounts of information in a single packet may also be interesting to look at. The distribution of these features can also be seen in Figure 4.5.

Furthermore, to have a better overview of the potential correlations or relationships between our different feature attributes, we leverage a pair-wise correlation matrix of the feature attribute values and then display this as a **heatmap** in Figure 4.4. A correlation between two variables tells us the degree to which these variables have a linear relationship. Furthermore, it can be utilized as an index of relevance for ranking individual features [43]. Linearly dependent variables have a high correlation value compared to linearly independent variables.

The correlation is calculated using the Pearson Correlation Coefficient. The formula to calculate the pearson correlation coefficient between two variables X and Y is provided below:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}$$

which boils down to:

$$r = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2 (y_i - \overline{y})^2}}$$

Using this coefficient, a value of 1 represents a strong positive correlation relationship, -1 represents a strong negative correlation relationship and 0 indicates the absence of any relationship whatsoever between variables. The correlation between the features in our feature set is quite important because features which are highly correlated with each other are then linearly dependent on each other, which insinuates that they have the same effect on the dependent variable. If this is the case, having both features in the feature set is redundant and does not help in the clustering process, meaning that one of the two features can be dropped in order to gain performance improvement without degrading results.

From the Figure 4.4, we observe that the heatmap has varying gradients based on the strength of the correlation between the feature sets. We visually observe that none of the features have a high correlation with each other, with the highest correlations having a value of 0.48 between count and PCR, followed by 0.36 between $src\_bytes$ and $SB/C$. In this case, none of the features from our feature set needs to be dropped. Features count and DB /DP are further discarded.

For more visualizations of the correlations between the features in our feature set, we make use of **pair-wise scatter plots**. These pairwise plots per feature are shown in Figure 4.5. The scatterplots show patterns between two features. From the scatterplots, we notice that the SB/SP and PCR features exhibit a positive relationship or correlation meaning an increase in the amount of bytes sent out per packet results in an increase in the PCR value which is exactly what we expect. This can also be seen from the heatmap in Figure 4.4 where these 2 features have a correlation value of 0.28. The DB/DP and PCR on the other hand exhibit a negative relationship or correlation. The pairwise scatter plot can be considered as the graphical version of the heatmap in Figure 4.4.

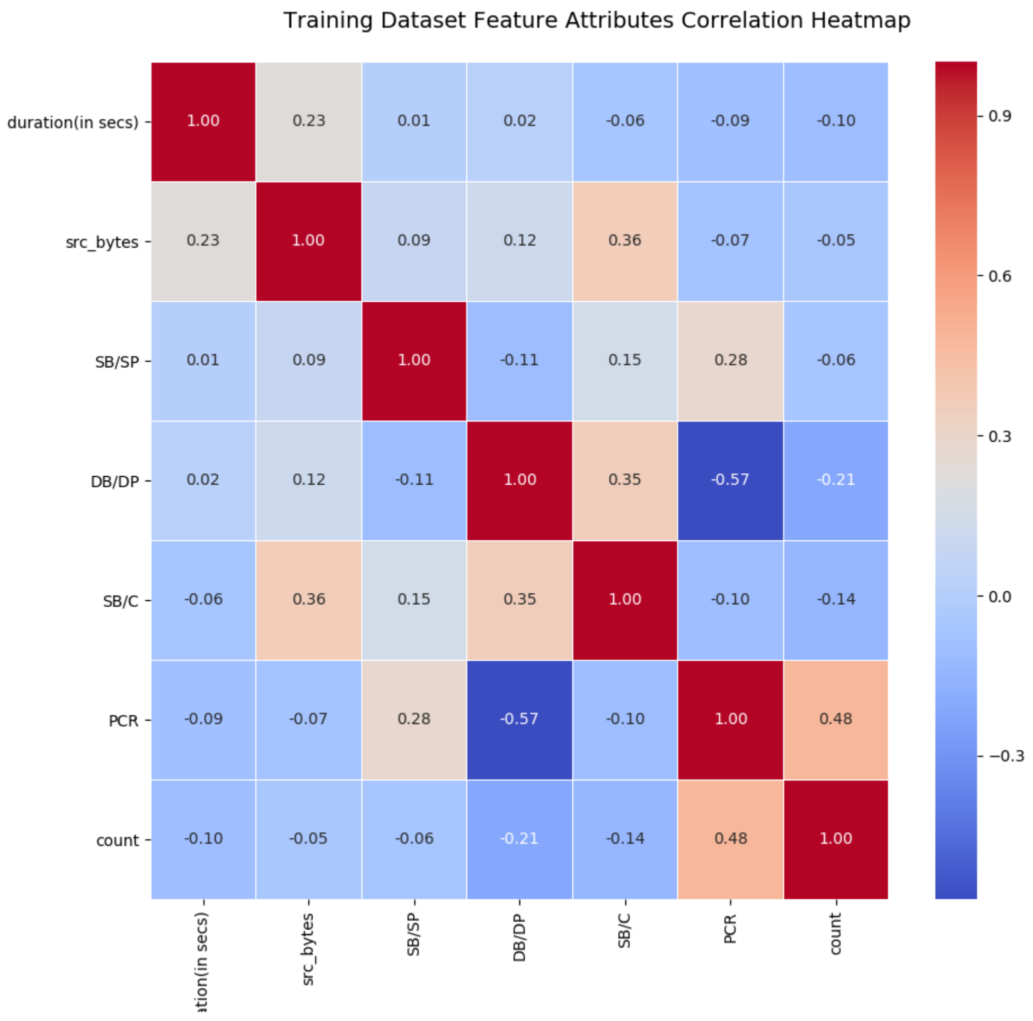


Figure 4.4: Correlation heatmap of the feature attributes for the training datasets

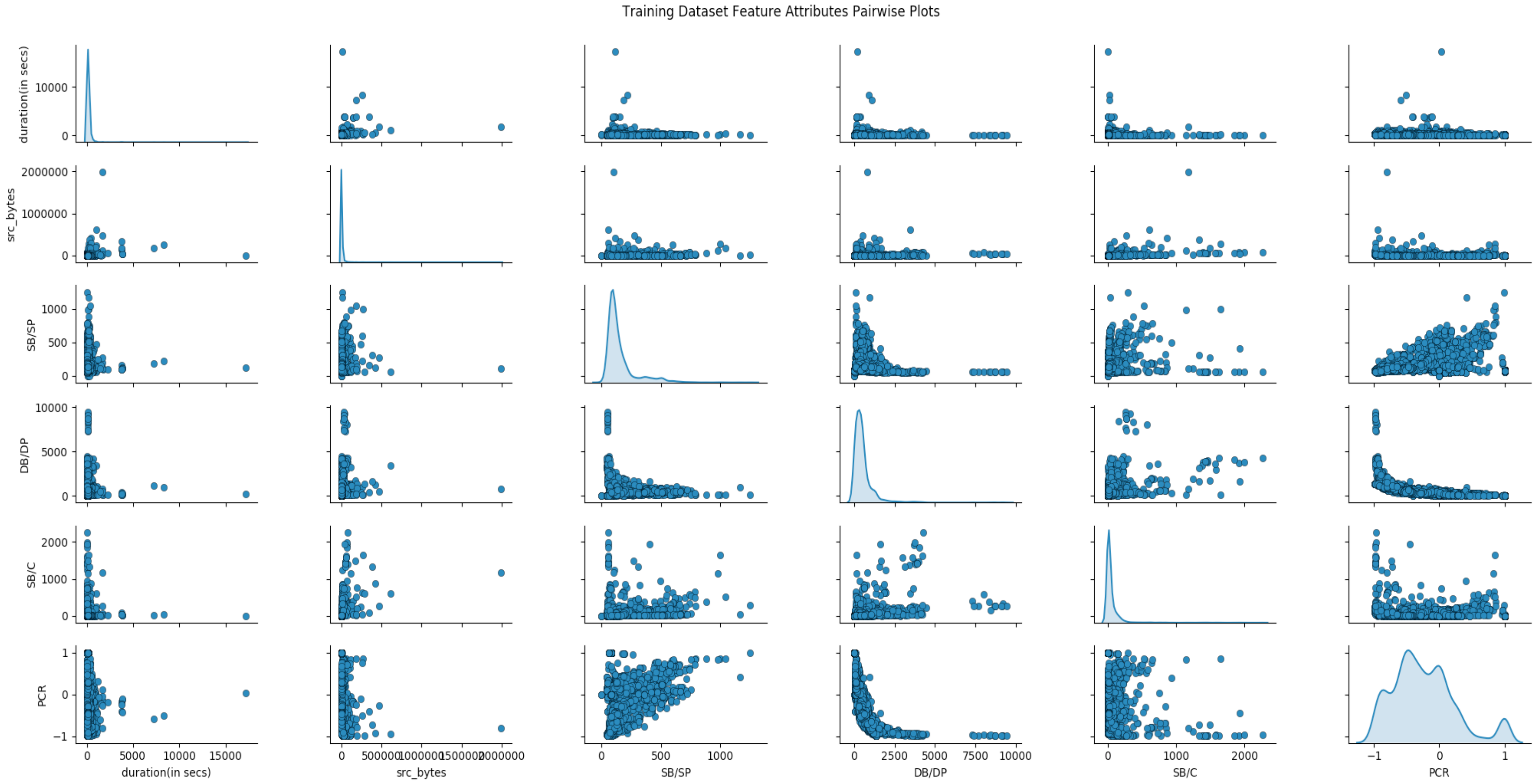


Figure 4.5: Pairwise plot of the features from the training dataset

## 4.3. Evaluation criteria

The goal of this research is the detection of data exfiltration using netflow metadata. More specifically, we are interested in detecting network anomalies which characterize a data exfiltration by using netflows.

Guided by the fact that we aim for not only a low number of false positives but also a low number of false negatives, we test the performance of our model by using the following standard metrics:**recall(R) and precision (P)** with respect to the data exfiltration scenarios present in the different test datasets. The aforementioned metrics are calculated based on a few evaluation criteria: false positives (FP), true positives (TP), true negative (TN) and false negative (FN). A FP in our case would be a benign connection that is detected by our model to be anomaly. A TP is a malicious connection that is indeed detected by our model as an anomaly. Benign connections that indeed get detected to be benign are TNs and a FN is a malicious connection that gets detected to be benign.

We additionally take the **false positive rate (FPR)** and the **false negative rate (FNR)** into account during this research. True positives (TP) are the actual data exfiltration instances which are correctly detected or predicted by our model as data exfiltration instances. False positives on the other hand are the instances that are incorrectly predicted by our model as data exfiltration instances which are actually not data exfiltration instances.True negatives are the non-data exfiltration instances which are correctly predicted as non-data exfiltration and false negatives are those actual data exfiltration instances that are incorrectly predicted by our model as non-data exfiltration instances. A layout is shown in Table 4.5. Once these are calculated, we then fine-tune our model by making some parameter changes and finally evaluatiing the performance of our fine-tuned model on the validation set.

| | | **Actual** | |
|---|---|---|---|
| | | **Positive** | **Negative** |
| **Predicted** | **Positive** | True Positive | False Positive |
| | **Negative** | False Negative | True Negative |

Table 4.5: Standard matrix for the evaluations of data exfiltration attacks

The **recall** defines the ratio of the number of correctly detected data exfiltration instances, or true positives to the total number of instances that are actually data exfiltration. This is expressed as follows:

$$\mathbf{R} = \frac{TP}{TP + FN}$$

The **precision** defines the ratio of correctly detected data exfiltration instances or true positives to the total number of detected data exfiltration instances. This is expressed as:

$$\mathbf{P} = \frac{TP}{TP + FP}$$

The **false positive rate (FPR)** expresses the ratio of the number of instances that are incorrectly predicted to be involved in data exfiltration to the total number of non-data exfiltration instances. This gives us the proportion of connections not involved in data exfiltration but detected by our model to be exfiltrating data. This is calculated as follows:

$$\mathbf{FPR} = \frac{FP}{FP + TN}$$

The **false negative rate (FNR)** expresses the ratio of the number of connections that are involved in data exfiltration but are detected as not being involved in data exfiltration. This gives the proportion of connections involved in data exfiltration and not detected. This is calculated as:

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

The recall, the precision, false positive rate and the true positive rate values give a clear overview of the number of true positives, true negatives, false positives and false negatives which are of importance for this research. These give us a solid basis to make conclusions on the performance of our model.

## 4.4. Summary

This chapter presents the various datasets collected and used in this research. The datasets used were collected at Adyen. One training dataset is collected containing only benign connections. Next, three test datasets are collected and used in this research. The first test dataset was collected with **partial knowledge** of our detection model. The second test dataset was collected with **no knowledge** of our detection model. The third test dataset was collected with **full knowledge** of our detection model. Each of these test datasets include one or more simulated data exfiltration scenarios. Lastly, a validation set was collected.. Furthermore, we present the evaluation criteria used in this research, which are recall, precision, FPR and FNR.

# 5 Clustering Methods

Once we selected and implemented features which can be used to detect data exfiltration using netflow metadata, we then needed to group our connections based on their feature values. In order to achieve this, we select a clustering algorithm out of the multiple clustering algorithms out there.

As a brief summary, we narrowed down the clustering algorithms to be used for this research based on the following criteria:

- The algorithm should be suitable for network traffic anomaly detection in an unsupervised setting.
- The algorithm should be scalable with larger datasets. The algorithm should be computational fast with the creation of clusters for larger datasets containing millions of datapoints or more.
- In order to exhaustively explore the parameter scope of the algorithm for optimal results, the algorithm should have minimal or fewer number of tweakable paramaters.
- The algorithm should produce self explanatory and easy to analyze clusters to domain-specific experts.

The first criteria used is the selection of clustering algorithms that have proven in literature to be suitable for network anomaly detection in an unsupervised setting [6, 14, 68]. In order to have a more specific, data exfiltration orientated and smaller subset of clustering algorithms, we consider the set of clustering algorithms that have been used specifically to detect data exfiltration. Looking at the clustering algorithms used in literature on network anomaly detection [6], we end up with k-means clustering and DBSCAN. Some research on network anomaly detection use some Expectation Maximization (EM) clustering algorithm [68].

A key and very important requirement when selecting the last set of clustering algorithms is that they must be computational fast when run on huge datasets. Adyen collects about 350,000 connections on average every minute on a daily basis, with this number increasing as the number of employees increase. This means about 21,000,000 connections are collected every hour on a daily basis. Our clustering engine must then be able to cluster a huge number of connections with very little computational overhead. As a result of this, we disregard the use of EM clustering algorithms for this research since they are considered to have a slow learning time [33]. With the aforementioned conditions in mind, we then end up with k-means, DBSCAN and HDBSCAN clustering algorithms. Most research in literature doing network traffic classification using clustering algorithms are noticed to make more use of either k-means clustering algorithm [33, 67, 73] and/or DBSCAN [6, 33].

In order to choose what clustering algorithm we will use for this research, we then consider the results shown in [60]. This shows the performance of ten different clustering algorithms used and implemented in python. It evaluates the performance of the various clustering algorithm on thousands of datapoints and further goes on to analyze the performance of the algorithms when the dataset is scaled to hundreds of thousands of datapoints. As a conclusion, HDBSCAN is considered to be the best clustering algorithm when it comes to the formation of the best clusters but k-means is considered to be the most suitable algorithm for larger datasets because it scales best with larger datasets containing datapoints in a magnitude of hundreds of thousands. Also, the k-means algorithm only needs the number of clusters, k, needed to be predefined, which can be easily found using the elbow curve or some statistical or mathematical method.

While we considered k-means, DBSCAN and HDBSCAN as potential clustering algorithms for this research, the results from the aforementioned criteria lead us to choosing kmeans as the clustering algorithm to be used for this research. We made use of scikit-learn's k-means algorithm.

## 5.1. K-means

K-means is a centroid-based clustering algorithm which groups datapoints into k different cluster, with datapoints in the same cluster having similar features and k being a predefined positive number. A summary of the steps taken by the k-means clustering algorithm is as follows:

1. Define the number of clusters to be formed, K.
2. Initialize the cluster centroids for your k clusters. This can be done by picking arbitrary points in the dataset as the cluster centroid, ensuring that the centroids are all different from each other or by some other method.
3. Iterate through all the datapoints in dataset and calculate their distances from all the cluster centroids.
4. Assign each datapoint to the cluster with the closest centroid or smallest centroid distance.
5. Recalculate the centroids of newly formed clusters.
6. Repeat step 3 and 4 until the centroids no longer change.

When a new dataset is introduced with new datapoints that need to be clustered based on the trained model, the k-means predict method then computes the Euclidean distance between a new datapoint's feature vector values and that of the predetermined cluster centroid. That new datapoint is then assigned to the cluster with the nearest cluster centroid.

First and foremost, in order to use k-means, we need to specify the number of clusters as explained above. There are some methods which exists for this particular purpose. These are:

- Visual Method: Elbow Curve
- Mathematical Method: Silhouette Coefficient
- Experimentations, visualization and interpretation

### 5.1.1. Elbow Curve

In this Method, we have a plot of the number of clusters against the Within Cluster Sum of Squared Errors (WCSSE) or simply referred to as the Sum of Squared Errors (SSE) which is the sum of all squared distances of every datapoint in our dataset to the center of its cluster. Once this plot is drawn, we identify the "elbow" point from the plot. This elbow then represents an optimal number of clusters with respect to the intra-cluster variations which can be used for further data analysis.

From such plots, it is generally recommended to choose either the point on the elbow as optimal number of clusters or the next point. Note that this is just a recommendation and all other values should be tested as well to see which creates the best cluster. Although the value from the elbow plot is not completely accurately, it gives a clear indication of what we can consider good enough to work with. From Figure 5.1, we observe an elbow between k = 3 and 4.

### 5.1.2. Silhouette coefficient

The **silhouette coefficient** is a mathematical or statistical method which can be used to determine the optimal number of clusters needed for k-means clustering and also measure or quantify the separation between clusters[84]. It has two main properties which are cohesion and separation. Cohesion entails telling how similar objects are to their own clusters, while separation deals with how objects are dissimilar to objects of other clusters. This score has a range of [-1, +1] with a high positive value indicating that an object is well matched to its own cluster and not well matched to other clusters and a low value indicating that an object is poorly matched to its own cluster.

The silhouette measures for each point p, the average distance between point p and all other points in the same cluster. It then goes on to find the average distance between point p and all the other points in a different cluster other than its own. The difference between these average distances divided by the maximum

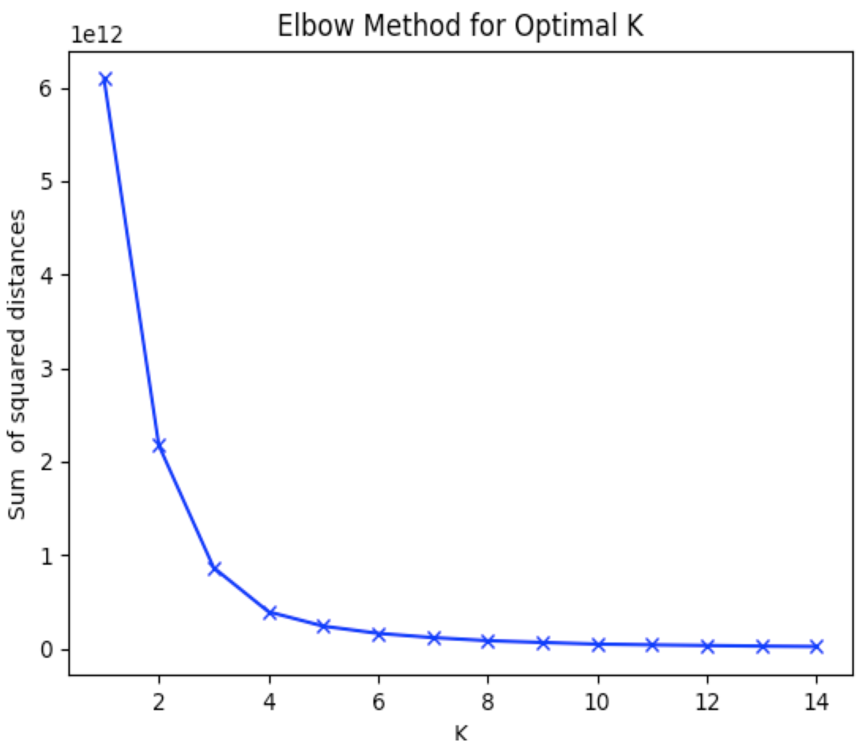


Figure 5.1: Elbow Plot of k-means clustering for k ∈ [1, 15] on training dataset

of both averages is the silhouette score. Assume i is the center of a centroid, s(i) is the silhouette coefficient of that centroid, a(i) is the average distance between i and all the other datapoints within the same cluster (cohesion), b(i) is the lowest average distance of i to all points in any other cluster for which i is not a member of (separation). The silhouette coefficient is then calculated using the formula:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

which can also be rewritten as:

$$\text{s(i)}=\begin{cases} 1-\dfrac{a(i)}{b(i)}, & \text{if } a(i) < b(i). \\ 0, & \text{if a(i) = b(i) .} \\ \dfrac{b(i)}{a(i)}-1, & \text{if } a(i) > b(i) \end{cases}$$

Using this formula, the silhouette values or scores pertaining to different clusters with their corresponding cluster number in the range [2, 14] are shown in Table 5.1 for our training dataset. From these values, we plot the graphical silhouette representation, which is shown in Figure 5.2 on a larger range of [2, 20]. The silhouette plot shows that the silhouette coefficient was highest when k=2, suggesting that this is the optimal number of clusters.

| **Cluster** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **silhouette** | 0.99 | 0.96 | 0.92 | 0.90 | 0.88 | 0.80 | 0.75 | 0.75 | 0.69 | 0.56 | 0.57 | 0.57 | 0.54 |

Table 5.1: All features Silhouette coefficients of k-means clustering for k ∈ [2, 14] on training dataset

The results from the silhouette coefficients in Figure 5.2 and the elbow curve in Figure 5.1 show a clear mismatch in the optimal number of clusters needed for clustering. While 4 is the optimum chosen by the elbow curve , the silhouette score finds 2. We discard the results from the silhouette after a visual analysis

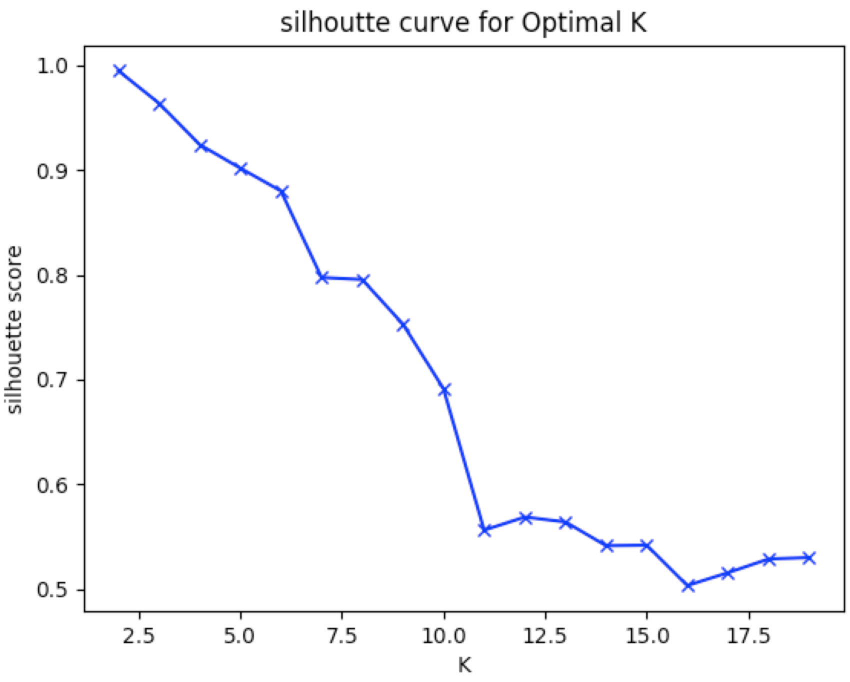


Figure 5.2: Silhouette Plot of k-means clustering for k ∈ [2, 20] on training dataset

is done. As a further means to find out the best solution for our case, we try out our clustering algorithm with the different cluster numbers and observe the characteristics of the cluster formed. Consequently, we observe that our algorithm performs better with 4 clusters based on the thickness of the clusters and the distinct characteristics of every cluster.

### 5.1.3. Visual Analysis

This is a manual technique which involves inspecting individual data points in a cluster and observing if their cluster allocation is the best. In this case, visualizations such as scatter plots and heat maps are used to ease the creation of visual clusters which can then be used to analyze if a data point fits well within its allocated cluster. The use of visual analyses should not be used for the validation of clusters, but should be used to check if a data point fits well within its allocated cluster. Using this technique to determine the number of optimal clusters could be time consuming and would require a very good understanding of the data.

### 5.1.4. k-means cluster analysis and evaluation

| Cluster | duration | Src_bytes | SB/SP | DB/DP | SB/C | PCR |
|---|---|---|---|---|---|---|
| 0 | 16.04 | 1.119909e+06 | 271.7 | 358.88 | 12818.9 | -0.08 |
| 1 | 103.79 | 7.849023e+03 | 113.5 | 1099.49 | 102.7 | -0.6 |
| 2 | 108.43 | 9.281875e+02 | 87.3 | 54.23 | 10.7 | 0.8 |
| 3 | 156.88 | 1.547767e+04 | 104.2 | 3372.5 | 96.7 | -0.9 |

Table 5.2: Mean feature values of all connections in the first test data grouped per cluster

Once the clusters were created, we aggregated the mean of all features in the training data, as shown in Table 5.3. To view the difference in the feature values per connections in different clusters, we visualized the **parallel coordinates** plot of the feature values per cluster. This result is shown in Figure 5.3. The parallel coordinates plot is utilized to visualize multivariate data for multiple attributes all together.

From the visualization shown in Figure 5.3, the connected line segments represent points in our dataset. A single line segment, therefore, pertains to a single connection in our dataset. Each vertical line represents a feature in our feature set. A set of line segments all of the same color that traverse all features are the connections from a single cluster. By having a closer look, we clearly observe that **duration**, **SB/SP** and **PCR**

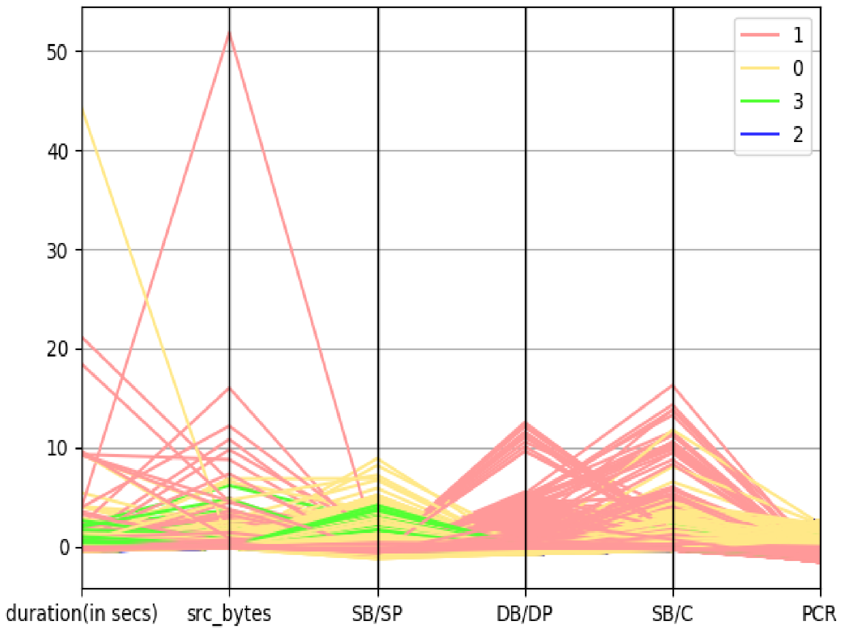


Figure 5.3: Parallel coordinates plot of feature values per connections in the training dataset per cluster

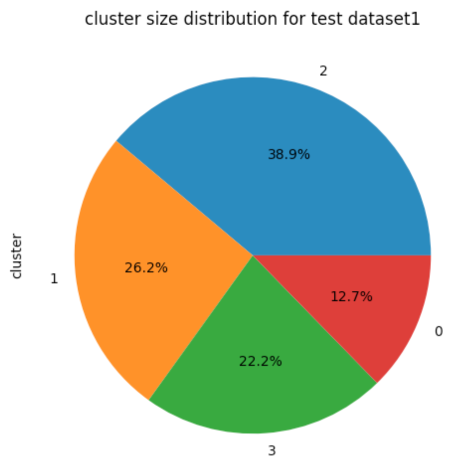
are slightly higher for cluster 0 in general than the other clusters even though cluster 1 has a few connections with higher duration. **Src_bytes**, **DB/DP** and **SB/C** are on the other hand higher for cluster 1 compared to the rest of the clusters. In order to validate this statement, we look at the values provided in Table 5.3, which contains the mean values of features in our feature set per cluster.

| Cluster | Duration | Src_bytes | SB/SP | DB/DP | SB/C | PCR |
|---|---|---|---|---|---|---|
| 0 | 194.5 | 6763.7 | 218.7 | 223.3 | 58.9 | 0.15 |
| 1 | 212.5 | 9262.2 | 116.5 | 847.1 | 68.2 | -0.6 |
| 2 | 30.8 | 92.16 | 84 | 32.6 | 3.85 | 0.61 |
| 3 | 164.13 | 4604.34 | 193.87 | 349.55 | 26.7 | -0.2 |

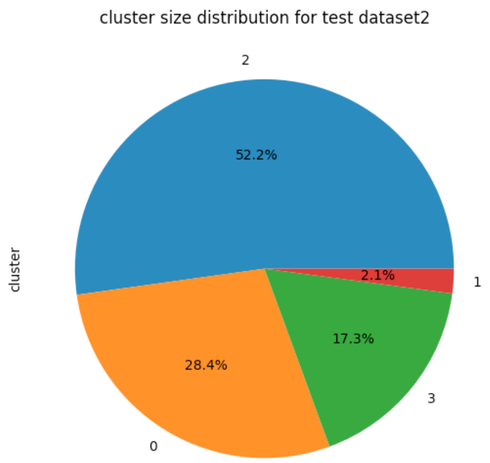
Table 5.3: Mean feature values of all connections in training dataset per cluster

Table 5.4 shows the number of connections found in each cluster per test dataset. The smallest cluster is the cluster with the least number of connections compared to the others. For test set 1, cluster 0 is the smallest cluster with 16 connections, test set 2 has cluster 1 as smallest cluster with just 667 connections, test set 3 has cluster 1 as smallest cluster with 1911 connections. The graphical overview is shown in figures 5.4 and 5.5. From these observations, it is quite important to notice that the clusters in all our different test sets are of different sizes. From this observation, we choose how to detect anomalies in our different test datasets using the following two of the three assumptions used when detecting anomalies by clustering [4, 6].

- In the case where clustering results to clusters of different sizes, the smallest cluster can be considered abnormal or anomalous and the thicker clusters normal.
- In the case where clustering results to clusters containing both benign and anomalous datapoints, the anomalous datapoints are located at a distance far away from the cluster centroid while the normal datapoints are located close to the cluster centroid. In this case, a distance measure can be used to detect anomalies.

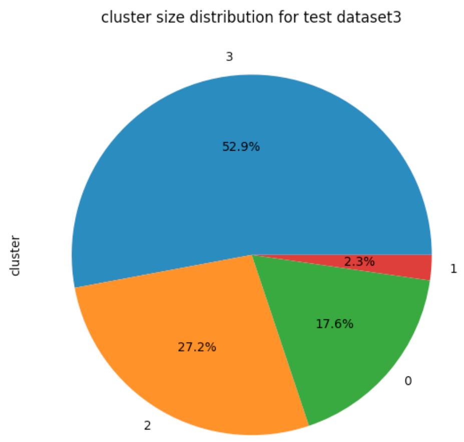


(a) Cluster size distribution for test dataset1

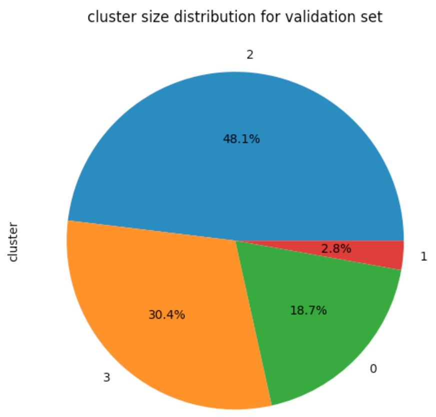


(b) Cluster size distribution for test dataset2

Figure 5.4: Cluster size distribution for different test datasets 1 and test datasets 2



(a) Cluster size distribution for test dataset3



(b) Cluster size distribution for validation dataset

Figure 5.5: Cluster size distribution for test dataset 3 and the validation dataset.

| **Cluster** | **Test dataset 1** | **Test dataset 2** | **Test dataset 3** | **Validation dataset** |
|---|---|---|---|---|
| 0 | 16 | 8945 | 14761 | 13945 |
| 1 | 33 | 667 | 1911 | 2078 |
| 2 | 49 | 1647 | 2286 | 35797 |
| 3 | 28 | 5458 | 44514 | 22617 |
| **Total** | 126 | 31542 | 84050 | 74437 |

Table 5.4: Cluster composition of the different test sets.

## 5.2. Summary

This chapter introduces the criteria used to select an appropriate clustering algorithm for this research. The clustering algorithm should be suitable for network traffic anomaly detection and be scalable with larger datasets. Additionally, in order to exhaustively explore its parameter space for optimal results, the clustering algorithm should have minimal parameters and it should produce self explanatory clusters to domain specific experts. With the aforementioned criteria, we select k-means as clustering algorithm and based on the results from the elbow method, 4 is used as predefined number of clusters. We further analyze and evaluate the characteristics of the clusters formed and their sizes. We observe that the formed clusters have different characteristics and are of different sizes in terms of number of connections they contain.

# 6 Anomaly Scoring

In Chapter 5, we made a few observations about the composition of the clusters obtained from our clustering algorithm. One of the observations, which holds for all the test datasets used in this research is that the clusters formed per dataset are all of different sizes. Based on this, we choose how to detect the simulated anomalous connections in our different test datasets using two of the three applicable assumptions used in recent literature when detecting anomalies by clustering. These assumptions are [4, 6]:

- In the case where clustering results to clusters of different sizes, the smallest cluster can be considered abnormal or anomalous and the thicker clusters normal.
- In the case where clustering results to clusters containing both benign and anomalous datapoints, the anomalous datapoints are located at a distance far away from the cluster centroid while the normal datapoints are located close to the cluster centroid. In this case, a distance measure can be used to detect anomalies.

The third assumption used in recent literature to detect anomalies after clustering, which is discarded in this research is as follows:

- : If clusters are created of only normal data, any new datapoints that do not fit well with these predetermined clusters are considered anomalous.[6, 34]

This assumption is not used because it is mostly applicable in the case where density-based clustering algorithms such as DBSCAN explained in Section 2.1.2 are used. These algorithms have a concept of noise where any datapoint that doesn't fit within a cluster is considered noise and in this case, will be considered anomalous. This research uses k-means which does not have the concept of noise as it fits all datapoints into one of the predefined clusters, therefore, resulting in the discarding of this assumption.

Once connections have been clustered and their distances from their respective cluster centroid derived, the next steps involve allocating an anomaly score to every connection. In this section, we explain and motivate the methods and techniques used to allocate anomaly scores to every connection. For every connection, $c_i$, we allocate three anomaly scores:

1. $S^1(c_i)$: Mean absolute deviation of connection $c_i$'s distance from its respective cluster centroid.
2. $S^2(c_i)$: Local outlier factor of connection $c_i$ with respect to its neighbors.
3. $S^3(c_i)$: Local outlier probability of connection $c_i$ with respect to the feature space of all connections.

The details and motivations behind the computation of the anomaly scores are presented in sections 6.1 , 6.2 and 6.3, with Section 6.2.1 containing an example case of how the LOF score is calculated for four 2-dimensional datapoints. Section 6.4 further discusses how we combine the three scores to compute the final anomaly score of each connection. A key feature here is that we use a combination of both distance-based (Section 6.1) and density-based anomaly detection (sections 6.3, 6.2 ) techniques to score our connections which eliminates the bias of using just one of both techniques. In this way, rather than focusing solely on anomalous connections based on distances calculations, we also monitor the density of the region in which that connection finds itself. A similar approach has been adopted in [1, 64].

## 6.1. Mean absolute deviation of the distance from cluster centroid Score (MAD)

Firstly, we compute a score $s^1$ for every connection that is dependent on the position of the connection in the cluster with respect to other connections in the same cluster. When connections are clustered, the clusters are composed of both benign and anomalous connections, implying that the main aim of this score is to deternine if a connection is located in an anomalous region of its cluster space. According to work carried out in [4, 6], it is mentioned that for clusters containing both benign and anomalous data points, the normal data points mostly reside close to the cluster centroid while the anomalous datapoints are located far away from their cluster centroids. This assumption is used in this research to score our clustered netflow connections based on their distances from their respective cluster centroids. **Clustering is a prerequisite for the calculation of this score as the distances from the cluster centroids are needed**.

Let us consider **X** to be the set of all connections gotten from the flow traffic logged on the netflow collector, that is:

$$\mathbf{X} = \{x_i \,:\, i \in [1,n]\,\}$$

where n is the total number of connections in the set X.

With the above connections, we then cluster them using the k-means clustering algorithm. The algorithm takes a dataset as inputs where the dataset contains a collection of features for each connection in set X and requires a a predefined k which is the number of clusters expected. As explained in Section 5.1, the algorithm starts with initial estimates for the k cluster centroids which can be randomly selected from the data set or randomly generated, with each centroid defining a cluster. Assume $c_j$ is the collection of centroids in the set C of clusters. Once the cluster centroids, $c_j$, which represent different clusters have been found, the connections, $x_i \in$ X are clustered based on their distances from the various cluster centroids. Connections are assigned to the nearest cluster centroid. This distance is calculated using the squared of the Euclidean distance described in Section 2.1. Formally represented, each connection $x_i$ is initially assigned to a cluster according to the following formula:

$$c(x_i) = \mathbf{argmin}_{c_j \in C}\, d(c_j, x_i)^2$$

where d is the is the Euclidean distance

Once all $x_i \in$ X are initially assigned to a cluster, the cluster centroids get updated until the optimal centroids are found. This is only achieved when the stopping criteria is met, which happens when either no change of clusters is experienced for data points, some maximum number of iterations is achieved or the sum of squared euclidean distances is minimized. The cluster centroids are updated using the following formula:

$$c_j = \frac{1}{|P_j|} \sum_{x_i \in P_j} x_i$$

where $P_j$ represents the set of connection assignments for each $j^{th}$ cluster centroid and $|P_j|$ represents the number of connections in the $j^{th}$ cluster.

As earlier mentioned, one of the stop conditions for the algorithm is when the sum of squared euclidean distances is minimized after updating the centroids $c_j$ for the $j^{th}$ cluster. The minimum value is calculated as follows:

$$\mathbf{minimum}\left(\left(\sqrt{\sum_{j=1}^{k}(c_j^1 - x_i)^2}\right)^2, \cdots, \left(\sqrt{\sum_{j=1}^{k}(c_j^n - x_i)^2}\right)^2\right)$$

where *n is the total number of times the cluster centroid* $c_j$ *is updated.*

which boils down to:

$$\mathbf{minimum}\left(\sum_{j=1}^{k}(c_j^1 - x_i)^2, \cdots, \sum_{j=1}^{k}(c_j^n - x_i)^2\right)$$

Once this minimum value is achieved, the algorithm stops and converges and connections are allocated to the best clusters. We then calculate their distances from their cluster centroids, and further calculate the **robust z-score** of these distances as follows:

$$rz(x) = \frac{d(c(x), x) - median(x)}{MAD(X) \cdot 1.4826}$$

where rz(x) corresponds to the robust z-score of x, $d(c(x), x)$ is the euclidean distance of connection x from its respective cluster centroid and MAD(X) is the mean absolute deviation of the column of cluster distances. This is calculated using the following formula:

$$MAD(X) = \frac{\sum_{i=1}^{n} |x_i - median(X)|}{n}$$

The robust z-scores are normalized using the **min-max scaler technique** described in Section 3.4.1. Since these scores are then normalized to a range of [0,1], we multiply by 100 to get our first anomaly score for every connection with range[0, 100], where 0 represents very normal connections and 100 very anomalous connections. For a connection x, this is represented as follows:

$$S^1(x) = 100 \cdot normalized(rz(x)) = 100 \cdot \left( \frac{rz(x) - min(rz(x))}{max(rz(x)) - min(rz(x))} \right)$$

The z-score can be used to detect outliers[29]. It extracts the mean from the element for which the z-score is to be calculated and divides that by the standard deviation:

$$Z - score(x) = \frac{|x - \mu(x)|}{\sigma(x)}$$

A primary concern with the z-score is that it works well for normally distributed data, making use of the mean and the standard deviation which are not robust to outliers[29]. Due to the fact that our data is skewed (mostly rightly skewed), as can be seen in figure 6.1 for our training data, we make use of a more robust z-score which on the other hand, uses the median and mean absolute deviation which are statistically more robust to outliers compared to the mean and standard deviation used in the z-score [29].

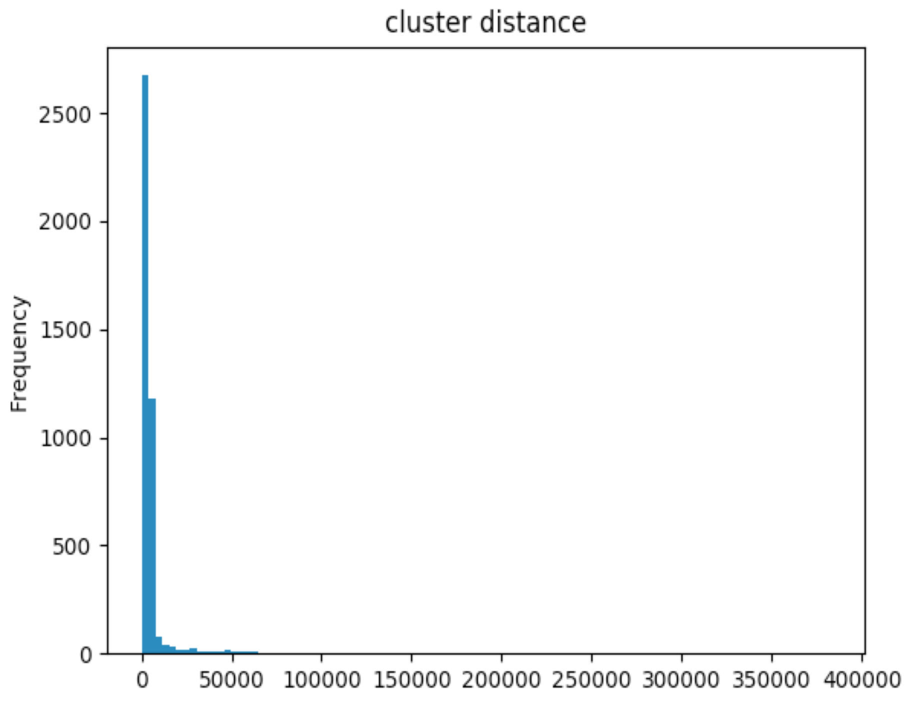


Figure 6.1: Histogram of the distances from cluster centroids of connections in the training dataset

## 6.2. Local Outlier Factor Score (LOF)

The local outlier factor introduced in [19] is a well known and extensively used anomaly detection algorithm that uses the idea of local anomalies. Due to its practicability and applicability, this concept is now adopted in several nearest-neighbors based algorithms. For this research, the main purpose of this score is to identify anomalous connections based on their local densities in the feature space. The principal intuition is that the local density of anomalous connections is significantly different from the local density of its neighbors. Anomalous network connections will be found in sparser regions of lower density compared to normal network connections which will be found in more dense regions. In order to achieve this, we will compare the relative density of every network connection to that of its neighbors as a measure of how much of an outlier that connection is.

A big advantage of the LOF algorithm, which also is one of the reasons why it is been used in this research is that the algorithm detects outliers irrespective of the data distribution [19]. This is because no assumptions are made concerning the distribution of the data been used as data are assumed to be generated independently from the same probability distribution. Another major reason why this algorithm is used is because of its extensive use in the detection of anomalies in NetFlow traffic [10, 79, 96]. Additionally, the LOF algorithm is an unsupervised outlier detection method, which makes it perfect for use with network traffic, which are normally not labeled.

The calculation of the LOF score involves three main steps:

1. The first step involves the determination of the k-nearest neighbor for every datapoint. In the case where a tie exists for the k-th nearest neighbor, then all these neighbors are considered in the calculation. The k has to be predefined and is a very important feature of this algorithm. Choosing a wrong k value could lead to completely different and wrong results. An explanation on how to choose an appropriate k-value is provided in [19].

2. Once, the k-nearest neighbors have been found, the local reachability density for every datapoint is then calculated as an estimate of the datapoint's local density.

3. The final step involves the calculation of the LOF score as a comparison of the datapoint's local reachability density with that of its k neighbors.

Once the LOF scores have been computed, they can be interpreted as follows:

- A datapoint with a LOF score approximately equal to 1 has almost the same local reachability density as its neighbors, which means almost similar densities as its neighbors. This datapoint is considered to be an inlier.

- A datapoint with a LOF score greater than 1 is found in a less dense region compared to its neighbors. This means that it has a higher reachability distance compared to its neighbors. This datapoint is an outlier.

- A datapoint with a LOF score less than 1 is found in a denser region compared to its neighbors. This means that it has a lower reachability distance compared its neighbors. This datapoint is an inlier.

Applying this to our case, for a network connection x, we first of all need to calculate the k-distance for connection x, denoted as , $d_k(x)$ which is the distance between connection x and its k-th nearest neighbors. Additionally, to know how relatively dense connection x is to its neighborhood, we need to calculate the k-distance neighborhood of x, denoted as $N_k(x)$. This is represented as:

$$N_k(x) = \{x^{'} \mid x^{'} \in D, d(x, x^{'}) \leq d_k(x)\}$$

where D is the entire dataset, considering the fact that we are looking at it locally and $d(x, x^{'})$ is the euclidean distance or some other distance metric between points x and $x^{'}$.

Furthermore, the reachability distance between connections $x^{'}$ and x need to be calculated in order to get the local reachability density(lrd) which is a measure of how close or reachable a connection is from its neighbors as follows:

$$lrd_k(x) = \frac{||N_k(x)||}{\sum_{x' \in N_k(x)} rd_k(x, x')}$$

where k is user-specified and $rd_k(x, x')$ is the reachability distance from connection $x'$ to x, calculated as follows:

$$rd_k(x, x') = max\{d_k(x), d(x, x')\}$$

Once this is calculated, we can then calculate $LOF(x)$ for connection x as follows:

$$LOF(x) = \frac{\sum_{x' \in N_k(x)} \frac{lrd_k(x')}{lrd_k(x)}}{||N_k(x)||}$$

which if broken down, boils down to:

$$LOF(x) = \sum_{x' \in N_k(x)} lrd_k(x') \cdot \sum_{x' \in N_k(x)} rd_k(x, x')$$

As already explained, the higher the local reachability density of the k-nearest neighbor of connection x, the lower its local reachabilty, the higher its LOF. The higher the local reachability of connection x, then the lower the local reachability of the k-nearest neighbor of x and the lower its LOF. This tells us that connections with a higher density compared to its k-nearest neighbors tends to be considered as more normal while those with a more sparse density compared to that of its k-nearest neighbors tends to be considered more as an outlier.

Once these LOF scores are gotten, we then normalize them using the **min-max scaler** technique and multiply these normalized scores by 100 to get $S^2$ as follows:

$$S^2(x) = 100 \cdot normalized(LOF(x))$$

### 6.2.1. Choosing an appropriate number of nearest neighbors, k

As earlier explained, a very important parameter of the LOF algorithm which needs to be predefined is, the number of nearest neighbors, k. As a reminder, the LOF algorithm compares the density of each datapoint to that of its k-nearest neighbors. In [19], a method is recommended for choosing an appropriate k-value. It is recommended to choose a minimum k and a maximum k, and then for each datapoint choosing the maximum LOF value for each k in that range. They recommend having a minimum k of atleast 10. In [10], a k-value is recommended equal to the square root of the number of datapoints present in the training dataset.

For this research, we will select a k-value at which the algorithm detects the highest number of attacks, with the least number of false positives. As shown in sections 4.1.2 and 4.1.2, test set 1 contains two anomalous connections in a total of 126 network connections. Test set 2 contains 21 anomalous connections in a total of 31542 connections. With these values, we then run the LOF algorithms with different k-values starting from 10. The value of the k parameter was incremented by 10 from 10 to 100 and from there on by 50 from 100 to 350 (half the number of flows in the smallest cluster). We then for each k-value return the top 2 and top 21 connections with the largest LOF scores for test set 1 and test set 2 respectively. This allows us to evaluate what k-value detects the highest number of attacks, while having the least number of false positives and the least number of false negatives. The results are shown in figure 6.2

We notice under the feature set used that the recall, precision and FNR are very dependent on the choice of the k parameter. As an example, in the interval $20 < k \leq 69$, the recall and precision values are changing from 0.52 to 0.95, the FNR also fluctuates between 0.047 to 1.0. We also notice that no anomalous connections were detected at all for some values of k ($k \leq 10$, $30 < k \leq 49$, $70 < k \leq 99$). At these values of the k parameter, all connections got a LOF score of 1 meaning all connections were considered to be inliers, and all top 21 detected connections were false positives. From the results shown above, we get the highest precision, recall and the lowest FNR at intervals $50 < k < 69$ and at $k > 100$. We, therefore, choose a value of $k = 69$ for use in this research.

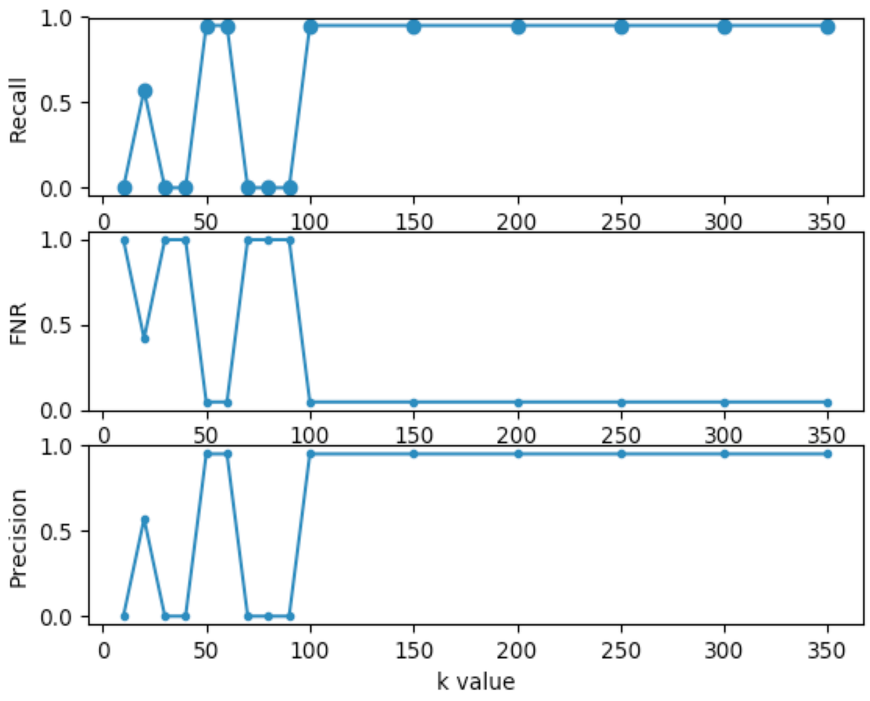


Figure 6.2: Recall, precision and FNR for different LOF k parameter values on test set 2 for the top 21 connections with largest LOF Score

## LOF score calculation Example

In order to show a simple demonstration of how the LOF score is calculated, let us consider the following datapoints:

m(0,0), n(2,3), p(5, 8), q(1,4)

For the sake of simplicity, we use 2-dimensional datapoints. Using the Manhattan distance described in Section 2.1.1, let us calculate the LOF score for every datapoint using the 3 steps provided above for a k value of 2.

**Step 1: Involves the determination of the k-nearest neighbor for all our datapoints.**
Since k is provided to be 2, we then need to calculate the second nearest neighbor for all our datapoints. In order to do this, we need to calculate the Manhattan distance between each datapoint to every other datapoint in our dataset, D. Recall, the formula for the Manhattan distance between points p and q in an n-dimensional dataspace is:

$$MD(p,q) = \sum_{i=1}^{n}(|p_i - q_i|)$$

$$\begin{aligned}
MD(m,n) &= |0-2| + |0-3| = 2+3 = 5 \\
MD(m,p) &= |0-5| + |0-8| = 5+8 = 13 \\
MD(m,q) &= |0-1| + |0-4| = 1+4 = 5 \\
MD(n,p) &= |2-5| + |3-8| = 3+5 = 8 \\
MD(n,q) &= |2-1| + |3-4| = 1+1 = 2 \\
MD(p,q) &= |5-1| + |8-4| = 4+4 = 8
\end{aligned}$$

Using the above distances, we can now find the $d_k(x)$: distance between datapoint x and its k-th nearest neighbor for every datapoint as follows:

$$
\begin{aligned}
d_2(m) &= MD(m,n) = 5\ (n\ and\ q\ are\ the\ second\ nearest\ neighbor)\\
d_2(n) &= MD(n,m) = 5\ (m\ is\ the\ second\ nearest\ neighbor)\\
d_2(p) &= MD(p,n) = 8\ (n\ and\ q\ are\ the\ second\ nearest\ neighbor)\\
d_2(q) &= MD(q,m) = 5\ (m\ is\ the\ second\ nearest\ neighbor)
\end{aligned}
$$

**Step 2: Calculate the local reachability density for every datapoint**

In order to calculate the local reachability density for every datapoint, we first of all need to find the k-distance neighborhood of all datapoints, denoted as $N_k(x)$ using the formula:

$$N_k(x) = \{x^{'} \mid x^{'} \in D, MD(x,x^{'}) \leq d_k(x)\}$$

where D is our dataset. Using the above formula, we get:

$$
\begin{aligned}
N_2(m) &= \{n,q\}\\
N_2(n) &= \{m,q\}\\
N_2(p) &= \{n,q\}\\
N_2(q) &= \{m,n\}
\end{aligned}
$$

Now we then calculate the local reachability density of the points in our dataset using the following formula:

$$lrd_k(x) = \frac{||N_k(x)||}{\sum_{x'\in N_k(x)} max\{d_k(x), MD(x,x^{'})\}}$$

From the above formula, $||N_k(x)||$ refers to the number of elements in the k-distance neighborhood of point x. For our datapoints, these are:

$$
\begin{aligned}
||N_2(m)|| &= ||\{n,q\}|| = 2\\
||N_2(n)|| &= ||\{m,q\}|| = 2\\
||N_2(p)|| &= ||\{n,q\}|| = 2\\
||N_2(q)|| &= ||\{m,n\}|| = 2
\end{aligned}
$$

The local reachability density for our datapoints then becomes:

$$lrd_2(m) = \frac{||N_2(m)||}{\sum_{x'\in N_2(m)} max\{d_2(m),\ MD(m,x^{'})\}}$$

$$
\begin{aligned}
\sum_{x^{'}\in N_2(m)} max\{d_k(m),\ MD(m,x^{'})\} &= max\{d_2(m),\ MD(m,n)\} + max\{d_2(m),\ MD(m,q)\}\\
max\{d_2(m),\ MD(m,n)\} &= 5\\
max\{d_2(m),\ MD(m,q)\} &= 5
\end{aligned}
$$

$$lrd_2(m) = \frac{2}{5+5} = \frac{1}{5} = 0.2$$

Using the same approach, we calculate the local reachability densities of the other data points n, p and q.

$$lrd_2(n) = \frac{2}{10} = 0.2 \quad lrd_2(p) = \frac{2}{16} = 0.125 \quad lrd_2(q) = \frac{2}{10} = 0.2$$

**Step3: Calculation of the LOF score as a function of the local reachability density**

Using the formula:

$$LOF(x) = \frac{\sum_{x'\in N_k(x)} \frac{lrd_k(x^{'})}{lrd_k(x)}}{||N_k(x)||}$$

We then calculate the LOF score for the datapoints as follows:

$$LOF(m) = \frac{\frac{lrd_2(n)}{lrd_2(m)} + \frac{lrd_2(q)}{lrd_2(m)}}{||N_2(m)||} = \frac{\frac{0.2}{0.2} + \frac{0.2}{0.2}}{2} = 1$$

$$LOF(n) = \frac{\frac{lrd_2(m)}{lrd_2(n)} + \frac{lrd_2(q)}{lrd_2(n)}}{||N_2(n)||} = \frac{\frac{0.2}{0.2} + \frac{0.2}{0.2}}{2} = 1$$

$$LOF(p) = \frac{\frac{lrd_2(n)}{lrd_2(p)} + \frac{lrd_2(q)}{lrd_2(p)}}{||N_2(p)||} = \frac{\frac{0.2}{0.125} + \frac{0.2}{0.125}}{2} = 1.6$$

$$LOF(q) = \frac{\frac{lrd_2(m)}{lrd_2(q)} + \frac{lrd_2(n)}{lrd_2(q)}}{||N_2(q)||} = \frac{\frac{0.2}{0.2} + \frac{0.2}{0.2}}{2} = 1$$

We then normalize the LOF scores for our datapoints using the **min-max scaler technique** described in Section 3.4.1 and multiplying these scores by 100 to get our second anomaly score.

## 6.3. Local Outlier Probability Score (LoOP)

The local outlier probability, introduced in [56] is another density-based anomaly detection algorithm that uses the idea of local densities. Other anomaly detection algorithms using the concept of local densities attribute an anomaly score to every datapoint. A major problem with attributing anomaly scores to datapoints is the unclarity around the threshold score used to detect anomalies. The local outlier probability tries to circumvent this problem by attributing an outlier probability in the range [0,1] to every datapoint rather than an outlier score. While this method is similar to the local outlier factor method described in Section 6.2, it uses a different methodology to compute datapoint densities. In this case, it assumes that the distances to the nearest neighbors of a datapoint follow a gaussian distribution. The local outlier probability further assumes a half gaussian distribution because of the fact that distances are positive, therefore ignoring the other negative half. With this assumption, it then calculates the probabilistic set of distances by using the standard distance similar to the standard deviation per datapoint. This probabilistic set of distances is then used as a local density estimation to calculate the local anomaly score, by comparing the ratio of the probabilistic set of distances per datapoint to that of its neighbors. In the final step, a gaussian error function and normalization are applied in order to convert the anomaly score into a probability.

Applying this to our research, assume we have C which is a set of n connections, d is a distance function used to detect outliers, pdist(c, T) is the probabilistic distance of connection c ∈ C to a context set T ⊆ C. The probabilistic distance pdist(c, T) of connection c to T has the following property:

$$\forall\ t \in\ T\ :\ \ \Pr\left[d(c,t)\ \leq pdist(c,T)\right] \geq\ \varphi$$

If we assume that the distances of t ∈ T to connection c have a half gaussian distribution and c is the center of the set T, we can then compute the standard distance of the connections in T to connection o as follows:

$$\sigma(c,T)\ =\ \sqrt{\frac{\sum_{t\in T} d(c,t)^2}{|T|}}$$

Note that the standard distance described above is different from the standard deviation. This is mainly because an assumption is made about T being normally distributed around c but we cannot assume that the standard distances themselves are normally distributed. Based on certain considerations enumerated in [56], the probabilistic set distance of connection c to set T with a significance value of $\lambda$ is defined as:

$$pdist(\lambda, c, T)\ :=\ \lambda\cdot\sigma(c,T).$$

While this probabilistic set distance is used as an estimate for the density around connection c based on set T, $\lambda$ is used as a factor for normalization in order to control the approximation of the densities. It, therefore, in no way has an effect on the ranking of outliers.

In order to locally model outliers probabilistically, two assumptions have to be made:

1. The context set T is assumed to be centered around the connection c.
2. The distances are assumed to mimic the positive tail area of a normal distribution.

With these assumptions taken into consideration, the **Probabilistic Local Outlier Factor**(PLOF) of a connection c with respect to a context set T(c) ⊆ C and a significance value of $\lambda$ is calculated using the formula:

$$PLOF_{\lambda,t(c)} := \frac{pdist(\lambda, c, T(c))}{\mathbb{E}_{t\in T}[pdist(\lambda, t, T(t))]} - 1$$

From the above, we notice that the PLOF value of a connection c calculates the ratio of the probabilistic set distances of c to T (which estimates the density around c based on the context set T(c) ) and the expectation of the probabilistic set of distances of all connections in T (which estimates the densities around the connections in T based on the context set T(t) ). From this, we notice that this ratio is not yet in a normalized form and it is also not a probability. In order to normalize these values in an independently from the data distribution, an aggregate value of all the computed PLOF values is gotten as follows:

$$nPLOF := \lambda \cdot \sqrt{\mathbb{E}[(PLOF)^2]}$$

Assuming a mean value of 0, the nPLOF value calculated above can be considered as some sort of standard deviation around the values of the PLOF. Once the nPLOF and the PLOF values are calculated, the Gaussian Error Function (gerrf) is then applied to obtain the **Local Outlier Probability** (LoOP) of connection c as follows:

$$LoOP_{t(c)} := max\left\{0,\ gerrf\left(\frac{PLOF_{\lambda,t(c)}}{nPLOF \cdot \sqrt{2}}\right)\right\}$$

The returned LoOP value above is then normalized in the range[0,1] with connections having values close to 1 being connections in very sparsely dense regions and connections attributed value 0 are located in dense regions. Consequently, connections with a value close to 1 will be more likely to be outliers. With the above information we then calculate $S^3(c)$.

$$S^3(c) = 100 \cdot LoOP_{t(c)}$$

Our third anomaly score is then in the range of [0,100] with connections having a value close to 100 having a higher chance of being anomalies and connections having a score of O being normal.

## 6.4. Combination of scores

The final step of our anomaly scoring component is the computation of the final scores for each connection established by an internal host and collected by our NetFlow collector. This is done by combining the three anomaly scores described in sections 6.1, 6.2, 6.3.

Several methods are used in research to combine different anomaly scores to a single final score. While some methods linearly combine scores to get a final score [71], others maintain the scores as they are and look at the individual scores in the further phases of their analysis. In [78], the simple average of scores from different outlier detection techniques is found to give a constant good performance. [57] also uses the average score as final score when combining scores from different outlier methods. Since we do apply different outlier detection techniques in this research, we adopt the average when combining our anomaly scores. We are also interested in any combination of the three scores that could potentially lead to a better performance of our model. As such, we also attribute for a connection c, with three anomaly scores $S^1(c)$, $S^2(c)$, $S^3(c)$, the following scores:

$$S_{MAD\ and\ LOF}(c) = \frac{S^1(c) + S^2(c)}{2}$$

$$S_{MAD\ and\ LOOP}(c) = \frac{S^1(c) + S^3(c)}{2}$$

$$S_{LOF\ and\ LOOP}(c) = \frac{S^2(c) + S^3(c)}{2}$$

$$S_{MAD,\ LOF\ and\ LoOP}(c) = \frac{\sum_{i=1}^{3}(S^i(c))}{3}$$

We then end up with every connection having four additional scores which are the averages of a combination of scores. Since all three anomaly scores are in the range of [0,100], we end up with these four extra scores in the same range of [0,100] with a score closer to 100 being more anomalous and that closer to 0 being more benign or normal.

In the final step, we evaluate the simulated data exfiltration detection performance of our model at different threshold ranges when each of the seven scores $S^1$, $S^2$, $S^3$, $S_{MAD\ and\ LOF}$, $S_{MAD\ and\ LOOP}$, $S_{LOF\ and\ LOOP}$ , $S_{MAD,\ LOF\ and\ LoOP}$ is used as anomaly scoring metric.

## 6.5. Summary

In this chapter, we present the algorithms used to allocate anomaly scores to the connections in our datasets. We use a distance and statistical based anomaly detection approach to allocate the first anomaly scores to connections based on the robust z-score of their distances from their respective cluster centroids. The main hypothesis is that anomalous connections will be located far away from their cluster centroids while normal connections will be located very close to the cluster centroid.

In order to eliminate the bias of a single technique, we also utilize a density-based approach to score connections. We use the local outlier factor (LOF) and local outlier probability (LoOP) to allocate the second and third anomaly scores to connections, which both use the idea of local anomalies to score connections. The main idea is that normal connections are located in denser regions compared to their neighbors while anomalous connections will, on the other hand be located in very sparse regions. To further simplify the understanding of how the LOF works, an example is shown of its calculation for four 2-dimensional datapoints. For their computation, these algorithms require the number of nearest neighbors, k, to be predefined. Based on the recall, precision, FNR and FPR values obtained for different k values, we choose and use k=69 in this research. Additionally, we explain and show how these three main scores are linearly combined by averaging to obtain other scores.

# 7
# Results and discussion

This chapter evaluates and presents the performance of our model in detecting simulated data exfiltration scenarios using netflow metadata and is divided into three main parts. The first part, Section 7.1, presents the detection performance of our model on the entire test datasets. The second part, Section 7.2, presents and discusses the detection performance on the smallest cluster of every dataset. The results from these evaluations are subsequently used to determine the best anomaly scoring metric and the threshold range at which it performs best. These, are then used to run our model on the validation set and the results presented in the last section, Section 7.4.

We score connections using the MAD (explained in Section 6.1), the LOF (explained in Section 6.2) and the LoOP (explained in Section 6.3) individually. The detection capability of our model for various threshold ranges is presented starting from the range [20,30] and incrementing this by 10 until we reach the maximum threshold range of [90,100]. Additionally, we then combine these scores as presented in Section 6.4 to eliminate the bias of using a single metric over the other. We further present and discuss the results obtained when the individual and combined scores are used.

**The fact that the results from Sections 7.1 and 7.2 could be overwhelming, we present a brief summary of their findings in Section 7.3.**

## 7.1. Performance on the entire test datasets

For each test dataset we present and discuss the results obtained from running our model with the seven different scoring metrics explained above. The results from test dataset 1 are presented in Section 7.1.1. Sections 7.1.2 and 7.1.3 present the results from test dataset 2 and test dataset 3 respectively. As a reminder, test dataset 1 contains 126 connections including a 2 simulated exfiltration connections. The second test dataset consists of 31542 connections from 4 hosts including a simulated data exfiltration in 21 connections and the third dataset contains 84050 connection including 102 connections in which a data exfiltration is carried out.

### 7.1.1. Performance on Test dataset 1

#### MAD score results

For the first test dataset consisting of 126 connections with 2 simulated data exfiltrations connections, we first present the results obtained from running our model using the MAD score as the only anomaly scoring metrics. The MAD score in this case is calculated for every connection in this dataset irrespective of its cluster. Since this score is on a range of [0,100], the detection performance of our model for various MAD score ranges are shown in Table 7.1.

Table 7.1 depicts the highest number of false positives at a threshold value of 20 and the lowest number of false positives for any threshold value ≥ 60. For threshold values in the range [20,59], the lowest number of false negatives are observed while the highest number of false negatives are observed at threshold ranges [60,100]. At threshold values in the range [40,59], both data exfiltration connections are detected with very low number of false positives and the least number of false negatives. This therefore results in this threshold range having the highest recall value, an averagely good precision value, low false positive rate value and lowest False negative rates. The threshold range [40,59] is the best for highest detection of data exfiltration

| **MAD Score >=** | **Number of detected events** | **TP** | **FP** | **FN** | **TN** | **Recall** | **Precision** | **FPR** | **FNR** |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 38 | 2 | 36 | 0 | 88 | 1.0 | 0.05 | 0.29 | 0.0 |
| 30 | 5 | 2 | 3 | 0 | 121 | 1.0 | 0.4 | 0.02 | 0.6 |
| 40 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 50 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 60 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.1: Model performance on test dataset 1 for different MAD score ranges only. These scores are normalized per connection on a scale of 0 - 100.

connections with low false positives and lowest false negatives and is recommended for use when using the MAD score only.

### LOF Score results

Table 7.2 shows the performance of our model when it is run with the LOF score as the only anomaly scoring metric for different LOF score ranges.

| **LOF Score >=** | **Number of detected events** | **TP** | **FP** | **FN** | **TN** | **Recall** | **Precision** | **FPR** | **FNR** |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 40 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.2: Model performance on test dataset 1 for different local outlier factor (LOF) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

Table 7.2 shows no false positives for threshold values ≥ 20. This results in a constant false positve rate of 0.0 and precision value of 1.0 for all threshold values greater than 20. The best detection results with highest number of true positives, no false positives and no false negatives are obtained at a threshold value in the range [20, 29]. At threshold values ≥ 30, we observe no change in the number of false positives which stays at 0, a reduction in the number of true positives and an increase in the number of false negatives.This results in higher false negative rates and lower recall. From these observations, a detection threshold value in the range [20, 29] is recommended for use when the LOF score is used as only anomaly scoring metric.

### LoOP Score results

The performance of our model when run with the LoOP as only scoring framework is shown in table 7.3. At threshold value range of [20, 59], 3 connections are detected with all 2 true positive connections been detected, no false negative and 1 false positive, which is considered to be a high number compared to the number of true positives in this dataset. This results in high recall values of 1.0, average precision values of 0.67, low false positive rate value of 0.0081 and a false negative rate value of 0.0.

From Table 7.3, the range of threshold values resulting in optimal number of detected events (2) which are all true positives, no false positives and no false negatives is [60,89]. This results in high recall and precision values of 1.0 and low false positive and false negative rates of 0.0. At threshold values ≥ 90, 1 event is detected which is a true positive but resulting in 1 false negative event. In the case when the LoOP is used as only scoring framework, a detection threshold of [60, 89] is recommended.

| LoOP Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 30 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 40 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 50 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 60 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 70 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 80 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.3: Model performance on test dataset 1 for different local outlier probability (LoOP) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

### Combination of MAD and LoOP Score results

When the MAD and LoOP scores are combined together, we analyze the detection performance of our model and present this in Table 7.4. We observe optimal results in terms of true positives, false positives and false negatives at a threshold value in the range [60-69]. All true positve events are detected, with no false positives and no false negatives resulting in high recall and precision value of 1.0, and low false positive and false negative rates of 0.0. At a threshold value below the optimal range (in the range [20, 59]), all true positive events are also detected together with a number of false positive events, resulting in decreased precision (from 1.0 to 0.67 for the range [30, 59] and 0.4 for [20, 29]) and increased false positive rate ( from 0.0 to 0.0081 for the range [30, 59] and 0.2 for [20, 29]). Threshold values above the optimal range ([70, 99]) result in an increased false negative rate (from 0.0 to 0.5) and a reduced recall (from 1.0 to 0.5).

| Average of MAD and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 5 | 2 | 3 | 0 | 121 | 1.0 | 0.4 | 0.02 | 0.0 |
| 30 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 40 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 50 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 60 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 70 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.4: Model performance on test dataset 1 for different threshold value ranges when the MAD score and LoOP score are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

When using the MAD and LoOP scores combined together, a threshold value in the range [60-69] is recommended.

### Combination of MAD and LOF Score results

Combining the MAD and LOF scores on the first test datasets leads to the results shown in 7.5. The optimal threshold value is found in the range [30-39], where all true positives are detected, with no false positives and no false negatives, leading to a recall and precision of 1.0 and a false negative and false positive rate of 0.0. At threshold values higher than the optimal threshold range (threshold values ≥ 40), the number of false positives increases and the number of true positives detected decreases. This results in a reduced recall from 1.0 to 0.5 and an increase false negative rate from 0.0 to 0.5. Threshold values lower than the optimal threshold range (in the range[20-29]), all true positives are detected but there is an increase in the number of false positives. This results in an increased false positive rate from 0 to 0.0081 and a decrease in precision from 1.0 to 0.67.

From the above observations, the threshold values in the range [30-39] give the best detection results.

| Average of MAD and LOF >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 30 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 40 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.5: Model performance on test dataset 1 for different threshold value ranges when the MAD score and LOF score are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

#### Combination of LOF and LoOP Score results

The results of the combination of the LOF and LoOP scores are presented in Table 7.6. We clearly observe optimal detection results for threshold values in the range [30-49]. In this case, all true positive events are detected with no false negatives and no false positives. Threshold values above the optimal threshold range lead to a decrease in the number of true positives, which in turn leads to an increase in the number of false negatives. Threshold values below the optimal threshold range on the other hand, result in an increase in the number of false positives although all true positive events are detected.

| Average of LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 30 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 40 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 50 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.6: Model performance on test dataset 1 for different threshold value ranges when the LOF score and LoOP score are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

The results presented above show that threshold values in the range [30, 49] led to the best detection results and should be used if using this scoring metrics.

#### Combination of MAD, LOF and LoOP Score results

When combining all the three scores together, we observe quite different results which are presented in Table 7.7. From this table, we see that the best detection results are obtained at a threshold value in the range [40-69] with 2 true positives, 0 false negatives and 0 false positives, leading to highest recall and precision values of 1.0 and lowest false negative and false positive rates of 0.0. Threshold values in the range [20-39] detect both true positive events but also detect a false positive event resulting to a reduced precision and an increased false positive rate when compared to the threshold values in the best detection threshold range. Threshold values in the range [70-99] miss 1 true positive event leading to an increase in the false positive rate and a decrease in the recall value.

When using these three scores combined, the best threshold values are in the range [40,69].

### 7.1.2. Performance on Test dataset 2

#### MAD score results

Evaluating the performance of running our model with the MAD as only scoring metrics on the entire second test dataset provides the results shown in Table 7.8. While all true positive events are detected at threshold

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 30 | 3 | 2 | 1 | 0 | 123 | 1.0 | 0.67 | 0.0081 | 0.0 |
| 40 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 50 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 60 | 2 | 2 | 0 | 0 | 124 | 1.0 | 1.0 | 0.0 | 0.0 |
| 70 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 124 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.7: Model performance on test dataset 1 for different threshold value ranges when the MAD, LOF and LoOP scores are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

values in the range [20, 59], a number of false positive events are also detected with the number of false positives decreasing as the threshold increases. The optimal threshold range for which our model is observed to perform best is [50,59] as all true positive events are detected, with 8 false positives. This leads to high recall and precision values of 1.0 and 0.72 respectively and low FPR and FNR values of 0.0003 and 0.0 respectively.

| MAD Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 556 | 21 | 535 | 0 | 30986 | 1.0 | 0.04 | 0.017 | 0.0 |
| 30 | 169 | 21 | 148 | 0 | 31373 | 1.0 | 0.12 | 0.0047 | 0.0 |
| 40 | 108 | 21 | 87 | 0 | 31434 | 1.0 | 0.19 | 0.003 | 0.0 |
| 50 | 29 | 21 | 8 | 0 | 31513 | 1.0 | 0.72 | 0.0003 | 0.0 |
| 60 | 5 | 2 | 3 | 19 | 31518 | 0.0952 | 0.4 | 0.0001 | 0.904 |
| 70 | 1 | 1 | 0 | 20 | 31521 | 0.048 | 1.0 | 0.0 | 0.952 |
| 80 | 1 | 1 | 0 | 20 | 31521 | 0.048 | 1.0 | 0.0 | 0.952 |
| 90 | 1 | 1 | 0 | 20 | 31521 | 0.048 | 1.0 | 0.0 | 0.952 |

Table 7.8: Model performance on test dataset 2 for different MAD score ranges. These scores are normalized per connection on a scale of 0 - 100.

Moving a threshold range down from the optimal threshold range, the number of false positives increases from 8 to 87 (for threshold range [40,49]), then further to 148 (for threshold range [30,39]) and finally to 535 (for threshold range [20,29]). While recall and FNR values remain the same, this in turn results in lower precision values of 0.19, 0.12, 0.04 and increased FPR values of 0.003, 0.005 and 0.17 respectively. The model performs worst at threshold ranges above the optimal threshold (threshold ≥ 60) as the number of false negatives drastically increases, leading to very low recall and very high FNR values.

### LOF Score results

Using the LOF as the only scoring metric on this dataset shows a low FPR value of 0.0 and high precision value of 1.0 throughout for all threshold ranges as shown in Table 7.9. For this scoring metric, the model is seen to perform best at a low threshold range of [10,19]. At this threshold, 20 true positive events are detected with 0 false positives and 1 false negative, leading to high recall and precision values of 0.9524 and 1.0 respectively and low FPR and FNR values of 0.0 and 0.05 respectively.

We observe a direct proportionality relationship between the threshold ranges and the number of false negatives, as an increase in the threshold leads to a direct increase in the number of false negatives, resulting in a decrease in the detection performance of our model.

### LoOP Score results only

The results obtained from using the LoOP as the only scoring metric are quite different as shown in Table 7.10. Our model is observed to generally perform badly as it doesn't detect any true positive event for all thresholds ≥ 20. 5 events are detected for all shown threshold ranges, which all turn out to be false positives. This results in recall and precision values of 0.0 throughout, high FNR values of 1.0 and FPR values of 0.0002.

| LOF Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 20 | 0 | 1 | 31521 | 0.9524 | 1.0 | 0.0 | 0.0476 |
| 20 | 12 | 12 | 0 | 9 | 31521 | 0.5714 | 1.0 | 0.0 | 0.428 |
| 30 | 7 | 7 | 0 | 14 | 31521 | 0.33 | 1.0 | 0.0 | 0.667 |
| 40 | 2 | 2 | 0 | 19 | 31521 | 0.0952 | 1.0 | 0.0 | 0.9048 |
| 50 | 2 | 2 | 0 | 19 | 31521 | 0.0952 | 1.0 | 0.0 | 0.9048 |
| 60 | 2 | 2 | 0 | 19 | 31521 | 0.0952 | 1.0 | 0.0 | 0.9048 |
| 70 | 1 | 1 | 0 | 20 | 31521 | 0.0476 | 1.0 | 0.0 | 0.9524 |
| 80 | 1 | 1 | 0 | 20 | 31521 | 0.0476 | 1.0 | 0.0 | 0.9524 |
| 90 | 1 | 1 | 0 | 20 | 31521 | 0.0476 | 1.0 | 0.0 | 0.9524 |

Table 7.9: Model performance on test dataset 2 for different local outlier factor (LOF) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

| LoOP Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 30 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 40 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 50 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 60 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 70 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 80 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |
| 90 | 5 | 0 | 5 | 21 | 31516 | 0.0 | 0.0 | 0.0002 | 1.0 |

Table 7.10: Model performance on test dataset 2 for different local outlier probability (LoOP) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

### Combination of MAD and LoOP Score results

Our model is observed to perform badly when the MAD and LoOP scores are combined together. Table 7.11 shows that while all true positive events are detected at a threshold range of [20, 29], the number of false positives is very high (92). This results in a FPR value of 0.003 and low precision values of 0.18. As the threshold ranges increase, our model performs worst as the number of true positives decreases leading to an increase in the number of false negatives. The number of false positives drops due to the fact that the number of detected events drop.

| Average of MAD and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 113 | 21 | 92 | 0 | 31429 | 1.0 | 0.18 | 0.003 | 0.0 |
| 30 | 10 | 2 | 8 | 19 | 31513 | 0.0952 | 0.2 | 0.0003 | 0.9048 |
| 40 | 6 | 1 | 5 | 20 | 31516 | 0.0476 | 0.17 | 0.0002 | 0.9524 |
| 50 | 5 | 1 | 4 | 20 | 31517 | 0.0476 | 0.2 | 0.0001 | 0.9524 |
| 60 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.11: Model performance on test dataset 2 for different threshold value ranges when the MAD and LoOP scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100.

### Combination of MAD and LOF Score results

Running our model with the MAD and LOF scores as the anomaly scoring metrics provides the results shown in Table 7.12. Our model detects best at thresholds in the range [20, 29] at which all true positive events are detected with a very small number of false positives (9). This leads to a high recall and precision score of 1.0 and 0.7 respectively and low FPR and FNR scores of 0.0003 and 0.0 respectively. Increasing the threshold reduces the number of detected events, reduces the number of true positives, increases the number of false negatives and decreases the detection performance of our model.

| Average of MAD and LOF >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 30 | 21 | 9 | 0 | 31512 | 1.0 | 0.7 | 0.0003 | 0.0 |
| 30 | 24 | 20 | 4 | 1 | 31517 | 0.9524 | 0.833 | 0.0001 | 0.0476 |
| 40 | 9 | 9 | 0 | 12 | 31521 | 0.4 | 1.0 | 0.0 | 0.57 |
| 50 | 2 | 2 | 0 | 19 | 31521 | 0.09 | 1.0 | 0.0 | 0.90 |
| 60 | 2 | 2 | 0 | 19 | 31521 | 0.09 | 1.0 | 0.0 | 0.90 |
| 70 | 1 | 1 | 0 | 20 | 31521 | 0.05 | 1.0 | 0.0 | 0.95 |
| 80 | 1 | 1 | 0 | 20 | 31521 | 0.05 | 1.0 | 0.0 | 0.95 |
| 90 | 1 | 1 | 0 | 20 | 31521 | 0.05 | 1.0 | 0.0 | 0.95 |

Table 7.12: Model performance on test dataset 2 for different threshold value ranges when the MAD and LOF scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100.

In this case, using an optimal threshold range of [20,29] should lead to the best detection performance of our model on this dataset.

### Combination of LOF and LoOP Score results

Using a combined LOF and LoOP scoring metric produces the results shown in 7.13. We notice an overall poor performance of our model as the number of true positives reported remains extremely low, leading to a high number of false negatives, leading to extremely low recall and precision scores and very high FNR values.

| Average of LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 7 | 2 | 5 | 19 | 31516 | 0.09 | 0.28 | 0.0002 | 0.9 |
| 30 | 7 | 2 | 5 | 19 | 31516 | 0.09 | 0.3 | 0.0002 | 0.9 |
| 40 | 6 | 1 | 5 | 20 | 31516 | 0.05 | 0.2 | 0.0002 | 0.9 |
| 50 | 3 | 1 | 2 | 20 | 31519 | 0.05 | 0.3 | 0.0001 | 0.9 |
| 60 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.13: Model performance on test dataset 2 for different threshold value ranges when the LOF and LoOP scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100.

At its best, our model detects 7 events, 5 of which are false positives and 2 true positives, leading to 12 false negative events. This happens at threshold ranges [20,39]. This results in low recall and precision values of 0.09 and 0.3 respectively and very high FNR values of 0.9. This portrays poor data exfiltration detection. Increasing the threshold leads to decreased number of detected events, decreased number of true positive events and an increase in the number of false negative events.

### Combination of MAD, LOF and LoOP Score results

Using the average of all three scores as final anomaly score shows that our model performs best at a low optimal threshold range of [20,29]. At this threshold range, 1 true positive event is missed and 9 false positives are reported. This leads to high recall values of 0.95, high precision scores of 0.7, low FNR of 0.05 and a very low FPR score of 0.0003. At thresholds higher than the optimal threshold ($\geq 30$) we observe a drastic drop in the number of true positive events detected and an increase in the number of false negatives.

This tells us that for this scoring metric, low threshold values should be used for better detection performance as very few to 0 events are detected at high threshold levels.

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 29 | 20 | 9 | 1 | 31512 | 0.95 | 0.7 | 0.0003 | 0.05 |
| 30 | 11 | 6 | 5 | 15 | 31516 | 0.3 | 0.6 | 0.0002 | 0.7 |
| 40 | 2 | 2 | 0 | 19 | 31521 | 0.09 | 1.0 | 0.0 | 0.9 |
| 50 | 1 | 1 | 0 | 20 | 31521 | 0.05 | 1.0 | 0.0 | 0.95 |
| 60 | 1 | 1 | 0 | 20 | 31521 | 0.05 | 1.0 | 0.0 | 0.95 |
| 70 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 21 | 31521 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.14: Model performance on test dataset 2 for different threshold value ranges when the MAD and LOF scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100.

### 7.1.3. Performance on Test dataset 3

#### MAD score results

Running our model with the MAD as only scoring metric on this dataset leads to the results presented in Table 7.15. At low threshold values in the range [20, 29], we observe a high number of true positive events been detected (101) leading to a low number of false negative events (1). This leads to a high recall value of 0.99 and low FNR value of 0.009. Although the FPR seems to be small (0.04), the number of false positives in this case is too high (3579) leading to very low precision values of 0.03. While the number of true positive events detected and number of false negatives remains the same, the number of false positives decreases to 990 for threshold values in the range [30, 39] leading to recall and precision values of 0.99 and 0.09 respectively and FPR and FNR values of 0.01. Our model performs poorly at higher thresholds ≥ 50 as it does no true positive events are detected, leading to low recall and precision values of 0.0 and high FNR values of 1.0.

| MAD Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 3680 | 101 | 3579 | 1 | 80369 | 0.99 | 0.03 | 0.04 | 0.009 |
| 30 | 1091 | 101 | 990 | 1 | 82958 | 0.99 | 0.09 | 0.01 | 0.01 |
| 40 | 575 | 97 | 478 | 5 | 83470 | 0.95 | 0.17 | 0.005 | 0.05 |
| 50 | 345 | 0 | 345 | 102 | 83603 | 0.0 | 0.0 | 0.004 | 1.0 |
| 60 | 139 | 0 | 139 | 102 | 83309 | 0.0 | 0.0 | 0.002 | 1.0 |
| 70 | 64 | 0 | 64 | 102 | 83884 | 0.0 | 0.0 | 0.0008 | 1.0 |
| 80 | 2 | 0 | 2 | 102 | 83946 | 0.0 | 0.0 | 0.0 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |

Table 7.15: Model performance on test dataset 3 for different MAD score ranges only. These scores are normalized per connection on a scale of 0 - 100.

#### LOF Score results

When the LOF is used, we observe that our model performs poorly for all threshold ranges as shown in Table 7.16. No true positive events are detected for any threshold value ≥ 10. This leads to recall and precision values of 0.0 and high FNR values of 1.0. Notice that this scoring metric results to lower number of false positive events when compared to the results from using the MAD.

#### LoOP Score results

Table 7.17 shows the results obtained when the LoOP score is used as scoring metric. Our model performs poorly as it does not detect any of the simulated data exfiltration events at any of the threshold values used. This results in a high number of false negatives, with very low number of false positive events (6 and 5) through out. This also leads to low recall and precision values of 0.0 and high FNR values of 1.0. Notice that the number of false positive events reported are smaller than those reported when the LOF or MAD are used as scoring metric.

| LOF Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 69 | 0 | 69 | 102 | 83879 | 0.0 | 0.0 | 0.0008 | 1.0 |
| 20 | 62 | 0 | 62 | 102 | 83886 | 0.0 | 0.0 | 0.0007 | 1.0 |
| 30 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 40 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 50 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 60 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 70 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 80 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |

Table 7.16: Model performance on test dataset 3 for different local outlier factor (LOF) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

| LoOP Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 6 | 0 | 6 | 102 | 83942 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 30 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 40 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 50 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 60 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 70 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 80 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 90 | 5 | 0 | 5 | 102 | 83943 | 0.0 | 0.0 | 0.0001 | 1.0 |

Table 7.17: Model performance on test dataset 3 for different local outlier probability (LoOP) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

### Combination of MAD and LoOP Score results

When the MAD and LoOP scores are combined together, our model detects a high number of true positive events (97) but also reports a high number of false positives (482) and low number of false negatives for thresholds in the range [20,29]. This leads to a high recall value of 0.95, low precision value of 0.17 and low FNR value of 0.05 as shown in Table 7.18. At any threshold values ≥ 30, the detection performance becomes really poor as no simulated data exfiltration events are detected, with a high number of false positive events, which drops as the detection threshold increases.

| Average of MAD and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 579 | 97 | 482 | 5 | 83466 | 0.95 | 0.17 | 0.006 | 0.05 |
| 30 | 144 | 0 | 144 | 102 | 83804 | 0.0 | 0.0 | 0.002 | 1.0 |
| 40 | 7 | 0 | 7 | 102 | 83941 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 50 | 6 | 0 | 6 | 102 | 83942 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 60 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.18: Model performance on test dataset 3 for different threshold value ranges when the MAD and LoOP scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

### Combination of MAD and LOF Score results

Table 7.19 shows the results obtained when the MAD and LOF scores are combined together by averaging and used as scoring metric. At low thresholds in the range [20,29], 97 true positive events are detected with as high as 481 false positive events. The detection performance of our model becomes poor as the threshold is increased as no true positive events are detected leading to high number of false negatives.

| Average of MAD and LOF >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 578 | 97 | 481 | 5 | 83467 | 0.94 | 0.17 | 0.006 | 0.06 |
| 30 | 140 | 0 | 140 | 102 | 83808 | 0.0 | 0.0 | 0.002 | 1.0 |
| 40 | 64 | 0 | 64 | 102 | 83884 | 0.0 | 0.0 | 0.0008 | 1.0 |
| 50 | 49 | 0 | 49 | 102 | 83899 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 60 | 2 | 0 | 2 | 102 | 83946 | 0.0 | 0.0 | 0.0 | 1.0 |
| 70 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 80 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |

Table 7.19: Model performance on test dataset 3 for different threshold value ranges when the MAD and LOF scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

#### Combination of LOF and LoOP Score results

Combining the LOF and LoOP scores also results in a poor detection performance our model as no simulated data exfiltration events are detected for all threshold ranges as shown in Table 7.20. This leads to very low recall and precision values and very high FNR values. Note that the number of false positive events in this case is quite low, leading to very low FPR values.

| Average of LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 6 | 0 | 6 | 102 | 83942 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 30 | 6 | 0 | 6 | 102 | 83942 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 40 | 6 | 0 | 6 | 102 | 83942 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 50 | 6 | 0 | 6 | 102 | 83942 | 0.0 | 0.0 | 0.0001 | 1.0 |
| 60 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.20: Model performance on test dataset 3 for different threshold value ranges when the LOF and LoOP scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

#### Combination of MAD, LOF and LoOP Score results

When all three scores are combined together, we notice that our model still performs really poorly for all threshold ranges considered as it does not detect any of the simulated data exfiltration events, resulting in a high number of false negatives and also quite a high number of false positive events for low threshold ranges.

## 7.2. Performance on the smallest cluster

In this section, we present the results of our system when run only on the smallest cluster of every test dataset. In this case, we first of all need to find out what the smallest cluster is in terms of number of connections.

The main advantage of this method is that it reduces the number of connections to be considered for analysis, thereby significantly reducing the computational time of our model. In an attacker-defender scenario where the defender has limited computational resources which the attacker doesn't know of, this method will be the best approach to follow. In the case where the attacker knows of the limited computational resources of the defender, then he can circumvent detection by making a small number of benign connections and an extremely large number of malicious connections having very similar feature set values such that the benign connections are placed together in a small cluster while the malicious or exfiltration connections together

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 145 | 0 | 145 | 102 | 83803 | 0.0 | 0.0 | 0.0017 | 1.0 |
| 30 | 67 | 0 | 67 | 102 | 83881 | 0.0 | 0.0 | 0.0008 | 1.0 |
| 40 | 4 | 0 | 4 | 102 | 83944 | 0.0 | 0.0 | 0.0 | 1.0 |
| 50 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 60 | 1 | 0 | 1 | 102 | 83947 | 0.0 | 0.0 | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 102 | 83948 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.21: Model performance on test dataset 3 for different threshold value ranges when the MAD, LOF and LoOP scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

form a thicker, larger cluster, leading to a high number of false negatives and false positives, which is a major disadvantage of this approach. The results of using this approach on the test datasets 1, 2 and 3 are presented in Sections 7.2.1, 7.2.2 and 7.2.3 respectively.

### 7.2.1. Performance on Test dataset 1

The smallest cluster consist of 16 connections as shown in Table 5.4.

#### MAD score results

The results obtained from running our model using the MAD score as the only anomaly scoring metrics is presented and the performance of our model for various MAD score ranges are shown in Table 7.22. We notice that higher thresholds in the range [20,59] can be used to detect both data exfiltrations with no false positives and no false negatives.

| MAD Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 40 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 50 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.22: Model performance on the smallest cluster of test dataset 1 for different MAD score ranges only. These scores are normalized per connection on a scale of 0 - 100.

Table 7.22 depicts no false positives for all thresholds ≥ 20. This results in a low FPR of 0.0 and high precision values of 1.0 at the depicted thresholds. At thresholds ≥ 60, we observe an increase in the number of false negatives and a decrease in the number of true positives. This in turn leads to lower recall values of 0.5 and a higher FNR of 0.5. For this scoring metric alone, thresholds in the range [20, 59] result in the highest recall and precision values of 1.0 and the lowest FPR and FNR values of 0.0. It is, therefore, recommended to use thresholds in the range [20,59] for highest detection of data exfiltration with no false positives and no false negatives.

#### LOF Score results

Table 7.23 shows the performance of our model when it is run with the LOF score as the only anomaly scoring metric for different LOF score ranges. This shows that only low threshold values in the range [20,29] can be used to detect both data exfiltrations in the smallest cluster of this dataset with no false positives and no false negatives. At higher threshold values, the detection performance of the model drops as the number of false negatives increases from 0 to 1.

| LOF Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 40 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.23: Model performance on the smallest cluster of test dataset 1 for different local outlier factor (LOF) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

Table 7.23 shows no false positives for threshold values ≥ 20. This results in a constant false positve rate of 0.0 and precision value of 1.0 for all threshold values ≥ 20. The best detection results with highest number of true positives, no false positives and no false negatives are obtained at a threshold value in the range [20, 29]. At threshold values ≥ 30, we observe no change in the number of false positives which stays at 0, a reduction in the number of true positives and an increase in the number of false negatives. This results in higher false negative rates and lower recall, dropping the detection performance of the model. From these observations, a detection threshold value in the range [20, 29] is recommended for use when the LOF score is used as only anomaly scoring metric.

### LoOP Score results

The performance of our model when run with the LoOP as only scoring framework is shown in Table 7.24. The model its observed to perform at its best for thresholds in the range [20,29], as all data exfiltrations are detected with no false negatives and no false positives, leading to recall and precision values of 1.0 and FNR and FPR values of 0.0. At higher thresholds, the performance of the model is observed to drop.

| LoOP Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 40 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 0 | 0 | 0 | 2 | 14 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 2 | 14 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.24: Model performance on the smallest cluster of test dataset 1 for different local outlier probability (LoOP) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

From Table 7.24, the performance of the model drops for thresholds in the range [30-79], with an increase in the number of false negatives, resulting in lower recall values and higher FNR values. The model is observed to perform worst at thresholds ≥ 80 where it does not detect any of the connections involved in data exfiltration. In the case when the LoOP is used as only scoring framework on the smallest cluster of this test dataset, a detection threshold in the range [20, 29] is recommended.

### Combination of MAD and LoOP Score results

When the MAD and LoOP scores are combined together by taking the average of both, we evaluate the detection performance of our model on the smallest cluster of the first test dataset and present the results in Table 7.25. We observe optimal results in terms of true positives, false positives and false negatives at a threshold value in the range [20,39]. All true positve events are detected, with no false positives and no false negatives resulting in high recall and precision value of 1.0, and low false positive and false negative rates of 0.0. The performance of the model drops at thresholds ≥ 40.

| Average of MAD and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 40 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 0 | 0 | 0 | 2 | 14 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.25: Model performance on the smallest cluster of test dataset 1 for different threshold value ranges when the MAD score and LoOP score are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

Increasing the threshold to a value in the range [40, 89] leads to decrease performance of the model as the number of false negatives increases from 0 to 1, leading to a drop in the recall value from 1.0 to 0.5 and an increase in the FNR from 0.0 to 0.5. The performance takes an even further dive at thresholds ≥ 90 for which no events are detected leading to a higher FNR and an even lower recall value of 0.0. When using the MAD and LoOP scores combined together on the smallest cluster of this test dataset, a threshold value in the range [20-39] is recommended.

## Combination of MAD and LOF Score results

Combining the MAD and LOF scores on the smallest cluster of the first test datasets leads to the results shown in 7.26. The optimal threshold value is found in the range [20-39], where all true positives are detected, with no false positives and no false negatives, leading to a recall and precision of 1.0 and a false negative and false positive rate of 0.0. The performance of the model decreases at threshold values higher than the optimal threshold range (threshold values ≥ 40).

| Average of MAD and LOF >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 40 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.26: Model performance on the smallest cluster of test dataset 1 for different threshold value ranges when the MAD score and LOF score are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

At thresholds ≥ 40, the performance of the model drops as the model detects just 1 of the 2 connections involved in data exfiltration, leading to an increase in the number of false negatives. This reduces the recall from 1.0 to 0.5 and increases the FNR from 0.0 to 0.5. From these results, a threshold in the range [20-39] is recommended.

## Combination of LOF and LoOP Score results

The results obtained whenthe LOF and LoOP scores are combined are presented in Table 7.27. We clearly observe optimal detection results for threshold values in the range [20-29]. In this case, all true positive events are detected with no false negatives and no false positives. Threshold values above the optimal threshold range lead to a decrease in the number of true positives, which in turn leads to an increase in the number of false negatives, decreasing the detection performance of the model.

| Average of LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 40 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 0 | 0 | 0 | 2 | 14 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.27: Model performance on the smallest cluster of test dataset 1 for different threshold value ranges when the LOF score and LoOP score are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

The results presented in Table 7.27 show that threshold values in the range [20, 29] led to the best detection results and should be used if using this scoring metrics. At thresholds in the range [30,89]. the performance of our model drops as the recall drops and the FNR increases. Threshold values ≥ 90 lead to worst performance as the model does not detect any data exfiltration connection, leading to the lowest recall values of 0.0 and the highest FNR of 1.0.

### Combination of MAD, LOF and LoOP Score results

Table 7.28 presents the results obtained when all three scores are combined together. From this table, we see that the best detection results are obtained at threshold values in the range [20-39] with 2 true positives, 0 false negatives and 0 false positives, leading to highest recall and precision values of 1.0 and lowest false negative and false positive rates of 0.0. Threshold values in this range are, therefore, observed to present the best detection results while higher threshold values (≥ 40) increase the number of false negatives with a constant low number of false positives of 0 leading to a decrease in the detection performance of our system.

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 30 | 2 | 2 | 0 | 0 | 14 | 1.0 | 1.0 | 0.0 | 0.0 |
| 40 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 50 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 60 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 70 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 80 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |
| 90 | 1 | 1 | 0 | 1 | 14 | 0.5 | 1.0 | 0.0 | 0.5 |

Table 7.28: Model performance on the smallest cluster of test dataset 1 for different threshold value ranges when the MAD, LOF and LoOP scores are combined together. These scores are combined by averaging them and are normalized per connection on a scale of 0 - 100.

When using these three scores combined on the smallest cluster of this test dataset, the best threshold values are in the range [20-39]. Threshold values ≥ 40 reduce the recall value from 1.0 to 0.5 and increase the FNR from 0.0 to 0.5.

### 7.2.2. Performance on Test dataset 2

The smallest cluster from this dataset contains 667 connections. It is important to note that in this case, all malicious connections were found in the smallest cluster.

### MAD score results

The results obtained when the MAD score is used as the only anomaly scoring metric on the smallest cluster of the second test dataset is shown in Table 7.29. We observe that all true positive events are detected with decreasing number of false positive events as the threshold increases for lower threshold values in the range [20,59]. All 21 true positive events are detected at a threshold in the range [20, 29] with a high number of false positives (245). While the number of true positive detected events remain 21, we notice a drop in the number of false positives from 245 to 103 as the threshold range increases from [20,29] to [30,39], and further to as low as only 2 false positive events for threshold values in the range [50, 59]. At thresholds ≥ 60, the detection performance of the model becomes really poor as the number of detected events drastically drops leading to a drastic increase in the number of false negatives from 0 to 20. This can be seen from the FNR which increases to a value of 0.95 and the recall value which drops from 1.0 to 0.05. The precision is observed to increase to 1.0 and the FPR drops from 0.0 but this is due to the very small number of total detected events.

| MAD Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 245 | 21 | 224 | 0 | 422 | 1.0 | 0.085 | 0.35 | 0.0 |
| 30 | 124 | 21 | 103 | 0 | 543 | 1.0 | 0.17 | 0.16 | 0.0 |
| 40 | 94 | 21 | 73 | 0 | 573 | 1.0 | 0.22 | 0.11 | 0.0 |
| 50 | 23 | 21 | 2 | 0 | 644 | 1.0 | 0.91 | 0.003 | 0.0 |
| 60 | 3 | 2 | 1 | 19 | 645 | 0.095 | 0.67 | 0.002 | 0.245 |
| 70 | 1 | 1 | 0 | 20 | 646 | 0.05 | 1.0 | 0.0 | 0.95 |
| 80 | 1 | 1 | 0 | 20 | 646 | 0.05 | 1.0 | 0.0 | 0.95 |
| 90 | 1 | 1 | 0 | 20 | 646 | 0.05 | 1.0 | 0.0 | 0.95 |

Table 7.29: Model performance on the smallest cluster of test dataset 2 for different MAD score ranges. These scores are normalized per connection on a scale of 0 - 100. smallestC

From Table 7.29, the detection performance of our model is observed to increase as the thresholds increase from threshold values in the range [20,29] to [50,59]. While the recall values and the FNR remain constant at 1.0 and 0.0 respectively , we observe a huge increase in the precision from 0.085 to 0.91 and a drop in the FPR from 0.35 to 0.003 as thresholds increase from threshold scores in the range [20,29] to thresholds in the range [50, 59]. Taking the above into consideration, a threshold value in the range [50, 59] produces the best performance results for this scenario.

## LOF Score results

Table 7.30 shows the results of running our model on the smallest cluster of the second test dataset with the LOF score as the only anomaly scoring metrics. We observe that lower thresholds perform best at detection compared to much higher threshold values. Best detection results are obtained are really low threshold values in the range [10,19], detecting 20 events, all true positive with 1 false negative and 0 false positives, leading to high recall and precision values of 0.95 and 1.0 respectively, and also low FPR and FNR values of 0.0 and 0.047 respectively. As the threshold increases, the performance of the model drops as the number of detected events decreases leading to increased number of false negatives. This in turn results in lower recall values and an increased FPR. It is quite important to note that all detected events for all threshold ranges shown in Table 7.30 are true positives meaning 0 false positives and resulting in a constant precision and FPR value of 1.0 and 0.0 respectively. The table shows us that the best detection threshold range for this scoring metric is [10,19].

## LoOP score results

The results obtained by running our model using just the LoOP score on the smallest cluster of the second test dataset is shown in Table 7.31. We also observe just like in the other scoring metrics that low threshold values in the range [20, 49] lead to the highest number of true positives (21) and therefore, no false negatives, with decreasing number of false positives as the threshold increases. While thresholds in the range [20, 29] detect all true positive events, the number of false positives is quite high (39), which results in high recall values of 1.0, low precision, FPR and FNR values of 0.35, 0.06 and 0.0 respectively. As the thresholds get increased to a value in the range [40,49], we observe that all true positives are still detected, with a smaller number of false positives (9), resulting in high recall and higher precision values of 1.0 and 0.7 respectively, and lower FPR and FNR values of 0.01 and 0.0 respectively.

The performance of the model drops as the threshold is increased above 50 as fewer events are detected, not all of them being true positives. This increases the number of false negatives and reduces the number

| LOF Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 20 | 20 | 0 | 1 | 646 | 0.95 | 1.0 | 0.0 | 0.047 |
| 20 | 12 | 12 | 0 | 9 | 646 | 0.57 | 1.0 | 0.0 | 0.42 |
| 30 | 7 | 7 | 0 | 14 | 646 | 0.33 | 1.0 | 0.0 | 0.667 |
| 40 | 2 | 2 | 0 | 19 | 646 | 0.09 | 1.0 | 0.0 | 0.9 |
| 50 | 2 | 2 | 0 | 19 | 646 | 0.09 | 1.0 | 0.0 | 0.9 |
| 60 | 2 | 2 | 0 | 19 | 646 | 0.09 | 1.0 | 0.0 | 0.9 |
| 70 | 1 | 1 | 0 | 20 | 646 | 0.0476 | 1.0 | 0.0 | 0.95 |
| 80 | 1 | 1 | 0 | 20 | 646 | 0.0476 | 1.0 | 0.0 | 0.95 |
| 90 | 1 | 1 | 0 | 20 | 646 | 0.0476 | 1.0 | 0.0 | 0.95 |

Table 7.30: Model performance on test dataset 2 for different local outlier factor (LOF) score ranges only. These scores are normalized per connection on a scale of 0 - 100. smallestC

| LoOP Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 60 | 21 | 39 | 0 | 607 | 1.0 | 0.35 | 0.06 | 0.0 |
| 30 | 38 | 21 | 17 | 0 | 629 | 1.0 | 0.6 | 0.02 | 0.0 |
| 40 | 30 | 21 | 9 | 0 | 637 | 1.0 | 0.7 | 0.01 | 0.0 |
| 50 | 17 | 13 | 4 | 8 | 642 | 0.61 | 0.76 | 0.006 | 0.38 |
| 60 | 11 | 7 | 4 | 14 | 642 | 0.33 | 0.63 | 0.006 | 0.67 |
| 70 | 5 | 2 | 3 | 19 | 643 | 0.09 | 0.4 | 0.0046 | 0.9 |
| 80 | 5 | 2 | 3 | 19 | 643 | 0.09 | 0.4 | 0.0046 | 0.9 |
| 90 | 3 | 2 | 1 | 19 | 645 | 0.09 | 0.67 | 0.0015 | 0.9 |

Table 7.31: Model performance on test dataset 2 for different local outlier probability (LoOP) score ranges only. These scores are normalized per connection on a scale of 0 - 100. smallestC

of true positives, leading to lower recall values and higher FNR values. While keeping the balance of high true positives, low false positives and low false negatives in mind, the model is observed to perform best at threshold values in the range [30,39].

### Combination of MAD and LoOP Score results

When combining the MAD and LoOp scores, the result obtained are shown in Table 7.35. While the number of true positives decreases as the threshold values increase, we also notice a decrease in the number of false positives and an increase in the number of false negatives. At higher threshold values, the model detects very few events which leads to a higher number of false negatives. At low threshold values, the model detects a higher number of events leading to a higher number of false positives. Based on the numbers shown on the table, the threshold range [40,49] produces the best detection results with all true positive events (21) been detected with very low number of false positives (5) and no false negatives leading to high recall and precision values of 1.0 and 0.8 respectively and low FPR and FNR values of 0.008 and 0.0 respectively. At thresholds ≥ 50, the number of detected events drops to below the number of actual true positives leading to an increase in the number of false negatives and a drop in the number of true positives, leading to a drop in the detection performance of the model.

### Combination of MAD and LOF Score results

Combining the MAD and LOF scores leads to the results presented in Table 7.33. From this table, we notice that at low threshold values in the range [20,29], a high number of events are detected (94), with a high number of false positives (73) and no false negatives. The fact that all attacks are detected leads to a high recall value of 1.0 and a low false negative value of 0.0 The high number of false positives leads to a FPR of 0.11 and low precision values of 0.22. Due to the high number of false positives, this threshold range is not considered to be optimal for data exfiltration detection using this scoring metric. The best or optimal performance is observed at threshold values in the range [30,39]. Although not all malicious events are detected, the model misses just 1 true positive, and has just 1 false positive. This leads to high recall and precision values of 0.95 each and low FPR and FNR values of 0.0015 and 0.05 respectively. At threshold values ≥ 40, the number of

| Average of MAD and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 144 | 21 | 123 | 0 | 523 | 1.0 | 0.15 | 0.19 | 0.0 |
| 30 | 35 | 21 | 14 | 0 | 632 | 1.0 | 0.6 | 0.02 | 0.0 |
| 40 | 26 | 21 | 5 | 0 | 641 | 1.0 | 0.8 | 0.008 | 0.0 |
| 50 | 16 | 13 | 3 | 8 | 643 | 0.62 | 0.81 | 0.005 | 0.38 |
| 60 | 5 | 4 | 1 | 17 | 645 | 0.19 | 0.8 | 0.0015 | 0.81 |
| 70 | 3 | 2 | 1 | 19 | 645 | 0.095 | 0.67 | 0.0015 | 0.9 |
| 80 | 3 | 2 | 1 | 19 | 645 | 0.095 | 0.67 | 0.0015 | 0.9 |
| 90 | 1 | 1 | 0 | 20 | 646 | 0.047 | 1.0 | 0.0 | 0.95 |

Table 7.32: Model performance on test dataset 2 for different threshold value ranges when the MAD and LoOP scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100. smallestC

detected events drastically drops to 7 and then to 2 and further to 1. This leads to a higher number of false negatives and a lower number of true positives, resulting in a bad detection performance of our model.

| Average of MAD and LOF >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 94 | 21 | 73 | 0 | 573 | 1.0 | 0.22 | 0.11 | 0.0 |
| 30 | 21 | 20 | 1 | 1 | 645 | 0.95 | 0.95 | 0.0015 | 0.05 |
| 40 | 7 | 7 | 0 | 14 | 646 | 0.3 | 1.0 | 0.0 | 0.67 |
| 50 | 2 | 2 | 0 | 19 | 646 | 0.095 | 1.0 | 0.0 | 0.9 |
| 60 | 2 | 2 | 0 | 19 | 646 | 0.095 | 1.0 | 0.0 | 0.9 |
| 70 | 1 | 1 | 0 | 20 | 646 | 0.05 | 1.0 | 0.0 | 0.95 |
| 80 | 1 | 1 | 0 | 20 | 646 | 0.05 | 1.0 | 0.0 | 0.95 |
| 90 | 1 | 1 | 0 | 20 | 646 | 0.05 | 1.0 | 0.0 | 0.95 |

Table 7.33: Model performance on test dataset 2 for different threshold value ranges when the MAD and LOF scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100. smallestC

### Combination of LOF and LoOP Score results

Combining the LOF and LoOP scores leads to quite interesting results. At threshold ranges of [20,29], 31 events are detected with 20 true positives, 11 false positives and 1 false negative. This gives a high recall of 0.95, average precision of 0.64, low FPR and FNR values of 0.017 and 0.05 respectively. A threshold range higher ([30,39]) leads to fewer number of detected events with higher number of false negatives (3) and lower number of false positives (4). At threshold ranges ≥ 40, the number of detected events drops drastically, increasing the number of false negatives and leading to a worse performance of the model with FNR values as high as 0.95 and recall values as low as 0.05. These results are shown in Table 7.34.

### Combination of MAD, LOF and LoOP Score results

Table 7.40 shows the results obtained when all 3 scores are combined together. We observe that all malicious events are detected at a threshold range of [20,29] with a high number of false positives (15). Increasing the threshold values to higher threshold ranges decreases the number of detected events, leading to an increase in the number false positives and a decrease in the number of true positives. For thresholds in the range [30,39], a high number of true positives are detected (20), with a very low number of false positives (4) and a low number of false negatives (1). This results in high recall and high precision scores of 0.95 and 0.83 respectively, and low FPR and FNR values of 0.006 and 0.05 respectively. At threshold values ≥ 40, the number of true positives drops drastically, increasing the number of false negatives. Although the number of false positives reduces, this is mainly due to the very small number of detected events. These poor results are shown in the low recall values and the high FNR values proving that the performance of the model decreases as the threshold increases.

| Average of LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 31 | 20 | 11 | 1 | 635 | 0.95 | 0.64 | 0.017 | 0.05 |
| 30 | 22 | 18 | 4 | 3 | 642 | 0.86 | 0.81 | 0.0062 | 0.14 |
| 40 | 10 | 7 | 3 | 14 | 643 | 0.33 | 0.7 | 0.0046 | 0.667 |
| 50 | 4 | 3 | 1 | 18 | 645 | 0.14 | 0.75 | 0.0015 | 0.86 |
| 60 | 2 | 2 | 0 | 19 | 646 | 0.095 | 1.0 | 0.0 | 0.9 |
| 70 | 2 | 2 | 0 | 19 | 646 | 0.095 | 1.0 | 0.0 | 0.9 |
| 80 | 1 | 1 | 0 | 20 | 646 | 0.047 | 1.0 | 0.0 | 0.95 |
| 90 | 1 | 1 | 0 | 20 | 646 | 0.047 | 1.0 | 0.0 | 0.95 |

Table 7.34: Model performance on the smallest cluster of test dataset 2 for different threshold value ranges when the LOF and LoOP scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100.

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 36 | 21 | 15 | 0 | 631 | 1.0 | 0.58 | 0.02 | 0.0 |
| 30 | 24 | 20 | 4 | 1 | 642 | 0.95 | 0.83 | 0.006 | 0.05 |
| 40 | 12 | 11 | 1 | 10 | 645 | 0.52 | 0.91 | 0.0015 | 0.47 |
| 50 | 8 | 7 | 1 | 14 | 645 | 0.33 | 0.87 | 0.0015 | 0.67 |
| 60 | 2 | 2 | 0 | 19 | 646 | 0.095 | 1.0 | 0.0 | 0.9 |
| 70 | 2 | 2 | 0 | 19 | 646 | 0.095 | 1.0 | 0.0 | 0.9 |
| 80 | 1 | 1 | 0 | 20 | 646 | 0.047 | 1.0 | 0.0 | 0.95 |
| 90 | 1 | 1 | 0 | 20 | 646 | 0.047 | 1.0 | 0.0 | 0.95 |

Table 7.35: Model performance on the smallest cluster of test dataset 2 for different threshold value ranges when the MAD, LOF and LoOP scores are combined together. These scores are combined by averaging and are normalized per connection on a scale of 0 - 100. smallestC

If these scores combined are used for anomaly scoring, then low threshold values lead to better detection compared to high detection thresholds. A threshold value in the range [30,39] leads to the best detection if this method is on the smallest cluster of the second test dataset.

### 7.2.3. Performance on test dataset 3

The smallest cluster from this dataset contains 1911 connections. It is also important to note in this case that all malicious connections were located in the smallest.

#### MAD score results

The results obtained when the MAD score is used as anomaly scoring metric on the smallest cluster of this dataset is presented in Table 7.36. All true positive events are detected at low threshold values in the range [20, 39] but with a high number of false positive events (892 for thresholds in the range [20,29] and 534 for thresholds in the range [ 30,39]). This leads to a high recall and FPR values of 1.0 and 0.5 for thresholds in the range [20,29] and high recall and FPR values of 1.0 and 0.3 for thresholds in the range [30,39]. The detection performance of the model is considered poor due to the high number of false positives. As the threshold increases, the number of detected events decreases as well. At thresholds ≥ 50, no true positive events are detected leading to low recall and precision values of 0.0 and high FNR values of 1.0, leading to extremely poor detection abilities of our model.

#### LOF Score results

Our model is also observed to perform poorly on the detection of simulated data exfiltration when the LOF score is used as the only anomaly scoring metric. This can be seen from the results presented in Table 7.37. The maximum number of detected events for all thresholds shown in the table is observed to be 6 while the dataset contains a simulated data exfiltration in 102 connections. From the 6 detected events, it is important to note that no true positive event is detected leading to low recall and precision values of 0.0 and high FNR

| MAD Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 994 | 102 | 892 | 0 | 917 | 1.0 | 0.103 | 0.5 | 0.0 |
| 30 | 636 | 102 | 534 | 0 | 1275 | 1.0 | 0.16 | 0.3 | 0.0 |
| 40 | 439 | 98 | 341 | 4 | 1468 | 0.96 | 0.22 | 0.19 | 0.04 |
| 50 | 253 | 0 | 253 | 102 | 1556 | 0.0 | 0.0 | 0.14 | 1.0 |
| 60 | 23 | 0 | 23 | 102 | 1786 | 0.0 | 0.0 | 0.013 | 1.0 |
| 70 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 80 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |

Table 7.36: Model performance on the smallest cluster of test dataset 3 for different MAD score ranges only. These scores are normalized per connection on a scale of 0 - 100.

values of 1.0. Due to the small number of detected events, the FPR value remains very low. We notice that for this dataset, our model does not detect any of the simulated exfiltrations.

| LOF Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 6 | 0 | 6 | 102 | 1803 | 0.0 | 0.0 | 0.003 | 1.0 |
| 20 | 4 | 0 | 4 | 102 | 1805 | 0.0 | 0.0 | 0.002 | 1.0 |
| 30 | 3 | 0 | 3 | 102 | 1806 | 0.0 | 0.0 | 0.0017 | 1.0 |
| 40 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 50 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 60 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 70 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 80 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |

Table 7.37: Model performance on the smallest cluster of test dataset 3 for different local outlier factor (LOF) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

## LoOP Score results

Our model also performs poorly when the LoOP is used as anomaly scoring metric as can be seen from Table 7.38. We observe that our model detects 2 events having anomaly scores ≥ 79 and 1 event which has an anomaly score ≥ 80. These detected events are all false positives leading to low recall and precision values of 0.0 and high FNR values of 1.0 for all the threshold ranges shown on the table. We, therefore, notice extremely poor detection capabilities of our model when this score is used on the smallest cluster of this dataset.

| LoOP Score >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 30 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 40 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 50 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 60 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 70 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 80 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |

Table 7.38: Model performance on the smallest cluster of test dataset 3 for different local outlier probability (LoOP) score ranges only. These scores are normalized per connection on a scale of 0 - 100.

## Combination of MAD and LoOP Score results

Combining the MAD and LoOP scores, denoted as $S_{MAD\ and\ LoOP}$ leads to 441 detected events with an anomaly scores ≥ 20. From the 441 detected events, 98 are indeed true positives and 345 are false positives, resulting in 4 false negative events. As a result of this, we notice high recall results of 0.96, low precision values of 0.22 and low FNR values of 0.04, indicating poor model performance . At thresholds ≥ 30, no true positive events are detected, resulting in recall values of 0.0 and FNR values of 1.0 as shown in Table 7.39.

| **Average of MAD and LoOP >=** | **Number of detected events** | **TP** | **FP** | **FN** | **TN** | **Recall** | **Precision** | **FPR** | **FNR** |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 441 | 98 | 343 | 4 | 1466 | 0.96 | 0.22 | 0.19 | 0.04 |
| 30 | 25 | 0 | 25 | 102 | 1784 | 0.0 | 0.0 | 0.01 | 1.0 |
| 40 | 3 | 0 | 3 | 102 | 1806 | 0.0 | 0.0 | 0.0017 | 1.0 |
| 50 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 60 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.39: Model performance on the smallest cluster of test dataset 3 for different threshold value ranges when the MAD and LoOP scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

### Combination of MAD and LOF Score results

Combining the MAD and LOF, denoted as $S_{MAD\ and\ LOF}$, leads to almost similar results as those obtained from combining the MAD and LoOP results as 98 true positive events are detected at threshold values in the range [20,29]. At this threshold range, a high number of false positive events are also reported together with 4 false negatives. While the model detects most of the TP events, the high number of false positives reported decreases its performance. This is seen in the low precision values of 0.22 and the high FPR values of 0.19 as shown in Table 7.40.

| Average of MAD and LOF >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 443 | 98 | 345 | 4 | 1464 | 0.96 | 0.22 | 0.19 | 0.04 |
| 30 | 33 | 0 | 33 | 102 | 1776 | 0.0 | 0.0 | 0.02 | 1.0 |
| 40 | 4 | 0 | 4 | 102 | 1805 | 0.0 | 0.0 | 0.002 | 1.0 |
| 50 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 60 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 70 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 80 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 90 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |

Table 7.40: Model performance on the smallest cluster of test dataset 3 for different threshold value ranges when the MAD and LOF scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

### Combination of LOF and LoOP Score results

Combining the LOF and the LoOP scores and using this as anomaly scoring metric, denoted as $S_{LOF\ and\ LoOP}$, leads to the results shown in Table 7.41. 4 events are detected for this scoring metric having anomaly scores ≥ 20, with 3 of these events having an anomaly score between 30 to 39 and 2 events with an anomaly score between 40 to 59. All of these detected events are false positives, leading to a high number of false negatives, which can be seen from the recall values of 0.0 and FNR values of 1.0. Also, no events are detected with an anomaly score ≥ 60. This portrays poor detection performance of our model on this dataset. It is worth mentioning that the number of detected events and the number of false positive events in this case are a lot lower compared to when $S_{MAD\ and\ LoOP}$ and $S_{MAD\ and\ LoOP}$ are used as scoring metric.

| Average of LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 4 | 0 | 4 | 102 | 1805 | 0.0 | 0.0 | 0.002 | 1.0 |
| 30 | 3 | 0 | 3 | 102 | 1806 | 0.0 | 0.0 | 0.0017 | 1.0 |
| 40 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 50 | 2 | 0 | 2 | 102 | 1807 | 0.0 | 0.0 | 0.001 | 1.0 |
| 60 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |
| 70 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |
| 80 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |
| 90 | 0 | 0 | 0 | 102 | 1809 | 0.0 | N/A | 0.0 | 1.0 |

Table 7.41: Model performance on the smallest cluster of test dataset 3 for different threshold value ranges when the LOF and LoOP scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

### Combination of MAD, LOF and LoOP Score results

Combining all three scores also does not lead to an improvement in the detection capability of our model. Results presented in 7.42 show that our model reports on 35 connections with a combined anomaly score ≥ 20 with 4 connections having a combined score ≥ 30 and 1 connection having a high combined score of 64. From all these detected events, no true positive event was reported or detected leading to low recall and precision values of 0.0 and FNR values of 1.0. This portrays poor detection performance of our model when the MAD, LOF and LoOP are combined and used as anomaly scoring metric on this dataset.

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 35 | 0 | 35 | 102 | 1774 | 0.0 | 0.0 | 0.02 | 1.0 |
| 30 | 4 | 0 | 4 | 102 | 1805 | 0.0 | 0.0 | 0.002 | 1.0 |
| 40 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 50 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 60 | 1 | 0 | 1 | 102 | 1808 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 70 | 0 | 0 | 0 | 102 | 1809 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 80 | 0 | 0 | 0 | 102 | 1809 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 90 | 0 | 0 | 0 | 102 | 1809 | 0.0 | 0.0 | 0.0006 | 1.0 |

Table 7.42: Model performance on the smallest cluster of test dataset 3 for different threshold value ranges when the MAD, LOF and LoOP scores are combined together. These scores are combined by using the average and are normalized per connection on a scale of 0 - 100.

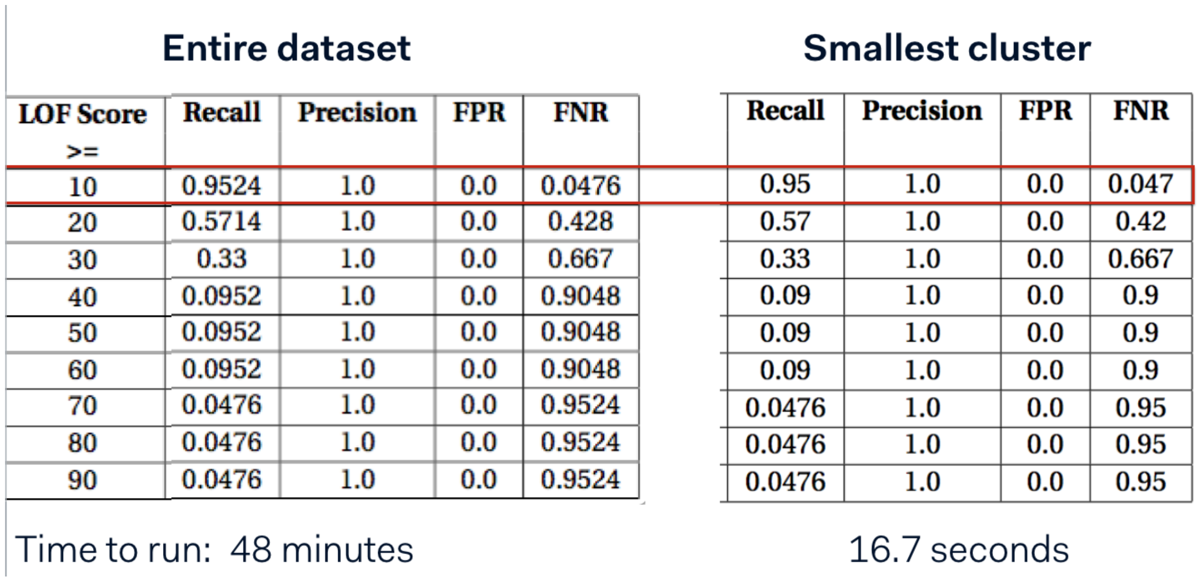
## 7.3. Summary of results from Sections 7.1 and 7.2

Based on the results (recall, precision, fpr, fnr) obtained on running our model on both the entire test datasets and the smallest clusters of the various test datasets, we observe that running our model on the smallest cluster significantly reduces the computation time while on the other hand resulting in very similar detection results when compared to running our model on the entire test datasets as shown in Figures 7.1 and 7.2. The computation times of our model on both the entire test sets and the smallest clusters are shown in Table 7.43. We clearly observe the significant drop in computation time from the entire dataset to the smallest cluster for all datasets.

Analyzing the results obtained when our model is run on the smallest cluster, we observe that the MAD score on all three test datasets always detects **all the true positive events with no false negatives at low thresholds in the range [30, 39]** leading to very high recall values and very low FNR values but with a high number of false positives, resulting in lower precision values and higher FPR values, as can partially be seen in Figure 7.2 for test set 3. The LOF and LoOP on the other hand detects at the same threshold range **only true positive events, with a few false negative events and no false positive events** resulting in high precision and FPR, low recall and FNR values, as can partially be observed in Figure 7.1 for test set 2. A brief summary of the findings is that the MAD performs well at detection but leads to a high number of false positives while the LOF and LoOP also perform well at detection but have a higher number of false negatives.

| Test Set | Computation time on entire dataset | Computation time on smallest cluster |
|---|---|---|
| Test set 1 | 1.45 seconds | 0.93 seconds |
| Test set 2 | 48 minutes | 16.77 seconds |
| Test set 3 | 5.8 hours | 67.2 seconds |

Table 7.43: Computation time on both the entire dataset and the smallest cluster per test set.


Entire dataset

| LOF Score >= | Recall | Precision | FPR | FNR |
|---|---|---|---|---|
| 10 | 0.9524 | 1.0 | 0.0 | 0.0476 |
| 20 | 0.5714 | 1.0 | 0.0 | 0.428 |
| 30 | 0.33 | 1.0 | 0.0 | 0.667 |
| 40 | 0.0952 | 1.0 | 0.0 | 0.9048 |
| 50 | 0.0952 | 1.0 | 0.0 | 0.9048 |
| 60 | 0.0952 | 1.0 | 0.0 | 0.9048 |
| 70 | 0.0476 | 1.0 | 0.0 | 0.9524 |
| 80 | 0.0476 | 1.0 | 0.0 | 0.9524 |
| 90 | 0.0476 | 1.0 | 0.0 | 0.9524 |

Time to run: 48 minutes

Smallest cluster

| Recall | Precision | FPR | FNR |
|---|---|---|---|
| 0.95 | 1.0 | 0.0 | 0.047 |
| 0.57 | 1.0 | 0.0 | 0.42 |
| 0.33 | 1.0 | 0.0 | 0.667 |
| 0.09 | 1.0 | 0.0 | 0.9 |
| 0.09 | 1.0 | 0.0 | 0.9 |
| 0.09 | 1.0 | 0.0 | 0.9 |
| 0.0476 | 1.0 | 0.0 | 0.95 |
| 0.0476 | 1.0 | 0.0 | 0.95 |
| 0.0476 | 1.0 | 0.0 | 0.95 |

16.7 seconds

Figure 7.1: LOF results on both the entire dataset and the smallest cluster of test data 2 and the corresponding running times. Results obtained at threshold range[10,19] show recall of 0.95, precision of 1.0, fpr of 0.0 and fnr of 0.047 for both the entire dataset and the smallest cluster, but the computation time is significantly reduced from 48 minutes on the entire dataset to 16.7 seconds on the smallest cluster. The LOF is chosen at random from the seven scoring metrics to display and the run times shown are the computation times of our model in the different instances on all the seven scoring metrics and not just the LOF.

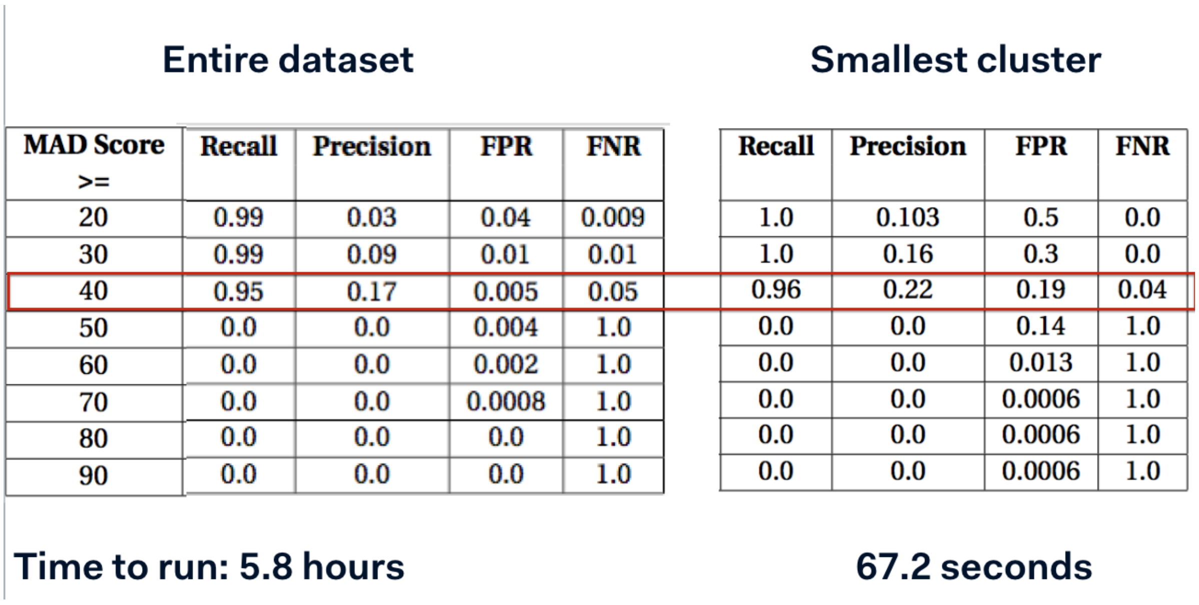
## 7.4. Performance on the validation set

Based on the observations made above, our analysis on the validation set is done only on the smallest cluster. Furthermore, in order to eliminate the biases of using a single individual scoring metric, we combine all three scores and use $S_{MAD,\ LOF\ and\ LoOP}$ (described in Section 6.4) as scoring metric when evaluating the performance on the validation set. The results are presented in Table 7.44. The smallest cluster of this dataset contains 2078 netflows, 152 of which are involved in a simulated data exfiltration. The results obtained from the different test sets shows us that a threshold in the range [40,49] is best to use for optimal results in terms of recall, precision, FPR and FNR scores.

| Average of MAD, LOF and LoOP >= | Number of detected events | TP | FP | FN | TN | Recall | Precision | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 75 | 11 | 64 | 141 | 1862 | 0.1 | 0.15 | 0.03 | 0.9 |
| 30 | 23 | 11 | 12 | 141 | 1914 | 0.1 | 0.5 | 0.006 | 0.9 |
| 40 | 14 | 11 | 3 | 141 | 1923 | 0.1 | 0.8 | 0.002 | 0.9 |
| 50 | 11 | 8 | 3 | 144 | 1923 | 0.1 | 0.7 | 0.002 | 0.9 |
| 60 | 3 | 1 | 2 | 151 | 1924 | 0.01 | 0.3 | 0.001 | 0.99 |
| 70 | 1 | 0 | 1 | 152 | 1925 | 0.0 | 0.0 | 1.0 | 1.0 |
| 80 | 1 | 0 | 1 | 152 | 1925 | 0.0 | 0.0 | 1.0 | 1.0 |
| 90 | 1 | 0 | 1 | 152 | 1925 | 0.0 | 0.0 | 1.0 | 1.0 |

Table 7.44: Model performance on the smallest cluster of the validation for different threshold ranges when the MAD, LOF and LoOP scores are combined by averaging. These scores are normalized per connection on a scale of 0 - 100.

Table 7.44 presents the results obtained when the MAD, LOF and LoOP are combined and used as scoring metric on the validation set. This dataset contains a total of 5 simulated data exfiltration scenarios (scenarios 4 - 8) carried out in a total of 152 malicious netflows, 131 of which are from scenario 8. Scenarios 4, 5, 6 and 7 contain 4, 6, 7 and 4 malicious netflows respectively. From the results shown, we notice that a maximum of 11 out of the 152 malicious netflows get detected at threshold ranges [20,49]. While these results may seem poor, it stems mostly from the fact that the netflows from scenario 8 (132 out of 152) all go undetected. This is mainly because their feature values are very similar to those of benign connections. At threshold ranges [40,49], which we determined as optimal from the results from the three test sets, 14 events are detected, 3



| | Entire dataset | | | | Smallest cluster | | | |
|---|---|---|---|---|---|---|---|---|
| MAD Score >= | Recall | Precision | FPR | FNR | Recall | Precision | FPR | FNR |
| 20 | 0.99 | 0.03 | 0.04 | 0.009 | 1.0 | 0.103 | 0.5 | 0.0 |
| 30 | 0.99 | 0.09 | 0.01 | 0.01 | 1.0 | 0.16 | 0.3 | 0.0 |
| 40 | 0.95 | 0.17 | 0.005 | 0.05 | 0.96 | 0.22 | 0.19 | 0.04 |
| 50 | 0.0 | 0.0 | 0.004 | 1.0 | 0.0 | 0.0 | 0.14 | 1.0 |
| 60 | 0.0 | 0.0 | 0.002 | 1.0 | 0.0 | 0.0 | 0.013 | 1.0 |
| 70 | 0.0 | 0.0 | 0.0008 | 1.0 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 80 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0006 | 1.0 |
| 90 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0006 | 1.0 |

Time to run: 5.8 hours

67.2 seconds

Figure 7.2: MAD results on both the entire dataset and the smallest cluster of test data 3 and the corresponding running times. Results obtained at threshold range[40,49] show similar recall, precision, fpr and fnr for both the entire dataset and the smallest cluster, but the computation time is significantly reduced from 5.8 hours on the entire dataset to 67.2 seconds on the smallest cluster. The MAD is chosen at random from the seven scoring metrics to display and the run times shown are the computation times of our model in the different instances on all the seven scoring metrics and not just the MAD.

of which turn out to be false positives. This results in low recall and fpr values and high precision and fnr values. A more detailed explanation of the detected data exfiltration scenarios in this dataset are presented in Section 7.5. The undetected data exfiltration scenarios are presented in Section 7.6 and a brief analysis of the false positives obtained at a threshold range of [40,49] is presented in Section 7.7.

## 7.5. Detected attack capabilities

### 7.5.1. Covert channel exfiltration

A huge advantage of our system is that it can also detect data exfiltrations that have been concealed through steganography or some other covert channels. Steganography is a storage covert channel in which sensitive data is hidden or encoded in non-sensitive data so that they seem non-sensitive [38]. This is predominantly because we analyze NetFlows in this research. Irrespective of what covert channel is used to conceal the data exfiltration been carried out, the basic high-level information about that connection is not altered, giving us the chance to still detect an attack without doing any form of deep packet inspection.

### 7.5.2. Exfiltrations via normal ports

As presented in Section 3.3, ports are not one of the features used in this research to detect data exfiltration. This sounds quite contradictory as attackers tend to use some specific ports to exfiltrate data [25]. Since attackers expect defenders to do checks on these specific ports, they mostly tend to exfiltrate data via normal ports in order to go undetected [25, 26]. We, therefore, discard this feature so as not to limit the data exfiltration scope of our model to only specific ports. Data exfiltration scenarios 1 and 2 in the second test set shown in Section 4.1.2 all went through normal ports and were still detected by our model. Scenarios 5 and scenarios 7 present in the validation dataset also use normal ports of the tcp protocol and are also detected by our model. In the case where our model considered only specific ports used for attacks over networks, then our model would not detect these simulated data exfiltration scenarios.

The netflows from scenario 7 and their feature-set values are shown in Table 7.45. The table shows us that the data was exfiltration in big individual uploads of sizes between 17MB and 70MB in a duration of about 3 minutes each. Also, we notice that they all have a high PCR value of 0.96 which indicates a pure push (as explained in Table 3.1). Note that these all occurred through normal ports.

| Netflow | Duration | SrcBytes | SrcBytes per packet | SrcBytes per second | PCR |
|---|---|---|---|---|---|
| 1 | 199 | 63359453 | 1379 | 318389.21 | 0.96 |
| 2 | 199 | 70320360 | 1379 | 353368.64 | 0.96 |
| 3 | 201 | 58692274 | 1379 | 292001.36 | 0.96 |
| 4 | 211 | 17514257 | 1378 | 83005.96 | 0.96 |

Table 7.45: Feature values of netflows involved in simulated data exfiltration scenario 7

### 7.5.3. Exfiltration via HTTP

HTTP which is the Hypertext Transfer Protocol is used to relay information between client and servers as it is used by web browsers to reach websites and communicate with the web servers. The fact that HTTP is very often used in every organization makes it suitable for use by attackers. Attackers blend in the data they are trying to exfiltrate with the high volumes of HTTP traffic going through the network in order to go undetected. Data exfiltration scenario 1 in the second test dataset presented in Section 4.1.2 which occurred through two websites didn't go undetected by our model. Also scenario 5 which occurred by wetransfer, which is also an HTTP service was also detected. The individual netflows from scenario 5 are presented in Table 7.46.

| Netflow | Duration | Src_bytes | Src_bytes per packet | Src_bytes per second | PCR |
|---|---|---|---|---|---|
| 1 | 148 | 27603842 | 1377 | 186512.45 | 0.97 |
| 2 | 148 | 43489080 | 1377 | 293845.14 | 0.97 |
| 3 | 148 | 32552527 | 1377 | 219949.51 | 0.97 |
| 4 | 149 | 27202136 | 1376 | 182564.67 | 0.97 |
| 5 | 149 | 8718674 | 1370 | 58514.59 | 0.97 |
| 6 | 150 | 5427687 | 1369 | 36184.58 | 0.97 |

Table 7.46: Feature values of netflows involved in simulated data exfiltration scenario 5

## 7.6. Undetected attacks scenarios

Running our model on the smallest cluster of the validation and test sets resulted in some simulated data exfiltrations going through undetected, mainly because they were not located in the smallest cluster of these datasets. While all the netflows in scenarios 3 and 4 go through completed undetected, a minority of the netflows in scenarios 6 and 8 get detected, leaving a majority undetected. Note that the exfiltrations in scenarios 3 and 8 are carried out using a very similar technique. The netflows in the various scenarios that went undetected by our model are then presented and explained in this section.

### 7.6.1. Scenario 3 and 8

Looking at the simulated data exfiltration scenarios present in our test and validation sets, we observe that our model does not detect data exfiltration scenario 3 in the third test dataset presented in Section 4.1.2. In this scenario, the attacker succeeds in circumventing detection because he has full knowledge of how our model operates. He then splits a huge file of 1MB into smaller 10KB chunks and exfiltrates these small files at different times. At connection-level, a connection uploading 10KB of data within a few minutes will seem benign when compared to the other benign connections in the dataset. In order to detect this attack scenario, a host-level detection should be implemented as explained in Section 8.2.2. Simulated data exfiltration scenario 8 has similar features and also goes undetected.

### 7.6.2. Scenario 4

Data exfiltration scenario 4 in Section 4.1.3 also goes undetected by our model. Looking at the netflows of this connection shown in Table 7.47, we see that they all have a pcr value $\leq 0$. This is because the uploaded source bytes is a lot smaller compared to the received destination bytes. Also, we observe that the netflows make small uploads of 3KB in 3 minutes, 12KB in 6 minutes , 4KB in 5 minutes and 135KB in 6 minutes causing them to seem normal as this characteristics are very similar to those of benign connections in this dataset.

| Netflow | Duration | Src_bytes | Src_bytes per packet | Src_bytes per second | PCR |
|---|---|---|---|---|---|
| 1 | 181 | 3030 | 94 | 16.74 | -0.63 |
| 2 | 381 | 12791 | 83 | 33.57 | -0.86 |
| 3 | 338 | 4405 | 97 | 13.03 | -0.23 |
| 4 | 383 | 135684 | 63 | 354.27 | -0.97 |

Table 7.47: Feature values of netflows involved in simulated data exfiltration scenario 4

### 7.6.3. Scenario 6

Most of the individual netflow connections of simulated data exfiltration scenario 6 also go undetected by our model. To gain an understanding of why this occurs, let us look at the individual netflows of this scenario presented in Table 7.48. While netflow 1 gets detected by our model, netflows 2 - 7 go undetected by our model. Notice that the feature values of netflow 1 are very similar to those of the individual netflows in scenarios 4 and 7 shown in tables 7.47 and 7.45 which are detected by our model as the upload is a big one of size 57MB sent within 2 minutes and having a PCR value of 0.98. Netflows 2-7 on the other hand are of a smaller size (2-7KB) and all have a negative PCR value. These are not detected by our model because their feature values are very similar to that of benign connections in the dataset. The fact that atleast one of the netflows is detected should lead to further investigation and possible detection of the other netflows.

This shows us that our model is biased towards big uploaders and also towards connections sending out a lot more information than they receive (pure pushers as explained in Table 3.1). This is mainly because the big uploader is considered malicious while the small uploaders with small or negative PCR values are considered more normal and as such go undetected.

| Netflow | Duration | Src_bytes | Src_bytes per packet | Src_bytes per second | PCR |
|---|---|---|---|---|---|
| 1 | 70 | 57250676 | 1378 | 817866.80 | 0.98 |
| 2 | 15 | 1040 | 130 | 69.33 | 0.27 |
| 3 | 186 | 4197 | 87 | 22.56 | -0.92 |
| 4 | 187 | 7858 | 70 | 42.02 | -0.91 |
| 5 | 186 | 7750 | 67 | 41.67 | -0.91 |
| 6 | 186 | 3350 | 95 | 18.01 | -0.63 |
| 7 | 186 | 2657 | 91 | 14.28 | -0.10 |

Table 7.48: Feature values of netflows involved in simulated data exfiltration scenario 6

## 7.7. False positives

Table 7.49 shows the 3 netflows from the validation set which were falsely detected as been involved in a data exfiltration for thresholds in the range [40,59] and their feature-set values. Looking at these netflows, we observe that their feature values are very similar to those of the netflows from the detected attack scenarios 5 and 7 presented in tables 7.46 and 7.45. These netflows contain huge uploads with source bytes in the magnitudes of MB with netflows 2 and 3 taking about 2 minutes while netflow 3 takes about 5 minutes. It is also worth noticing that the PCR values of these netflows are also very high, all having a value of 0.97, which is very similar to the PCR value of the netflows from the detected data exfiltration scenarios. With these feature values, it makes complete sense that our model detects them as been malicious.

| Netflow | Duration | Src_bytes | Src_bytes per packet | Src_bytes per second | PCR |
|---|---|---|---|---|---|
| 1 | 520 | 398590512 | 1377.0 | 766520.22 | 0.97 |
| 2 | 78 | 61182784 | 1349.0 | 784394.67 | 0.97 |
| 3 | 73 | 33936422 | 1250.0 | 464882.49 | 0.97 |

Table 7.49: Feature values of netflows wrongly considered to be involved in data exfiltration

# 8 Limitations and Future Work

In this section, we present the limitations of our system and the recommendations for future work which could either improve its performance or extend its functionality.

Our research contains limitations related to both our dataset and techniques used. While the former are limitations that occur from our dataset and can, therefore, be fixed by using a different dataset, the latter refers to those limitations that arose due to the methods used in our approach and can only be fixed by using a different methodology or approach.

## 8.1. Limitations

The biggest limitation of our system in its current state is the **performance**. The step that takes the most amount of time is the computation of the **local outlier factor** and **local outlier probability** anomaly scores. For both of these algorithms, more than 99% of their computation time is spent on finding the nearest neighbors of every connection. This step has a quadratic time complexity which makes the algorithm slower in the case of larger datasets. On average, executing our anomaly scoring step on test dataset 2 presented in Section 4.1.2, containing 63,220 flows takes on average about **40 minutes**. Since this dataset contains the netFlows of just four hosts for about a week, analyzing the weekly flows of all hosts in an enviroment with about 1000 hosts (larger industrial companies have more hosts) will take approximately 166.6 hours. Analyzing the netFlows of all active hosts in a day will take about 23.8 hours, assuming the computational duration increases linearly over time, meaning an entire day is required to analyze the flows of 1000 hosts for a day. In this research, we found another solution to this problem which is performing the analysis on the smallest cluster of every dataset. This drastically reduces the number of datapoints to be considered for the analysis, significantly reducing the computation time from 40 minutes to 16.7 seconds.

However, running our model on the smallest cluster of every dataset could be a potential limitation. An attacker who produces a substantial proportion (more than 30%) of the network traffic can go undetected by ensuring that all his netflows have similar feature values. In this case, all his netflows will be placed in the same cluster due to the similarity of their feature values. Since the attacker accounts for more than 30% of the total network traffic, the attacker's flows will never be in the smallest cluster considering the fact that in this research we use a k value of 4 for the total number of clusters. A major disadvantage with this method of evading detection is that a single internal host who accounts for more than 30% of the network traffic in a large-scale environment of about 1000 hosts also raises an alarm and will call for further investigation.

One of the limitations of our system in its current state is that it cannot handle streaming data. While this will be a good to have feature in the long run, our system currently handles only static data. Most clustering algorithms are not designed to take data streaming into consideration. This is a problem that has been extensively researched in the past years and has been found as a limitation [2, 12, 31, 37, 75]. In order to accommodate streaming data, the building of clusters would need to be made more dynamic, which is out of the scope of this research.

Detecting data exfiltrations at connection-level granularity leads to a bias on large uploaders. Small uploaders are considered to be normal and an attacker who schedules small uploads at specific time intervals could go undetected as his traffic could be considered normal. Analysis on a host-level granularity could lead to interesting results as the aggregated information of a host could disclose a potential data exfiltration, which can not be detected by analyzing its individual connections. Although this solution is not implemented, the choice of feature-set which can be used on the host-level is selected, explained and motivated in Section 3.2.

While using the producer-consumer ratio explained in Section 3.1.5 has an added advantage in the easy detection of connections involved in a potential data exfiltration, it leads to a bias towards connections doing a pure push (explained in Table 3.1). Connections doing a pure push have a high PCR value $\geq 0.8$ and are considered by our model to be potential data exfiltrations, mostly because the send out way more information compared to what they receive. Connections with a lower PCR value are considered normal although data could also be exfiltrated in an email in which case, the PCR value is 0.4 and also with HTTP browsing where the PCR value is -0.5.

## 8.2. Future Work

The work carried out in this research has shown promising results in the detection of data exfiltration scenarios over the network in a large-scale industrialized context. This proof of concept opens doors to possible future work which can be done not only on the industrialized context but also on the research aspects stemming from this work. These are presented in the sections below.

### 8.2.1. Other clustering algorithm

An interesting component of our system is the clustering engine. In this research, K-means is used as the main clustering algorithm based on the criteria presented in Section 5. As a result of this, we might have missed a clustering algorithm that does not fulfill the aforementioned criteria but that could be better in grouping netflows for anomaly detection. For future work, other clustering algorithms which are suitable for network traffic anomaly detection, such as DBSCAN and HDBSCAN can be used.

### 8.2.2. Host-level classification

This research focused on detecting data exfiltration on a connection-level granularity. Performing the same analyses on a host-level granularity and adding this as an extra layer to our proposed solution could lead to a more robust model as it could reduce the number of false positives and false negatives. In this case, hosts features to be used have been selected and presented in Section 3.2. The same methodology used in this research to detect data exfiltration on a connection-level could also be applied on a host-level. This could be interesting as hosts who act normal on a connection-level could be detected by looking at their aggregated host-level features. Looking at data exfiltration scenario 3 in Section 4.1.2, we observe that this scenario is not detected by our model at connection-level as an upload of 10KB seem reasonably normal when compared to the other benign connections in the dataset. If the analysis was done at host-level, this host might stick out uploading an aggregate of 1MB of data in 3.5 hours.

### 8.2.3. Cluster change over time

Analyzing the cluster change per host over time could lead to quite interesting results (in the detection of data exfiltration). This can be done only when detection is done on host-level granularity. Hosts transitioning from a larger, more normal cluster to a smaller anomalous cluster could be a potential indicator of a host exfiltrating data.

### 8.2.4. Alternative distance metric

Finally, the euclidean distance is used to measure the distance between every connection and its cluster centroid. Considering the fact that connections are in a multivariate feature space, some other distance measure can be used, such as the mahalanobis distance which works well in the computation of the distance between points in a multivariate space. We made use of the euclidean distance because k-means which is the clustering algorithm used in this research is implemented with the euclidean distance as distance metric.

# 9 Conclusion

This research presents an exploratory study towards, **finding an automated and non-privacy invasive data exfiltration detection solution**.

We propose a technique which utilizes lightweight non-privacy invasive netflow features to detect data exfiltrations over networks. The key intuition behind our proposed solution is that connections involved in data exfiltrations differentiate themselves from normal network connections based on their extracted features. We then cluster connections based on their feature values using k-means. We further use a combination of statistical (robust zscore calculation), density (LOF and LoOP) and distance (distance from cluster centroid) based anomaly detection techniques, to score connections. These scores are allocated based on both the robust Z-score of the distance from the respective cluster centroid and the density of the region in which connections find themselves in the feature space, compared to that of their k-nearest neighbor. Combining these approaches eliminates the bias of using just a single approach and leads to better performance, with a significant drop in the number of false positives and false negatives.

Answering the research question posed in this dissertation led to some major contributions. These major contributions are:

- Identifying what features from netflow metadata can be used to detect data exfiltration over networks at different granularities. At connection-level granularity, the duration, source bytes, source bytes per second, source bytes per packet and the producer-consumer ratio are used as feature-set. The ports and protocols are discarded.
- Identifying an appropriate clustering algorithm for grouping network connections for data exfiltration detection, producing self explanatory clusters, having minimal parameter space, being scalable with larger datasets.
- Identifying a technique to detect anomalous connections involved in data exfiltration based on their respective clusters and cluster centroids. This technique detects a high number of true positives, but with a significant number of false positives.
- Identifying a technique independent of the formed clusters and cluster centroids to detect anomalous connections involved in data exfiltration. This technique performs well in the detection of true positives, with a few false negatives.
- Combining both a distance, statistical and density based anomaly scoring approach for better data exfiltration detection results, leading to low false positives and low false negatives.
- Analyzing the smallest clusters instead of the entire datasets, which, leads to similar detection results but significantly reduces the computation time.

We present the answers to the research questions posed in Section 1.5 below:

**RQ: Is the use of NetFlow metadata effective in detecting data exfiltration over networks in a large-scale industrial context?**

Our approach uses non-privacy invasive netflows to detect data exfiltrations over networks. The results obtained from our solution shows that connections can indeed be scored using a combination of density, distance and statistical anomaly scoring approaches which can then be used to detect connections involved in a data exfiltration. Based on the allocated scores and an optimal threshold, we observe that most but not all simulated data exfiltration scenarios in the various test datasets are indeed detected. The undetected attack scenarios have similar feature values to that of benign connections, increasing the difficulty of their detection. Also, some benign connections with feature values similar to those of the detected attack connections are falsely classified as being an attack.

**Sub-RQ1: What features extracted from the available NetFlow metadata are effective in detecting data exfiltration?**

The choice of features used for the detection of data exfiltrations over networks from netflows depends on the granularity. Features can be chosen for detection at either the connection-level or aggregated host-level. In this research, although we focus on detection of data exfiltrations at a connection level, we also provide a set of features which can be used for detection of data exfiltration at the host-level. These host-level features are presented in section 3.2 and are proposed by [30]. On the connection-level granularity, the duration of a connection, the source bytes, the source bytes per packet, the source bytes per second and the producer-consumer ratio are used as feature-set as presented in section 3.1. These features are observed to be effective in the detection of data exfiltration as motivated in Section 3.1.

**Sub-RQ2: What clustering technique is appropriate to use for the detection of data exfiltration using NetFlow metadat?**

In order to select one clustering algorithm out of the several algorithms out there, we narrowed down the clustering algorithms to be considered based on a few criteria. The clustering algorithm should first of all be suitable for the detection of network traffic anomaly detection in an unsupervised setting. Also, due to the large number of netflows logged and collected at Adyen, the clustering algorithm should be scalable with large datasets. Furthermore, in order to exhaustively explore the parameter space for optimal results, the algorithm should have a minimal number of parameters. Lastly, the clusters produced should be self-explanatory to domain experts. With these criteria, we end up choosing K-means presented in Section 5.1.

**Sub-RQ3: What techniques can we use to detect NetFlow traffic connections involved in data exfiltration based on the results from the clustering algorithm used in sub-question 2?**

After exploring the clusters produced by our clustering algorithm on the different test datasets (containing some simulated data exfiltration scenarios), we notice that our clusters contained both normal and anomalous connections. As proposed in [4, 6], when clusters are formed with both benign and anomalous datapoints, the normal datapoints tend to be closer to the cluster centroid while the anomalous datapoints tend to be further away from the cluster centroid. With this in mind, we calculate per connection, the distance from their cluster centroid and use that as a distance-based technique to detect connections involved in data exfiltration. The distances from the cluster centroids are then normalized and scaled in the range [0, 100] such that 0 represents benign connections and 100 represents anomalous connections. This score is used as the **first anomaly score** allocated to all connections in the test and validation datasets. This technique is explained and presented in Section 6.1. This technique was effective in detecting some data exfiltration scenarios but resulted in a significant number of false positives.

**Sub-RQ4: What other techniques independent from the technique in sub-question 3 can we use to detect connections involved in data exfiltration and how do we combine them?**

In order to eliminate the bias of using only a distance-based approach and reduce the number of false positives, we adopt a density-based approach. Based on several reasons presented in Section 6.2, we use the local outlier factor and the local outlier probability algorithms, which both use the idea of local densities to detect anomalies. The outputs of these algorithms are then normalized and scaled in the range of [0, 100], with a score of 0 meaning that connection is benign and a score of 100 meaning that connection is most likely involved in data exfiltration. These scores are used to allocate the **second and third anomaly scores** to every connection, as presented in Sections 6.2 and 6.3. Although this technique was effective in detecting some data exfiltrations, it resulted in a small number of false negatives. These scores are then linearly combined with the scores from the distance-based technique by averaging as explained in Section 6.4. The result of this is a significant drop in the number of false positives and false negatives.

**Sub-Question 5: What anomaly scoring metric performs best in the detection of simulated data exfiltration scenarios and at what threshold does this perform best?**

Observing the detection capabilities of our model on the different test sets shows that the combination of all three anomaly scoring metrics works best in the detection of the simulated data exfiltration scenarios. Although not all data exfiltration scenarios are detected, threshold values in the range [40,49] work best for this scoring metric, as they result in high precision values and low fpr. Due to the undetected scenarios, the recall value is low and the fnr high.

To conclude, the results obtained from this research show that netflow metadata (lightweight, high level and non privacy-invasive) can indeed be used to detect some data exfiltration scenarios. Although these results are promising, our solution has some limitations such as, a bias towards large uploaders, a bias towards pure pushers (explained in Section 3.1.5), inability to handle streaming data. Some optimizations are, therefore, needed to make our developed proof of concept ready for deployment in a real world setting.

# Bibliography

[1] Aymen Abid, Abdennaceur Kachouri, and Adel Mahfoudhi. Anomaly detection through outlier and neighborhood data in wireless sensor networks. In *2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, pages 26–30. IEEE, 2016.

[2] Charu C Aggarwal. A framework for diagnosing changes in evolving data streams. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 575–586. ACM, 2003.

[3] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, pages 420–434. Springer, 2001.

[4] Mohiuddin Ahmed and Abdun Naser Mahmood. A novel approach for outlier detection and clustering improvement. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*, pages 577–582. IEEE, 2013.

[5] Mohiuddin Ahmed and Abdun Naser Mahmood. Clustering based semantic data summarization technique: a new approach. In *2014 9th IEEE Conference on Industrial Electronics and Applications*, pages 1780–1785. IEEE, 2014.

[6] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[7] Muhammad Qasim Ali, Ehab Al-Shaer, Hassan Khan, and Syed Ali Khayam. Automated anomaly detector adaptation using adaptive threshold tuning. *ACM Transactions on Information and System Security (TISSEC)*, 15(4):17, 2013.

[8] Muhammad H Arshad and Philip K Chan. Identifying outliers via clustering for anomaly detection. Technical report, 2003.

[9] Julija Asmuss and Gunars Lauks. Network traffic classification for anomaly detection fuzzy clustering based approach. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 313–318. IEEE, 2015.

[10] Juozas Auskalnis, Nerijus Paulauskas, and Algirdas Baskys. Application of local outlier factor algorithm to detect anomalies in computer network. *Elektronika ir Elektrotechnika*, 24(3):96–99, 2018.

[11] Stefan Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security (TISSEC)*, 3(3):186–205, 2000.

[12] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM, 2002.

[13] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[14] Robin Berthier, Michel Cukier, Matti Hiltunen, Dave Kormann, Gregg Vesonder, and Dan Sheleheda. Nfsight: netflow-based network awareness tool. In *Proceedings of LISA'10: 24th Large Installation System Administration Conference*, page 119, 2010.

[15] Elisa Bertino and Gabriel Ghinita. Towards mechanisms for detection and prevention of data exfiltration by insiders: keynote talk paper. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 10–19. ACM, 2011.

[16] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336, 2013.

[17] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 159–164. ACM, 2006.

[18] Daniela Brauckhoff, Xenofontas Dimitropoulos, Arno Wagner, and Kavè Salamatian. Anomaly extraction in backbone networks using association rules. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 28–34. ACM, 2009.

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

[20] Ross Brewer. Advanced persistent threats: minimising the damage. *Network security*, 2014(4):5–9, 2014.

[21] John Carter. *PCR - A new flow metric*, 2014 (accessed May 1, 2019). URL https://qosient.com/argus/presentations/Argus.FloCon.2014.PCR.Presentation.pdf.

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

[23] David D Clark and Susan Landau. The problem isn't attribution: it's multi-stage attacks. In *Proceedings of the Re-architecting the Internet Workshop*, page 11. ACM, 2010.

[24] Robert Todd Graham Collins. The privacy implications of deep packet inspection technology: Why the next wave in online advertising shouldn't rock the self-regulatory boat. *Ga. L. Rev.*, 44:545, 2009.

[25] The Mitre corporation. *Commonly used ports*, 2018 (accessed August 13, 2019). URL https://attack.mitre.org/techniques/T1043/.

[26] The Mitre corporation. *Exfiltration over alternative protocols*, 2018 (accessed August 13, 2019). URL https://attack.mitre.org/techniques/T1048/.

[27] The Mitre corporation. *Data exfiltration transfer file limits*, 2018 (accessed September 6, 2019). URL https://attack.mitre.org/techniques/T1030/.

[28] The Mitre corporation. *Scheduled data exfiltration file transfer*, 2018 (accessed September 6, 2019). URL https://attack.mitre.org/techniques/T1029/.

[29] Michal Daszykowski, Krzysztof Kaczmarek, Yvan Vander Heyden, and Beata Walczak. Robust statistics in data analysis—a review: Basic concepts. *Chemometrics and intelligent laboratory systems*, 85(2): 203–219, 2007.

[30] Guillaume Dewaele, Yosuke Himura, Pierre Borgnat, Kensuke Fukuda, Patrice Abry, Olivier Michel, Romain Fontugne, Kenjiro Cho, and Hiroshi Esaki. Unsupervised host behavior classification from connection patterns. *International Journal of Network Management*, 20(5):317–337, 2010.

[31] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Kdd*, volume 2, page 4, 2000.

[32] Julie S Downs, Mandy B Holbrook, and Lorrie Faith Cranor. Decision strategies and susceptibility to phishing. In *Proceedings of the second symposium on Usable privacy and security*, pages 79–90. ACM, 2006.

[33] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM workshop on Mining network data*, pages 281–286. ACM, 2006.

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

[35] Kieran Flanagan, Enda Fallon, Abir Awad, and Paul Connolly. Self-configuring netflow anomaly detection using cluster density analysis. In *2017 19th International Conference on Advanced Communication Technology (ICACT)*, pages 421–427. IEEE, 2017.

[36] Chris Fraley and Adrian E Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

[37] Olga Georgieva and Frank Klawonn. Dynamic data assigning assessment clustering of streaming data. *Applied Soft Computing*, 8(4):1305–1313, 2008.

[38] Annarita Giani, Vincent H Berk, and George V Cybenko. Data exfiltration and covert channels. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense V*, volume 6201, page 620103. International Society for Optics and Photonics, 2006.

[39] Prasanta Gogoi, DK Bhattacharyya, Bhogeswar Borah, and Jugal K Kalita. A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, 54(4):570–588, 2011.

[40] John Clifford Gower. Euclidean distance geometry. *Math. Sci*, 7(1):1–14, 1982.

[41] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998.

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

[43] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

[44] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of intelligent information systems*, 17(2-3):107–145, 2001.

[45] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[46] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[47] Gaofeng He, Tao Zhang, Yuanyuan Ma, and Bingfeng Xu. A novel method to detect encrypted data exfiltration. In *2014 Second International Conference on Advanced Cloud and Big Data*, pages 240–246. IEEE, 2014.

[48] Khadija Ramah Houerbi, Kavé Salamatian, and Farouk Kamoun. Scan surveillance in internet networks. In *International Conference on Research in Networking*, pages 614–625. Springer, 2009.

[49] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[50] Ahmad Jakalan, Jian Gong, Zhang Weiwei, and Qi Su. Clustering and profiling ip hosts based on traffic behavior. *Journal of Networks*, 10(2):99, 2015.

[51] Vijay Karamcheti, Davi Geiger, Zvi Kedem, and Shanmugavelayutham Muthukrishnan. Detecting malicious network traffic using inverse distributions of packet contents. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 165–170. ACM, 2005.

[52] Darren R Kerr and Barry L Bruins. Network flow switching and flow data export, June 5 2001. US Patent 6,243,667.

[53] Andreas Kind, Marc Ph Stoecklin, and Xenofontas Dimitropoulos. Histogram-based traffic anomaly detection. *IEEE Transactions on Network and Service Management*, 6(2):110–121, 2009.

[54] Benjamin King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62 (317):86–101, 1967.

[55] Edwin M Knox and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the international conference on very large data bases*, pages 392–403. Citeseer, 1998.

[56] Hans-Peter Kriegel, Peer Kröger, Erich Schubert, and Arthur Zimek. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1649–1652. ACM, 2009.

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

[58] R Kumari, MK Singh, R Jha, NK Singh, et al. Anomaly detection in network traffic using k-mean clustering. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 387–393. IEEE, 2016.

[59] Anukool Lakhina, Mark Crovella, and Christophe Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM computer communication review*, volume 35, pages 217–228. ACM, 2005.

[60] Steve Astels Leland McInnes, John Healy. *Benchmarking performance and scaling of python clustering algorithms*, 2016 (accessed July 5, 2019). URL https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html.

[61] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

[62] Bingdong Li, Mehmet Hadi Gunes, George Bebis, and Jeff Springer. A supervised machine learning approach to classify host roles on line using sflow. In *Proceedings of the first edition workshop on High performance and programmable networking*, pages 53–60. ACM, 2013.

[63] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. Detection and identification of network anomalies using sketch subspaces. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 147–152. ACM, 2006.

[64] Wei-Chao Lin, Shih-Wen Ke, and Chih-Fong Tsai. Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78:13–21, 2015.

[65] Yali Liu, Cherita Corbett, Ken Chiang, Rennie Archibald, Biswanath Mukherjee, and Dipak Ghosal. Detecting sensitive data exfiltration by an insider attack. In *Proceedings of the 4th annual workshop on Cyber security and information intelligence research: developing strategies to meet the cyber security and information intelligence challenges ahead*, page 16. ACM, 2008.

[66] Yali Liu, Cherita Corbett, Ken Chiang, Rennie Archibald, Biswanath Mukherjee, and Dipak Ghosal. Sidd: A framework for detecting sensitive data exfiltration by an insider attack. In *2009 42nd Hawaii International Conference on System Sciences*, pages 1–10. IEEE, 2009.

[67] Yingqiu Liu, Wei Li, and Yun-Chun Li. Network traffic classification using k-means clustering. In *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, pages 360–365. IEEE, 2007.

[68] Wei Lu and Hengjian Tong. Detecting network anomalies using cusum and em clustering. In *International Symposium on Intelligence Computation and Applications*, pages 297–308. Springer, 2009.

[69] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[70] Matthew V Mahoney. Network traffic anomaly detection based on packet bytes. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 346–350. ACM, 2003.

[71] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. Analysis of high volumes of network traffic for advanced persistent threat detection. *Computer Networks*, 109:127–141, 2016.

[72] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14, 2011.

[73] Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, pages 13–14, 2007.

[74] George Nychis, Vyas Sekar, David G Andersen, Hyong Kim, and Hui Zhang. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 151–156. ACM, 2008.

[75] Liadan O'callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha, and Rajeev Motwani. Streaming-data algorithms for high-quality clustering. In *Proceedings 18th International Conference on Data Engineering*, pages 685–694. IEEE, 2002.

[76] PandaLabs. *PandaLabs Annual Report 2018*, December 2018 (accessed May 5, 2019). URL https://partnernews.pandasecurity.com/uk/src/uploads/2018/12/PandaLabs-2018_Annual_Report-uk.pdf.

[77] Jamie Parfet. *Conducting and Detecting Data Exfiltration*, May 2018 (accessed September 5, 2019). URL https://www.mindpointgroup.com/blog/operations/conducting-and-detecting-data-exfiltration/.

[78] José Ramón Pasillas-Díaz and Sylvie Ratté. An unsupervised approach for combining scores of outlier detection techniques, based on similarity measures. *Electronic Notes in Theoretical Computer Science*, 329:61–77, 2016.

[79] Nerijus Paulauskas and Ąžuolas Faustas Bagdonas. Local outlier factor use for the network flow anomaly detection. *Security and Communication Networks*, 8(18):4203–4212, 2015.

[80] Slobodan Petrovic, Gonzalo Alvarez, Agustin Orfila, and Javier Carbo. Labelling clusters in an intrusion detection system using a combination of clustering evaluation techniques. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, volume 6, pages 129b–129b. IEEE, 2006.

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

[82] Pradeep Rai and Shubha Singh. A survey of clustering techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010.

[83] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[84] JPJ Rousseeuw. A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational Application Math*, 1989.

[85] Adriana C Ferrari Santos, José Demisio Simoes da Silva, Lılia de Sá Silva, and Milena Prado da Costa Sene. Network traffic characterization based on time series analysis and computational intelligence. *Journal of Computational Interdisciplinary Sciences*, 2(3):197–205, 2011.

[86] IBM security and ponemon institute. *Cost of a data breach study*, July 2019 (accessed August 28, 2019). URL https://newsroom.ibm.com/2019-07-23-IBM-Study-Shows-Data-Breach-Costs-on-the-Rise-Financial-Impact-Felt-for-Years#assets_all.

[87] Shokri Z Selim and Mohamed A Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on pattern analysis and machine intelligence*, (1):81–87, 1984.

[88] Archana Singh, Avantika Yadav, and Ajay Rana. K-means with three different distance metrics. *International Journal of Computer Applications*, 67(10), 2013.

[89] Mayank Pal Singh, N Subramanian, et al. Visualization of flow data based on clustering technique for identifying network anomalies. In *2009 IEEE Symposium on Industrial Electronics & Applications*, volume 2, pages 973–978. IEEE, 2009.

[90] Peter HA Sneath, Robert R Sokal, et al. *Numerical taxonomy. The principles and practice of numerical classification.* 1973.

[91] Marc Ph Stoecklin, Jean-Yves Le Boudec, and Andreas Kind. A two-layered anomaly detection technique based on multi-modal flow behavior models. In *International Conference on Passive and Active Network Measurement*, pages 212–221. Springer, 2008.

[92] Duygu Sinanc Terzi, Ramazan Terzi, and Seref Sagiroglu. Big data analytics for network anomaly detection from netflow data. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 592–597. IEEE, 2017.

[93] Faheem Ullah, Matthew Edwards, Rajiv Ramdhany, Ruzanna Chitchyan, M Ali Babar, and Awais Rashid. Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, 101:18–54, 2018.

[94] Risto Vaarandi. Detecting anomalous network traffic in organizational private networks. In *2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, pages 285–292. IEEE, 2013.

[95] Kurt Varmuza and Peter Filzmoser. *Introduction to multivariate statistical analysis in chemometrics*. CRC press, 2016.

[96] Alampallam Ramaswamy Vasudevan and Subramanian Selvakumar. Local outlier factor and stronger one class classifier based hierarchical model for detection of attacks in network intrusion detection dataset. *Frontiers of Computer Science*, 10(4):755–766, 2016.

[97] Verizon. *2018 Data Breach Investigations Report*, 2018 (accessed April 5, 2019). URL https://enterprise.verizon.com/resources/reports/DBIR_2018_Report_execsummary.pdf.

[98] Nikos Virvilis, Oscar Serrano, and Luc Dandurand. Big data analytics for sophisticated attack detection. *Isaca Journal*, 3:22–25, 2014.

[99] Arno Wagner and Bernhard Plattner. Entropy based worm and anomaly detection in fast ip networks. In *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*, pages 172–177. IEEE, 2005.

[100] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[101] Stefan Weigert, Matti A Hiltunen, and Christof Fetzer. Community-based analysis of netflow for early detection of security incidents. In *LISA*, 2011.

[102] Charles T Zahn. Graph theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(SLAC-PUB-0672-REV):68, 1970.