



UNIVERSIDADE  
FEDERAL DE  
ALAGOAS



UNIVERSIDADE FEDERAL DE ALAGOAS  
INSTITUTO DE COMPUTAÇÃO  
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Dissertação de Mestrado

**A Framework of Unsupervised Techniques for  
Anomaly-Based Intrusion Detection**

Anderson Santos da Silva  
ass2@ic.ufal.br

**Orientador:**

Baldoino Fonseca dos Santos Neto

MACEIÓ

2019

Anderson Santos da Silva

# **A Framework of Unsupervised Techniques for Anomaly-Based Intrusion Detection**

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Informática do Instituto de Computação da Universidade Federal de Alagoas.

**Orientador:**

Baldoino Fonseca dos Santos Neto

Maceió  
2019

**Catálogo na fonte**  
**Universidade Federal de Alagoas**  
**Biblioteca Central**

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

S586f Silva, Anderson Santos da.

A framework of unsupervised techniques for anomaly-based intrusion detection /Anderson Santos da Silva. – 2019.  
50 f. : il.

Orientador: Balduino Fonseca dos Santos Neto.

Dissertação (mestrado em Informática) - Universidade Federal de Alagoas.  
Instituto de Computação. Maceió, 2019.

Bibliografia: f. 47-50.

1. Algoritmos - anomalias. 2. Detecção de intrusão. 3. Algoritmos não supervisionados. 4. Ciberterrorismo. I. Título.

CDU: 004.49



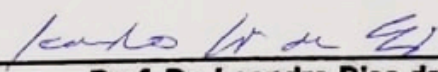
UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL  
Programa de Pós-Graduação em Informática – Ppgi  
Instituto de Computação  
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins  
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

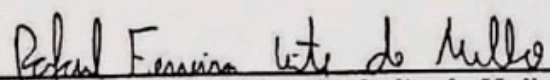


Membros da Comissão Julgadora da Dissertação de Anderson Santos da Silva intitulada: *"A Framework of Unsupervised Techniques for Anomaly-Based Intrusion Detection"*, apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas em 24 de janeiro de 2019, às 15h00, na Sala Alan Turin, do Instituto de Computação da UFAL.

### COMISSÃO JULGADORA

  
\_\_\_\_\_  
**Prof. Dr. Balduino Fonseca dos Santos Neto**  
UFAL – Instituto de Computação  
Orientador

  
\_\_\_\_\_  
**Prof. Dr. Leandro Dias da Silva**  
UFAL – Instituto de Computação  
Examinador Interno

  
\_\_\_\_\_  
**Prof. Dr. Rafael Ferreira Leite de Mello**  
Universidade Federal Rural de Pernambuco  
Examinador Externo

# Acknowledgments

In memory of my mother, Iracilda Santos da Silva, who passed away during this journey, you will be forever in my heart, thanks for all the affection, attention, and love, always educating and guiding me to follow the best steps. I would like to thank the almighty God for giving me strength and perseverance to move on even in the hardest times. My Father João Francisco da Silva; to my sister Aparecida Santos da Silva, my brother Alecson Santos da Silva and my sister-in-law Munick França Pinheiro; My aunt, Ana Falconeri. They were always present, giving all kinds of support. Without my family, I would have not come this far.

To my advisor Balduino Santos Neto, for encouraging me through hard times and being harsh when it was needed. Thanks for the trust, support, and patience during this time.

To all the friends who followed this trajectory very closely: Francisco Dalton, Anderson Oliveira, Jairo Raphael, and Estevam. A special thanks to Caio Barbosa and Filipe Falcão whom with I had the great opportunity to work and live abroad.

Finally, I gratefully acknowledge the Higher Education Personnel Improvement Coordination (CAPES), and the European FP7-IRSES DEVASSES program for financial support.

# Abstract

Dozens of algorithms have been proposed for anomaly detection, and, when applied to intrusion detection, they can detect suspect attacks whenever relevant deviations from the expected behavior are observed. The research community still lacks a universal comparative evaluation as well as standard publicly available datasets. It is in general challenging to provide a description that suffices in details, and that is easy to understand and compare. It may often appear that valuable solutions are presented and specially tested in such a way that re-implementation by a third party or comparison with others solutions is difficult, time-consuming and the result might not even be the same. For example, a step in the algorithm might say: *"We pick an element from the frontier set"* but which element do you pick? Will the first one do? Why will any element suffice? As another example, the author may probably want to give more implementation details but is constrained by the paper page limit. Additionally, sometimes the author's description in-lines other algorithms or data structures that perhaps only that author is familiar. In general, it is a common struggle to research and show a quantitative comparison that gives evidence of the quality of a solution. While this is undoubtedly essential for further researches and improvements in the topic, it is challenging to create a quantitative comparison which allows a fair comparison of different anomaly detection techniques. Thus, a public quantitative analysis for anomaly detection algorithms, which can be used by anyone and eventually allow anyone to contribute to, implying that the tests are in a standard format, is much needed.

**Keywords:** Anomaly Detection, Intrusion Detection, Unsupervised Algorithms, Attacks Datasets, Attack Model

# Resumo

Dezenas de algoritmos foram propostos para detecção de anomalias e, quando aplicados à detecção de intrusão, eles podem detectar ataques suspeitos sempre que forem observados desvios relevantes do comportamento esperado. A comunidade de pesquisa ainda carece de uma avaliação comparativa universal, bem como de conjuntos padrão de dados disponíveis ao público. É geralmente um desafio fornecer uma descrição que seja suficiente em detalhes e que seja fácil de entender e comparar. Pode parecer que soluções valiosas são apresentadas e especialmente testadas de forma que a reimplementação por terceiros ou a comparação com outras soluções é difícil, demorada e o resultado pode nem ser o mesmo. Por exemplo, uma etapa do algoritmo pode dizer: "Escolhemos um elemento do conjunto de fronteiras", mas qual elemento você escolhe? O primeiro fará? Por que algum elemento é suficiente? Como outro exemplo, o autor provavelmente pode querer fornecer mais detalhes de implementação, mas está restrito pelo limite de páginas de um artigo. Além disso, às vezes a descrição do autor alinha outros algoritmos ou estruturas de dados que talvez apenas esse autor esteja familiarizado. Em geral, é um problema comum pesquisar e mostrar uma comparação quantitativa que evidencie a qualidade de uma solução. Embora isso seja, sem dúvida, essencial para novas pesquisas e melhorias no tema, é um desafio criar uma comparação quantitativa que permita uma comparação justa de diferentes técnicas de detecção de anomalias. Portanto, é necessária uma análise comparativa para algoritmos de detecção de anomalias, que possa ser usado por qualquer pessoa e, eventualmente, permitir que alguém contribua com esses testes estando em um formato padrão.

**Palavras-chave:** Detecção de Anomalias, Detecção de Intrusão, Algoritmos Não Supervisionados, Conjuntos de Dados de Ataques, Modelo de Ataque

# List of Figures

2.1	Advantages of density-based techniques. . . . .	19
2.2	Difference between the neighborhoods computed by LOF and COF. . . . .	20
3.1	Methodology to select, transform and aggregate surveyed data. We show the three dimensions of analysis selection of the algorithms, selection of the datasets, unified attack model, and selection of the metrics. . . . .	26
3.2	A taxonomy of unsupervised anomaly detection techniques. . . . .	31
4.1	The architecture with its three layers. . . . .	33
4.2	Sequences of time series inputs for the fuzzers type of attack. . . . .	34
4.3	Creation of the sub-datasets, isolating single types of attacks for KDD-CUP-99 and NSL-KDD datasets. . . . .	35
4.4	Creation of the sub-datasets, isolating single types of attacks for UNSW-NB15 dataset. . . . .	35
4.5	Class Diagram of Algorithm Detectors the Detector Module. . . . .	36
4.6	Class Diagram of the Algorithm Detectors Execution at Detector Layer. . . . .	37
4.7	First step: <i>data layer</i> . . . . .	38
4.8	Second step: data layer and detector layer . . . . .	38
4.9	Final step. data layer, detector layer and evaluator layer . . . . .	39
5.1	Results on all the datasets and all the attacks, grouped by algorithms families. Columns report median scores, while error bars depict the standard deviation . . . . .	42
5.2	Median metric scores for all the datasets and algorithms, grouped by attacks. . . . .	43



# Table of contents

<b>1</b>	<b>Introduction</b>	<b>12</b>
1.1	Proposal . . . . .	13
1.2	Relevance . . . . .	14
1.3	Structure . . . . .	14
<b>2</b>	<b>Background and Related Work</b>	<b>15</b>
2.1	The Anomaly Detection Problem . . . . .	15
2.2	Definitions . . . . .	16
2.3	Anomaly Detection Techniques . . . . .	17
2.3.1	Distance-based . . . . .	17
2.3.2	Statistical . . . . .	21
2.3.3	Classification . . . . .	22
2.3.4	Angle-based . . . . .	22
2.4	Comparing Anomaly Detectors . . . . .	23
<b>3</b>	<b>Study Design</b>	<b>25</b>
3.1	Datasets . . . . .	26
3.2	Unified Attack Model . . . . .	27
3.3	Metrics for Evaluation . . . . .	28
3.4	Anomaly Detection Techniques . . . . .	30
3.4.1	Selection of the Algorithms . . . . .	30
<b>4</b>	<b>The Framework for Quantitative Comparison of Anomaly Detection Techniques</b>	<b>32</b>
4.0.1	General Concepts About Frameworks . . . . .	32
4.1	Designing a Framework for Anomaly Detection Techniques . . . . .	33
4.1.1	The Data Layer . . . . .	34
4.1.2	The Detector Layer . . . . .	35
4.1.3	The Evaluator Layer . . . . .	37
4.2	Execution Sequence . . . . .	37
<b>5</b>	<b>Results and Discussion</b>	<b>40</b>
5.1	Experiment Comparison and Previous Results . . . . .	40
<b>6</b>	<b>Conclusion and Future Work</b>	<b>45</b>

# 1

## Introduction

The size and complexity of computer-intensive systems have grown dramatically during the past decade, and the trend will undoubtedly continue in the future. The demand for complex hardware/software systems has increased more rapidly than the ability to design, implement, test, and maintain them. After all, when the requirements and dependencies on computers increase, the possibility of crises due to computer failures also increases. Examples of consequences are an inconvenience (e.g., malfunctions of home appliances), economic damage (e.g., interruptions of banking systems) and loss of life (e.g., failures of flight systems or medical software).

The reliability defined as the probability that a system (component) will function over a while [1] has become a significant concern for our society. Although in dependable and secure systems often is mandatory to deploy monitoring solutions of the system and it is services to timely detect an unexpected behavior that may differ from the norm and can lead to failures, such process is known as *anomaly detection*, and these patterns are called *anomalies*. Even though anomalous events infrequently occur, when they do occur, such consequences can be quite dramatic, often in a negative way.

In order to avoid cyber-attacks in systems and networks, amongst protection countermeasures, *Intrusion Detection Systems* (IDSs, [2]) were proposed to enhance network and system security. IDSs collect and analyze data from networks and systems indicators to detect malicious or unauthorized activities, based on the hypothesis that an ongoing attack has distinguishable effects on such indicators. Most of enterprise IDSs adopt *signature-based* detection algorithms [2], which consist of looking for predefined patterns (or "signatures") in the monitored data in order to detect an ongoing attack. Data is usually seen as a flow of data points, which represent observations of the values of the indicators at a given time. Signature-based approaches usually score high detection capabilities and low false positive rates when experimenting known attacks [?], but they cannot effectively adapt their behavior when systems evolve or when their configuration is modified. As an additional consequence, they are not meant to detect *zero day attacks*, which are novel attacks that cannot be matched to any known signature [3]. Moreover, when a zero-day attack that exploits newly added or undiscovered system vulnerabilities are identified, its signature needs to be derived and added as a new rule to the IDS [4].

To mitigate the problem above, *Anomaly-based* IDSs rely on anomaly de-

tection algorithms. Anomaly detection has received much attention in the last decade, and numerous detectors have been proposed. Despite that, operators often discredit the alarms reported by anomaly detectors due to several drawbacks discrediting them [5, 6]. Anomaly detection techniques are often applied on unlabeled data. In contrast to standard classification tasks, taking only the structure of the data set into account and the reason why is that given the ever-increasing scale coupled with the high complexity of software, applications and workloads patterns, anomaly detection methods must operate automatically at runtime without the need for prior knowledge about normal or anomalous behavior [7]. Without prior knowledge of the system behavior, unsupervised anomaly detection is addressed in much practical application, for example, in fraud detection, network intrusion detection as well as in the life science and medical domain. It assumes that anomalies are much less common than the normal data in the dataset.

Different anomaly detection algorithms usually exhibit different rates of missed (*False Negatives*) and wrong detection (*False Positives*) and, consequently, have different detection capabilities. Although most of such algorithms have a generic, context-independent formulation, they are often more effective to detect specific attacks on specific systems or applications. Moreover, the manifestation of the anomaly is usually different from attack to attack and from system to system. Consequently, selecting the correct detection algorithm represents a crucial decision when defining an anomaly-based IDS. A wrong choice of the algorithm will decrease the attack detection capabilities of the IDS, consequently reducing the ability to secure the target system and network. With that in mind, we aim to present a framework for a methodology for quantitative comparison of anomaly detection algorithms applied to multiple attack datasets.

## 1.1 Proposal

In general, it is a common struggle to research and show a quantitative comparison that gives evidence of the quality of a solution. While this is undoubtedly essential for further researches and improvements in the topic, it is challenging to create a robust comparison. Thus, a public quantitative comparison for anomaly detection algorithms, which can be used by anyone and eventually allow anyone to contribute to, implying that the tests are in a standard format, is much needed.

Therefore, this work aims to provide the following contributions:

- A comparative experimental evaluation between the execution of unsupervised anomaly detection algorithms targeting the different data sets. This will offer a first identification of how the different algorithms behave under the different workload and faultload offered by the different data sets. A categorical classification of different anomaly detection algorithms will allow organizing different algorithms in classes so that comparison between algorithms (of the same, similar or even different classes) can be evaluated appropriately accordingly to their;

- A methodology and a related environment, more precisely a framework that allows a quantitative comparison of different anomaly detection algorithms. The methodology and environment are devised, starting from the experience acquired in the previous steps. The objective is to create a way to compare the different algorithms in a fair and reproducible way. This will require the identification of the target workloads and faultloads, under which each of the different algorithms can be classified. A set of different datasets will be defined, each one representing the data collected of a different kind of software. A selection of the metrics used for the comparison between the algorithms is needed and shall be applied consistently.

## 1.2 Relevance

Summarizing, there are no extensive comparisons of unsupervised anomaly detection algorithms for intrusion detection systems. The works [8] and [9] considered multiple aspects, but they do not mainly focus on aspects of security and intrusion detection systems; in fact, they used datasets from multiple domains. Moreover, in [10] the authors considered a single proprietary dataset, while the work in [11] uses two datasets and four algorithms, without taking into account all the main families of algorithms defined in [12] and refined in [8]. Similarly, in [13] the authors used 3 algorithms on 2 datasets, while, [14] uses 3 unsupervised algorithms on 2 datasets. As a final remark, none of the reviewed papers organizes the results according to a unified attack model, that categorizes attacks from the different datasets.

To fill this gap, in this work it will be i) defined a pool of algorithms, selected from the families of unsupervised algorithms identified in [12], [8], ii) selection of publicly available attacks datasets, iii) define an unified attack model which categorizes attacks from the datasets above, iv) adopt the most used scoring metrics [15].

## 1.3 Structure

The structure of the remainder of this thesis is organized as follows. In chapter 2 presents the problem, some definition about the theme of anomaly detection with background anomaly detection techniques and the limitations of the related work. Chapter 3 presents the study design and the methodology used to compare different anomaly detection algorithms. Chapter 4 presents the structure and execution of the framework for anomaly detection. Chapter 5 presents some preliminary results obtained by experiments and shown as research questions. Chapter 6 concludes the thesis with some discussions about the results, methodology, and future works.

## 2

# Background and Related Work

## 2.1 The Anomaly Detection Problem

There always been a great interest in detecting "not-normal" instances within the dataset. In machine learning, this process is commonly known as anomaly detection or outlier detection. Probably, the first definition was given by Grubbs in 1969 [16]: An outlying observation, or *outlier*, is one that appears to deviate considerably of other instances of the sample in which it occurs. At the time, the main reason for the detection was to remove the outliers ultimately from the training data once pattern recognition algorithms were quite sensitive to outliers in the data; a procedure called *data cleansing*. The development of more robust classifiers decreased interest in anomaly detection. Nevertheless, the interest of researchers in anomalies itself started to grow around the year 2000, since they are often associated with particular events that may lead to suspicious data records. Since then, many new algorithms were developed. The definition of anomalies was extended such that today, they are known to have two essential characteristics:

1. Anomalies are different from the norm with respect to their features and
2. They are rare in a dataset compared to normal instances.

Other several topics are related to anomaly detection. Noise detection which focuses on processing the data for the removal of noises which would turn difficult the study patterns in the data, this deviates from anomaly detection in the sense that in anomaly detection, it considers more critical finding those records rather than just filtering them. Moreover, Novelty Detection refers to the detection of new patterns that were previously absent or overlooked. The normal model is usually modified by those patterns, which is the significant disparity between it and anomaly detection. The approaches used to solve those topics are often similar to outlier detection.

The traditional rule-based systems are often enhanced in many application domains by the use of anomaly detection algorithms. Several challenges make the anomaly detection problem increasingly obscure. To begin with, the borderline between normal and anomalous behavior is often imprecise. **Intrusion detection** is probably the most well-known application of anomaly detection [17, 18]. In this scenario, occurs the monitoring of network traffic and server

application. Potential intrusion attempts and exploits should then be identified using anomaly detection techniques. In this domain, the normal behavior is constantly evolving, such that those changes are mistakenly identified as outliers. Moreover, the anomaly detection techniques need to be adapted to different application domains. Also, the scarcity of labeled data for training and validation imposes limitations on results and conclusions reached.

## 2.2 Definitions

According to [19] authors classify anomalies into either:

- *Point* anomalies are when single data records deviate from the remainder of the dataset. This is the simplest kind and most addressed by existing algorithms.
- *Contextual* anomalies are when the record has behavioral and contextual attributes. The same behavioral attributes could be considered normal in a giving context and anomalous in another.
- *Collective* anomalies are when a group of similar data are deviating from the remainder of the data set. This can only occur in data sets where records are related to each other. They can be converted into point anomalies by aggregating over the context.

At this research all implemented algorithms explicitly handle point anomalies.

Depending on the system, data can be stored in different ways. The way the data is stored strongly influences the data analysis strategy that turns out to be suitable for such a case study. More in details, some datasets provide one or more labels associated with a data instance, denoting if that instance of data corresponds to an unstable state of the system, e.g., an attack was performed at the time instant in which the data instance was collected. It should be noted that to obtain labeled data which is accurate and representative for all types of behaviors is often prohibitively expensive. The activity of labeling the data is often done manually by a human expert and hence requires a substantial effort to obtain the labeled training dataset. Basically, the anomaly detections setup depends on the labels available in the dataset, and we can distinguish between the following three modes [20].

1. **Supervised Anomaly Detection** describes the setup where the data comprises of fully labeled training and test data sets. An ordinary classifier can be trained first and applied afterwards. This scenario is very similar to traditional pattern recognition with the exception that classes are typically strongly unbalanced. Not all classification algorithms suit therefore perfectly for this task. There are two major issues that arise in supervised anomaly detection. First, the anomalous instances are far fewer compared to the normal instances in the training data. Issues that arise due to imbalanced class distributions have been addressed in the data mining and



machine learning literature [21]. Second, obtaining accurate and representative labels, especially for the anomaly class is usually challenging. A number of techniques that inject artificial anomalies in a normal data set to obtain a labeled training data set have been proposed [22, 23].

2. **Semi-supervised Anomaly Detection** also uses training and test datasets, whereas training data only consists of normal data without any anomalies. The basic idea is, that a model of the normal class is learned and anomalies can be detected afterwards by deviating from that model. Since they require less labels, their range of applicability is wider than supervised techniques.
3. **Unsupervised Anomaly Detection** is the most flexible setup which does not require any labels. Furthermore, there is also no distinction between a training and a test dataset. The idea is that an unsupervised anomaly detection algorithm scores the data solely based on intrinsic properties of the dataset. Typically, distances or densities are used to give an estimation what is normal and what is an outlier. This article only focuses on this unsupervised anomaly detection setup.

## 2.3 Anomaly Detection Techniques

### 2.3.1 Distance-based

The notion of distance-based outliers was introduced in the study of databases [24]. According to this notions, a point  $P$ , in a multidimensional data set is anomalous if there are less than  $p$  points from the data in the  $\varepsilon$ -neighborhood of  $P$ , where  $p$  is a user-specified constant. Ramaswamy et al. [24] described an approach that is based on computing the *Euclidian distance* of the  $k^{th}$  nearest neighbor from a point  $O$ .

There are three major approaches:

- Neighbor-based;
- Density-based;
- Clustering-based.

#### Neighbor-based

The neighbor-based algorithms consider anomalies as point most distant from other points, it classifies points by assigning them to the class that appears most frequently among the  $k$  nearest neighbors. Therefore, for a given point  $O$ ,  $d_k(O)$  denotes the *Euclidian distance* from the point  $O$  to its  $K^{th}$  nearest neighbor and can be considered as the "*degree of outlierness*" of  $O$ . If one is interested in the top  $n$  outlier, if the distance to its  $K^{th}$  nearest neighbor is smaller than the corresponding valor for no more than  $(n - 1)$  other points. In other words, the top  $n$  outliers with the maximum  $d_k(O)$  values are considered

as outliers. While the advantage of the  $K^{th}$  nearest neighbors approach is that it is robust to noisy data, the approach suffers from the drawback that it is very difficult to choose an optimal value for  $k$  in practice.

Several papers [25, 26] have proposed using  $K^{th}$  nearest neighbor outlier detection algorithms for the purpose of anomaly detection.

### Density-based

Density-based anomaly detection techniques estimate the density of the neighborhood of each data instance. An instance that lies in a neighborhood with low density is declared to be anomalous while an instance that lies in a dense neighborhood is declared to be normal.

For a given data instance, the distance to its  $K^{th}$  nearest neighbor is equivalent to the radius of a hypersphere, centered at the given data instance, which contains  $K$  other instances. Thus the distance to the  $K^{th}$  nearest neighbor for a given data instance can be viewed as an estimate of the inverse of the density of the instance in the data set and the basic nearest neighbor-based technique can be considered as a density-based anomaly detection technique.

There are three major approaches:

- Local Outlier Factor (LOF);
- Connectivity Outlier Factor (COF);
- Multi-Granularity Deviation Factor (MDEF).

Breunig et al. [27] assign an anomaly score to a given data instance, known as **Local Outlier Factor (LOF)**. For any given data instance, the LOF score is equal to ratio of average local density of the  $K$  nearest neighbors of the instance and the local density of the data instance itself.

These are the steps in order to compute LOF:

For each data point  $q$  compute the distance to the  $K^{th}$  nearest neighbor ( $k$ -distance).

Compute *reachability distance* (reachdist) for each data example  $q$  with respect to data example  $p$  as:

$$reachdist(q, p) = \max\{k - distance(p), d(q, p)\} \quad (2.1)$$

Compute *local reachability density* (lrd) of data example  $q$  as inverse of the average reachability distance based on the *MinPts*, which is the number of nearest neighbors used in defining the local neighborhood of the data example  $q$

$$lrd(q) = \frac{MinPts}{\sum_p reachdist_{MinPts}(q, p)} \quad (2.2)$$

Compute  $LOF(q)$  as ratio of average local reachability density of  $q$ 's  $k$ -nearest neighbors and reachability density of the data record  $q$



$$LOF(q) = \frac{1}{MinPts} \cdot \sum_p \frac{lrd(p)}{lrd(q)} \quad (2.3)$$

In the example show in the figure 2.1, int the neighbor-based approach,  $p_2$  in not considered as outlier, while the LOF approach find both  $p_1$  and  $p_2$  as outliers. The  $K^{th}$  nearest neighbor-based approach may consider  $p_3$  as outlier, but LOF approach doesn't.

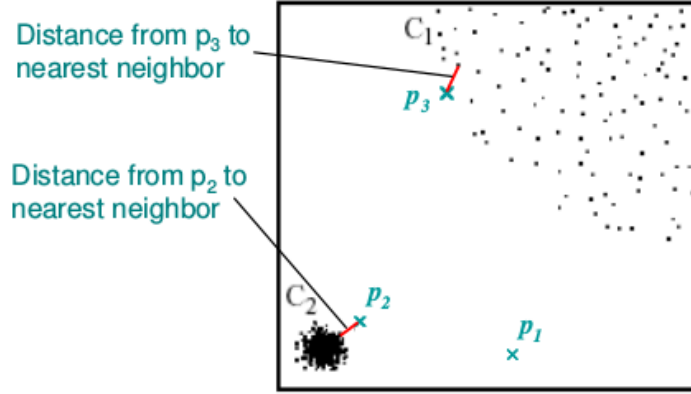


Figure 2.1: Advantages of density-based techniques.

Tang et al. [28] discuss a variation of the LOF, which they call **Connectivity-based Outlier Factor (COF)**. The difference between LOF and COF is the manner in which the  $k$  neighborhood for an instance is computed. In LOF, the  $k$ -nearest-neighbors are selected based on Euclidean distance. This indirectly assumes, that the data is distributed in a spherical way around the instance. If this assumptions is violated, for example if features have a direct linear correlation, the density estimation is incorrect. COF wants to compensate this shortcoming and estimates the local density for the neighborhood using a shortest-path approach, called the *chaining distance*.

In COF outliers are points  $p$  where average chaining distance  $acdist_{kNN(p)}(p)$  is larger than the average chaining distance ( $acdist$ ) of their  $k$ -nearest neighborhood  $kNN(p)$ . COF identifies outliers as points whose neighborhoods is sparse than the neighborhoods of their neighbors.

This leads to the following formula:

$$acdist_G(p) \equiv \sum_{i=1}^r \frac{2(r-i)}{r(r-1)} dist(e_i) \quad (2.4)$$

The smaller  $acdist$ , the more compact is the neighborhood  $G$  of  $p$ . COF is computed as the ratio of the  $acdist$  (average chaining distance) at the point and the mean  $acdist$  at the point's neighborhood.

Similar idea as LOF approach:

A point is an outlier if its neighborhood is less compact than the neighborhood of its neighbors

$$COF_k(p) \equiv \frac{acdist_{N_k(p) \cup p}(p)}{\frac{1}{k} \sum_{o \in N_k(p)} acdist_{N_k(o) \cup o}(o)} \quad (2.5)$$

COF is able to capture transition like in figure 2.2.

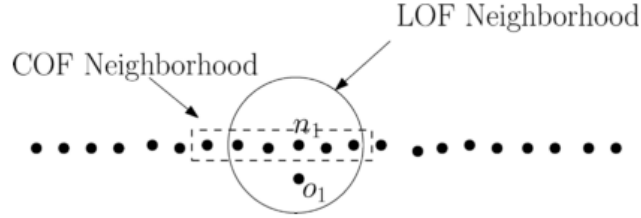


Figure 2.2: Difference between the neighborhoods computed by LOF and COF.

A simpler version of LOF was proposed by Hautamaki et al. [25], which calculates a quantity called **Outlier Detection using In-degree Number (ODIN)** for each data instance. For a given data instance, ODIN is equal to the number of  $k$  nearest neighbors of the data instance which have the given data instance in their  $k$  nearest neighbor list.

Papadimitriou et al. [29] propose a measure called **Multi-Granularity Deviation Factor (MDEF)**, which is a variation of LOF. MDEF for given data instance is equal to the standard deviation of the local densities of the nearest neighbors of the given data instance (including the data instance itself). The inverse of the standard deviation is the anomaly score for the data instance. The anomaly detection technique presented in the paper is called LOCI, which not only finds anomalous instances but also anomalous micro-clusters.

### Clustering-based

Clustering-based anomaly detection techniques rely on the process of building similar objects into clusters and assume that anomalous points lie in sparse and small clusters, or that they are not assigned to any cluster, or that they lie far from their cluster centroid. Clustering is primarily an unsupervised technique though semi-supervised clustering has also been explored lately [30]. Clustering-based anomaly detection techniques can be grouped into three categories.

The first category of clustering-based techniques relies on the following assumption:

**Assumption 1.** Normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster.

**Assumption 2.** Normal data instances lie close to their closest cluster centroid, while anomalies are far away from their closest cluster centroid.

**Assumption 3.** *Normal data instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters.*

**k-means clustering (K-means)** [31] is an algorithm that groups data points into  $K$  clusters by their feature values. First, the  $K$  cluster *centroids* are randomly initialized. Then, each data record is assigned to the cluster with the nearest *centroid*, and the *centroids* of the modified clusters are re-calculated. This process stops then the *centroids* are not changing anymore. Data points are put in the same cluster when they have similar feature values according to a given distance function. Finally, scores of each data point inside a cluster are calculated as the distance to its *centroid*. Data points which are far from the *centroid* of their clusters are labeled as anomalies.

**Local Density Cluster-based Outlier Factor (LDCOF)** [32] estimates the density of clusters assuming a spherical distribution of the data points. The clusters are generated using the K-means clustering algorithm; then, the clusters are separated into small and large groups following the procedure of [14]. For each cluster, the average distance of all its data points to the centroid is calculated, normalized by the average distance of the data points of this cluster to its centroid, and used as anomaly score. Therefore, expected data points result in smaller scores, i.e., close to 1.0, because their densities are as big as the densities of their cluster neighbors. Instead, anomalies will result in larger scores, since their densities are smaller than the densities of their neighbors.

### 2.3.2 Statistical

The underlying principle of any statistical anomaly detection technique is: “An anomaly is an observation which is suspected of being partially or wholly irrelevant because it is not generated by the stochastic model assumed” [33]. Statistical anomaly detection techniques are based on the following key assumption:

**Assumption.** *Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model.*

The **Histogram-based Outlier Score (HBOS)** is a *statistical* approach [34] that generates an histogram for each independent feature of the given dataset. The values of the features of all the available data points are first used to build histograms; at a later stage, for each data point, the anomaly score is calculated as the multiplication of the *inverse heights* of the columns in which each of its features fall. When features are dependent, these dependencies need to be neglected, leading to possible errors. Indeed, the assumption of independence between features makes HBOS a fast algorithm even when dealing with large datasets [8].

The **Robust Principal Component Analysis (rPCA)** technique [35] is based on the Principal Component Analysis (PCA), a popular technique used for dimensionality reduction. PCA is used to detect subspaces in a dataset and has been applied to anomaly detection to identify deviations from the ‘expected’

subspaces, which may indicate anomalous data points. The principal components of PCA are the *eigenvectors* of the covariance matrix, which is computed twice to improve robustness.

### 2.3.3 Classification

Classification [36, 37] is used to learn a model (classifier) from a set of labeled data instances (training) and then, classify a test instance into one of the classes using the learnt model (testing). Classification based anomaly detection techniques operate under the general assumption that a classifier can be learnt from a given feature space that can distinguish between normal or anomalous classes.

The **One-class Support Vector Machine (one-class SVM)** [38] algorithm, differently from the supervised support vector machines (SVMs), is a commonly used method for semi-supervised anomaly detection [12], that aims to learn a decision boundary to group the data points [39]. However, *one-class SVMs can be used for unsupervised anomaly detection*: the one-class SVM is trained with the dataset and then each data point is classified considering the normalized distance of the data point from the determined decision boundary [38].

**Isolation Forest (IF)** [40] is based on the concept of isolation by structuring data points as nodes of a tree, assuming that anomalies are rare events with feature values that differ a lot from expected data points. Therefore, *anomalies are more susceptible to isolation than the expected data points*, since they are isolated closer to the root of the tree instead of the leaves. It follows that a data point can be isolated and then classified according to its distance from the root of the tree.

### 2.3.4 Angle-based

To alleviate the drawbacks of distance-based models in high-dimensional spaces, a relatively stable metric in high-dimensional spaces, angle, was used in anomaly detection [41, 42].

The **Angle-Based Outlier Detection (ABOD)** [43] measures the variance in angle between the difference vectors of a datapoint to the other points and using it as anomaly score with good scalability. Each data point in the dataset is used as the middle point  $p_2$  of a polygonal chain  $(p_1, p_2, p_3)$ , while  $p_1$  and  $p_3$  are any two different values of the dataset,  $p_1 \neq p_2 \neq p_3$ . Then, all the possible angles  $p_1\hat{p}_2p_3$  are measured, and their variance is used to calculate the *Angle-Based Outlier Factor* (ABOF). Expected data points have a large ABOF, while anomalies typically result in very small variance in the angles from couples of points.

The **Fast Angle-Based Outlier Detection (FastABOD)** [43], similarly to ABOD, detects anomalous data points depending on the variance of angles between pairs of distance vectors to other points. However, it works in sub-quadratic time by considering only angles between pairs of neighbours. For each data point, the algorithm first calculates the ABOF to its k-nearest neighbor as the normalized scalar product of the difference vectors of any pair of neighbors. Then, FastABOD ranks the data points according to their ABOF.

The smaller the ABOF, the bigger the probability that the data point is anomalous.

## 2.4 Comparing Anomaly Detectors

Different anomaly detection algorithms usually exhibit different rates of missed (*False Negatives*) and wrong detections (*False Positives*) and, consequently, have different detection capabilities. Although most of such algorithms have a generic, context-independent formulation, they are often more effective to detect specific attacks on specific systems or applications. Moreover, the manifestation of the anomaly is usually different from attack to attack and from system to system.

On comparing anomaly detection algorithms, the authors of [10] used seven algorithms on a single proprietary dataset containing HTTP traffic and providing an open-source intrusion detection system. Similarly, in [44], authors evaluate four algorithms on a single dataset, focusing more on feature selection. Instead, in [11], authors presented a comparative study for intrusion detectors where *k-Nearest Neighbors* (kNN), *Mahalanobis*-based, *Local Outlier Factor* (LOF) and one-class *Support Vector Machines* (SVM) were evaluated using only the DARPA 98 dataset [45] and real network data (for a total of 2 datasets).

Similarly, in [13] authors compared three unsupervised anomaly detection algorithms for intrusion detection: *Cluster-based Estimation*, kNN and *one-class SVM* using network records stored in the *KDD Cup 99* dataset and system call traces from the *1999 Lincoln Labs DARPA evaluation*. The evaluation of four algorithms in [9] presents a review of novelty detection methods that are classified into semi-supervised and unsupervised categories. The algorithms are exercised on ten different datasets regarding medical and general-purpose data. Some of these datasets were also used in [8], where authors presented a comparison of anomaly detection algorithms for multivariate data points. In this case, 19 anomaly detection methods were evaluated in 10 different datasets from different domains, ranging from brain cancer to satellite activity; however, the only attacks dataset is the *KDD Cup 99*.

Lastly, in [14] the authors compared six supervised and unsupervised algorithms for system log analysis using two datasets: the HDFS logs from Amazon EC2 platform and the BGL BlueGene/L supercomputer system at *Lawrence Livermore National Labs* (LLNL). In [10] seven anomaly detectors for intrusion detection systems are tested on HTTP traffic using a proprietary dataset.

In order to protect systems and networks, security specialists are continuously researching mechanisms and strategies that aim at neutralizing attacks or, if not possible, mitigating their adverse effects. The more complex the system is, the more ways and techniques an attacker has to conduct attacks by exploiting vulnerabilities [2]. Different attacks may have common characteristics, that allow organizing the attacks into attack categories. In the literature, several standards [46], [47], open source libraries [48], and research articles [49, 50], [51, 46], provide taxonomies of cyber-attacks that are largely adopted by topic-related studies. As an additional example, in [52], authors classify attacks depending on accountability of users' actions. Depending on *audit trails*,

they defined strategies to face either external or internal penetrators, masqueraders and clandestine users. In [8] it's shown a comparative evaluation of 19 different unsupervised algorithms on 10 datasets from different application domains, not focusing in intrusion detection and type of attacks, also it's not considered a infrastructure to perform the experiments. The ELKI [53] is a environment for developing KDD-Applications supported by index-structures but it's not considering groups of metrics and dataset, or evaluation of metrics. In [54] it's reported on contemporary unsupervised outlier detection techniques for multiple types of datasets and provide a taxonomy framework and two decision trees to select the most suitable technique based on datasets but it's not focusing on the analysis of types of attacks in intrusion detection systems.

# 3

## Study Design

The anomaly-based detectors characterize an expected behavior of the system to be protected, and generate an alert when the current state of the system does not conform with such expectations [12], which are often characterized as an expected value or trend of some indicators, e.g., rate of ICMP requests, number of TCP packages received. Such indicators are monitored through time, generating time series composed by different data points. These data points contain information that is used by the algorithms to represent their notion of expected behavior and, consequently, to detect anomalies.

There is no need for any label information in practice for unsupervised anomaly detection, however they are needed for evaluation and comparison.

Before running initial experiments, each dataset is partitioned to create sub-datasets that contain only a single type of attack labelled in the data.

First, we surveyed the literature to select the most relevant state-of-the-art contributions on anomaly-based intrusion detectors, prioritizing both the most cited works and recent trends. Each work was analysed to extract common elements such as:

- the proposed algorithm;
- the considered dataset(s);
- attack model;
- data filtering or refinement strategies to improve data quality;
- metrics used to scoring performance.

These elements became the *dimensions of analysis*, with the exception of data filtering. In fact, filtering or refinement of datasets is usually algorithm-specific [55]. Figure 3.1 summarizes the process with the three dimensions mentioned above.



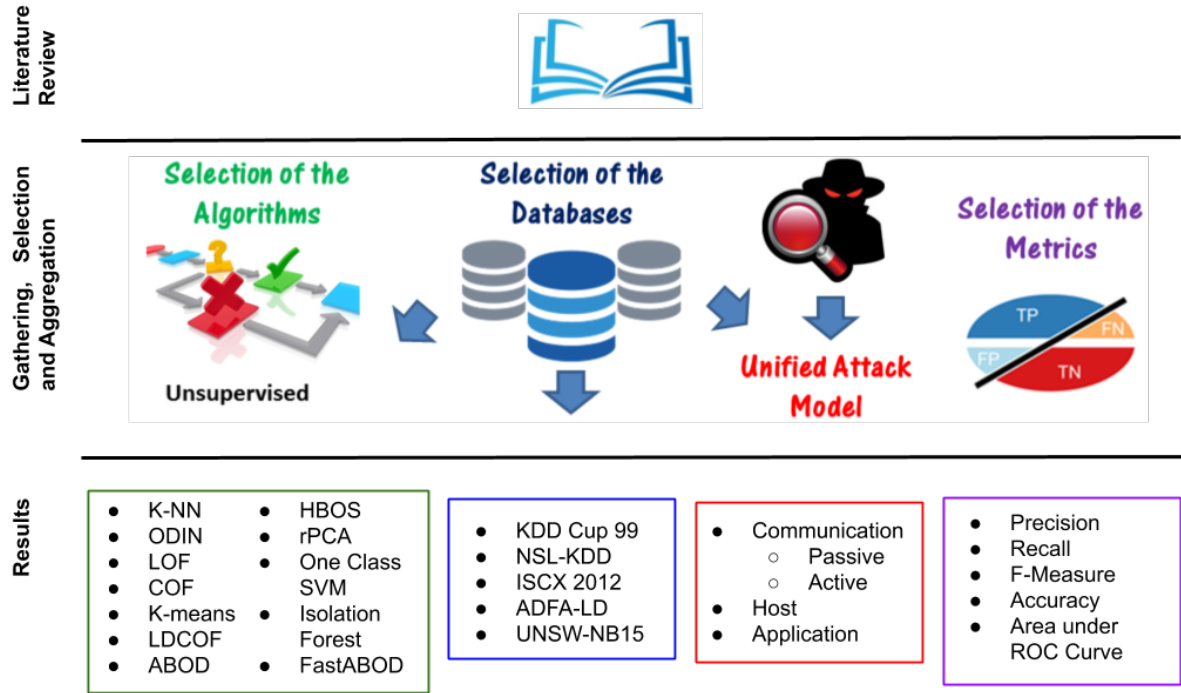


Figure 3.1: Methodology to select, transform and aggregate surveyed data. We show the three dimensions of analysis selection of the algorithms, selection of the datasets, unified attack model, and selection of the metrics.

### 3.1 Datasets

First, the datasets were selected to match the following criteria:

1. They shall contain enough data points to ensure statistical evidence when evaluating the algorithms, e.g., DARPA 1999 dataset [56] was discarded since it contains only 201 data points related to attacks.
2. They shall be labeled on the basis of actual attack activities, i.e., all attacks that occurred should be correctly labeled. Following this criteria, we disregard datasets as MAWI [57], CAIDA [58] or DEFCON [58], that are constituted of sniffed data that is labeled applying detection algorithms: consequently, the labeling may include false positives and negatives.
3. Data points should be complete for all the features in the datasets, to avoid the application of feature recovery strategies [59] that may burden our study by adding another parameter to our analysis.

The table 3.1 summarizes the 5 datasets selected according to the criteria. These datasets are shortly described below, ordered by ascending publication date. Each dataset is matched to an acronym that will be used to refer to it.

[KC] *KDD Cup 99 (1999)* [60]. This is the most popular dataset in the anomaly-based intrusion detection area, being used in recent experiments and surveys [61], [8] and works prior the release of the updated NSL-KDD [62]. Fore this reason, we could not ignore it despite being almost 20-years-old. The



Dataset	Size	Attacks	% Attacks	Features
KDD Cup 99 (KC)	311,028	223,298	71.79	41
NSL-KDD (NK)	22,542	12,832	48.05	42
ISCX2012 (IX)	571,698	66,813	11.68	17
ADFA-LD (AL)	2,122,085	238,160	11.22	1
UNSW-NB15 (UN)	175,341	119,341	68.06	46

Table 3.1: Description of the Selected Data sets

dataset contains the following attacks: *DoS* (Denial of Service), *R2L* (unauthorized access from a remote machine), *U2R* (unauthorized access to *superuser* or *root* functions) and *Probing* (gather information about a network).

[NK] *NSL-KDD (2009)* [63]. This intrusion detection data set was created to solve problems in the *KDD Cup 99* dataset as i) the presence of redundant records in train sets, and ii) duplicates in test sets. The attacks are the same as KC.

[IX] *ISCX (2012)* [58]. It is generated in a controlled testbed environment based on a realistic network and traffic, to depict the real effects of attacks over the network and the corresponding responses of workstations. The data points are labeled in a controlled and deterministic environment that allows distinguishing anomalous activities from expected traffic. Four different attack scenarios are simulated: *infiltration*, *HTTP denial of service*, a *distributed denial of service* by using an IRC botnet, and *SSH bruteforce* login attempts.

[AL] *ADFA-LD (2013)* [64]. Released by the Australian Defence Force Academy, this dataset contains expected and anomalous *Linux system call* traces generated by emulation. The occurrence of *AddUser*, *Java*, *Meterpreter*, *Hydra* *SSH-FTP*, and *Web* attacks is labelled, although not detailed.

[UN] *UNSW-NB15 (2015)* [65]. As ADFA-LD, this dataset was also released by the *Australian Defence Force Academy* in the *University of New South Wales*. Authors used the *IXIA PerfectStorm tool* [66] to generate expected and anomalous network traffic. The following attacks can be simulated by the tool. *Exploits*: the attacker exploits a generic vulnerability, *DoS*: a (Distributed) Denial of Service, *Worms*: a script that replicates itself to spread to other networked computers, *Generic*: a technique that works against all block-ciphers, with a given block and key size, *Reconnaissance*: attack that gather information, *Shellcode*: a small piece of code used as the payload in exploits, and *Backdoors*: a security mechanism is stealthily bypassed to access functionality or data).

## 3.2 Unified Attack Model

The types of attacks for each dataset are classified according to an *unified attack model*. Each dataset uses inconsistent naming and grouping of categories of attacks, so it's defined an attack model that is generic enough to aggregate all

the attacks in the different datasets. In [49] the authors partition the attacks defined in the *NIST 800-53* [46] standard into i) **Communication**: attacks directed to network interfaces, ii) **Host**: malware or malicious code injected in a target host exploiting vulnerabilities of the operating system, iii) **Application**: attacks that exploits vulnerabilities of (web)services, and iv) **Generic**, which contains everything that is not related to the first three categories. The work [49] provides a simplified abstraction of the existing *NIST* attack list, which we can use as a starting point to build our generic attack model.

None of the specific attacks falls into the **Generic** category of [49], which is discarded. However, the datasets contain data related to several attacks that mainly involve the communication channel with different means and purposes. To differentiate among these attacks, the **Communication** category was splitted as defined in [49] into two separated categories (see rows 1 and 2 of table 3.2). The subcategories are i) **communication passive**, that represents attacks directed to gather or steal data through the passive observation of the communication channel, and ii) **communication active**, which represents attacks which use the communication channel as a way to send malicious data or requests to the target system. This ultimately results in an attack model composed of four categories, and is show in table 3.2. Along with the description of the attacks, the last column of Table 3.2 maps all the different attacks referenced in the datasets to our attack categories. As example, *Exploits* attacks of UNSW-15 fall into the *Application* category, as it can be observed at the bottom right cell of the table. Attacks with different labels, or reported in different datasets, that resemble the same attack are merged into a unique attack, e.g., DoS attacks, which can be found in both NK and IX datasets.

### 3.3 Metrics for Evaluation

Comparing the anomaly detection performance of unsupervised anomaly detection algorithms is not as straight forward. In contrast to simply compare as accuracy value or precision/recall. Here the metrics described in [15] that were used more frequently in the surveyed studies and that are applied in the evaluation. It's not take into account the time-related metrics as the execution time and the detection time, because they are usually dependent on the hardware resources available for a given system and on the specific implementation of the algorithm. The selected algorithms to understand if and how they are able to identify anomalies due to attacks, without including metrics that may be dependent on the quality of the implementation and on the available resources.

- **Precision.** Considering *TP* (true positives) as the amount of detected anomalies that corresponds to manifestations of attacks and *FP* (false positives) as the amount of detected anomalies that do not, *Precision* is defined as the fraction of *TP* among the whole set of true and false positives.

$$Precision : P = \frac{TP}{TP + FP} \quad (3.1)$$

Attack Category	Source [49] Category	Description	Mapping of (Dataset) Attack
Communication - Passive	Communication	Attacks which targets the communication channel to gather information without active damage	(KD - NK) Probing, (IX) Infiltration, (UN) Reconnaissance, (UN) Analysis
			(IX) Bruteforce, (KD - NK - IX - UN) DoS, (IX) DDoS, (UN) Fuzzers, (UN) Backdoors
Communication - Active	Communication	Attacks that are conducted through the communication channel to actively damage the system	(KD - NK) U2R, (KD - NK) R2L, (UN) Worms, (UN) Shellcode, (UN) Malware
Host	Host	Attacks which targets a given host by installing malicious code into it	(UN) Exploits, (UN) Generic, (AL) AddUser (AL) Java, (AL) Meterpreter, (AL) Web, (AL) Hydra
Application	Application	Attacks which targets a given application aiming at executing malicious code by penetrating interfaces	

Table 3.2: The Unified Attack Model developed to categorize attacks from the five datasets.

- **Recall.** Usually presented together with Precision. It is defined as the ratio of  $TP$  over the union of  $TP$  and the undetected anomalies (false negatives,  $FN$ ) .

$$Recall : R = \frac{TP}{TP + FN} \quad (3.2)$$

- **F-Score and F-Measure.** The  $F-Score(\beta)$  metric combines both Precision and Recall by using a parameter  $\beta$ : when  $\beta > 1$ , Recall is weighted more than Precision, as shown in the equation below.

$$F-Score(\beta) : F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \quad (3.3)$$

More in detail,  $F-Measure$ , or rather  $F_1$  Score, is defined as the balanced mean of Precision and Recall [15]. This metric is adopted when data analysts evaluate FPs and FNs as equally undesired.

$$F-Measure : F_1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (3.4)$$

- **Accuracy.** Defined as the ratio of correct detections (both true positives and true negatives) among the total amount of examined data points. This is a way to aggregate the positive and negative scores into a unique metric.

$$\text{Accuracy : } ACC = \frac{TP + TN}{TP + FP + TN + FN} \quad (3.5)$$

- **Area Under ROC curve (AUC).** *ROC curve* is a graphical plot that represents the performance of binary classifiers when their discrimination thresholds vary: the curve is depicted by plotting R against the false positive rate. A high value of the area underlying the ROC curve usually indicates that the identified algorithm suits the target dataset [15], [67].

## 3.4 Anomaly Detection Techniques

### 3.4.1 Selection of the Algorithms

Considering the amount of existing anomaly detection algorithms out there described in the literature, and focusing on dependable and security systems. The first step was to start looking for some of the most relevant anomaly detection algorithms through the literature survey. This required to devise a rationale for the selection of algorithms, the datasets, metrics in order to define a benchmark.

The selection of anomaly detection algorithms was a systematic process where the state of art algorithms were analyzed. Several well-known anomaly detection algorithms of different categories and intended to detect different anomalies in different kinds of systems were analyzed. Algorithms were selected according to the following criteria:

- Unsupervised algorithms were selected. There are many approaches such as Supervised Anomaly Detection which describes the setup where the data is composed by fully labeled training and test data sets. Semi-supervised Anomaly Detection also uses training and test datasets, whereas training data only consists of normal data without any anomalies. The basic idea is, that a model of the normal class is learned and anomalies can be detected afterwards by deviating from that model, but the approaches that are more widely applicable are the unsupervised approaches as they do not require labeled data used as a training data for either normal or anomalous populations;
- Several unsupervised anomaly detection algorithms were identified during the survey activity. A total of 12 algorithms.
- The selection consists of only algorithms that were applied in dependable or secure systems and based on the 3.2 categorization:

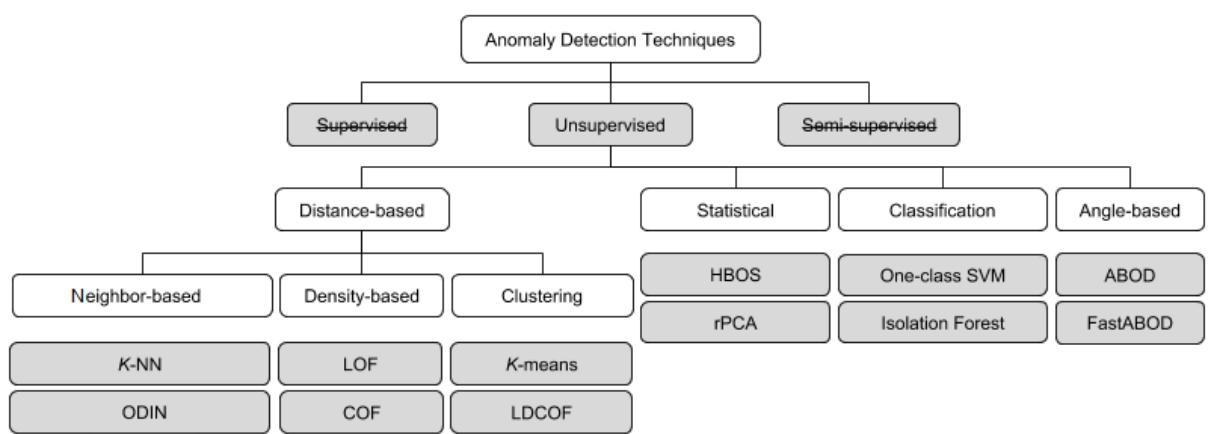


Figure 3.2: A taxonomy of unsupervised anomaly detection techniques.

## 4

# The Framework for Quantitative Comparison of Anomaly Detection Techniques

In this section, it's presented the main design principles as well as the structure of the framework for current evaluation and execution of anomaly detection techniques. Also, it's describes each module that composes the framework.

### 4.0.1 General Concepts About Frameworks

Frameworks are tools used to generate applications for a family of related problems in a specific domain [68]. The choice to use an existing framework or developing new application generator reflects on whether the framework can offer design and code reuse. Frameworks contain fixed and flexible points known as **frozen spots** and **hot spots**, respectively. Hot spots are extension points that allow developers to create a new application from the framework process of instantiating. In this case, developers should create specific application code for each hot spot through the implementation of abstract classes and methods defined in the framework. Frozen spot consist of the framework's kernel, corresponding to all the fixed parts, previously implemented that would hardly change.

In addition to the characteristics cited from the definitions above, others should be listed as the extension technique [69]:

- **White box framework:** They are strongly linked to the characteristics of the languages, to make the customization they use inheritance. It requires a good understanding of the framework for creating an application;
- **Black box framework:** They are instantiated from some kind of configuration, such as an XML file for example. These frameworks rely primarily on composition (classes or components) and delegation to perform customization. It does not require understanding of internal details to produce an application;
- **Gray Box Framework:** These frameworks are designed to avoid the disadvantages presented by the white box and black box framework, allowing

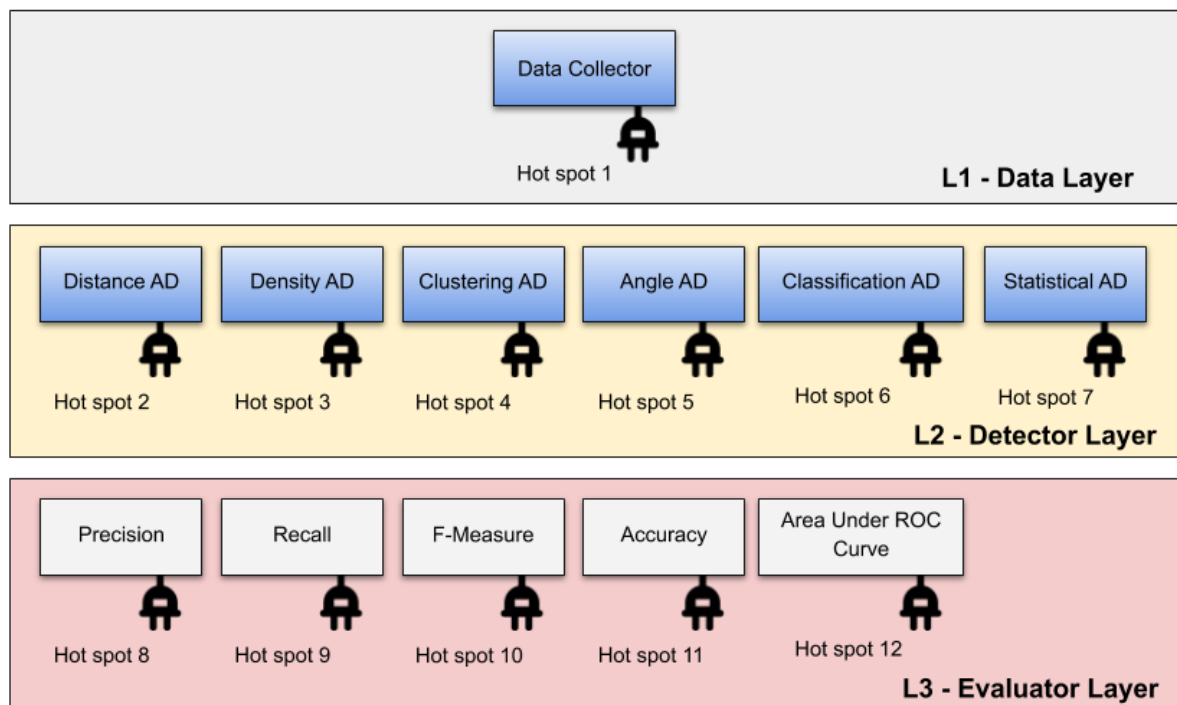


Figure 4.1: The architecture with its three layers.

some degree of flexibility and extensibility without exposing unnecessary internal information.

## 4.1 Designing a Framework for Anomaly Detection Techniques

The problem dealt with in this work has many points of variability, so a framework would better address this problem. The development was focused on white box framework and the advantage of developing such kind of framework is that the user has more control of the details of the code to create an application and at the same time is not "stuck" to what is already defined (as in this case, black box). The framework consists of three layers: the *data layer*, the *detector layer* and the *evaluator layer*, each one with its well-defined functionality 4.1.

The layer L1 and L2 are composed of hot spots where the data collector for L1 will indicate what's the input data for the analysis, as seen in table 3.1 including some of the datasets most used in the literature, but the inclusion of others datasets can be done. The same happens for layer L2 where we can see in figure 3.2 the trend algorithms for anomaly detection implemented so that this set of algorithms can be used in the literature for the quantitative analysis even with other ones implemented for a specific category described in L2. The frozen spots in L3 can be extended by hot spots composing metrics for evaluation.

The data layer is responsible to manage information about the datasets to be



analysed and insert into a database with a grouped data from different sources of labeled datasets based on intrusion detection intended to be used by the anomaly detector layer which will have some implemented anomaly detection algorithms and will provide an infrastructure to implement new ones for further analysis and comparison between them, and a module responsible to collect all the results of the algorithms performed and showing the calculated metrics seen in 3.3 grouped and processed to generate the graphs.

#### 4.1.1 The Data Layer

Here, the data module reads the datasets for the analysis in principle a format of CSV or ARRF files, each dataset is partitioned in order to create sub-datasets, isolating single types of attacks, for example, in the file 4.2 the attack name is **fuzzers** and the dataset is the *UNSW-NB15*, generating datasets as seen in figure 4.3 and 4.4 for the dataset *UNSW-NB15* which will be described in section 3.1) and creating a unlabeled version of the dataset with the normal data for algorithm execution and the faulty dataset which is labeled for the future comparison with the output of the algorithm executed.

```
.950813,255,2279165235,2713761123,255,0.2003,0.105871,0.094429,53,45,0,0,2,1,1,1,2,0,0,1,2,0,"Normal","no"
730437,255,3806211560,3702903947,255,0.247578,0.114124,0.133454,53,44,0,0,2,1,1,1,2,0,0,1,2,0,"Normal","no"
5234,255,3357951987,2103193873,255,0.177967,0.065765,0.112202,59,45,0,0,3,1,1,1,3,0,0,0,1,3,0,"Normal","no"
.902422,255,4015132233,3506072718,255,0.194394,0.118837,0.075557,59,44,0,0,3,1,1,1,1,3,0,0,0,1,3,0,"Normal","no"
22,255,3546145968,1311075600,255,0.199991,0.081543,0.118448,53,45,0,0,2,1,1,1,2,0,0,0,1,2,0,"Normal","no"
.999609,255,1715283593,4192780118,255,0.127664,0.066214,0.06145,53,57,0,0,3,1,1,1,3,0,0,0,3,3,0,"Normal","no"
321656,255,1386473382,3528822378,255,0.124404,0.074397,0.050007,53,44,0,0,3,1,1,1,3,0,0,0,1,3,0,"Normal","no"
16.475406,255,1520755369,2873882484,255,0.224244,0.110448,0.113796,53,57,0,0,3,1,1,1,3,0,0,0,1,3,0,"Normal","no"
.822973,255,2162593193,2671705107,255,0.162189,0.101743,0.060446,218,57,0,0,3,1,2,2,1,3,0,0,0,5,3,0,"Fuzzers","yes"
1.756141,255,1702919478,632938280,255,0.148118,0.107154,0.040964,133,57,0,0,2,1,2,1,1,3,0,0,0,2,2,0,"Fuzzers","yes"
385,10979.056,255,3730612194,2988658491,255,0.193921,0.085057,0.108864,56,57,0,0,3,1,1,1,3,2,2,0,1,3,0,"Fuzzers","yes"
1.781102,255,2181035859,3743429902,255,0.215281,0.104195,0.111086,58,57,0,0,2,1,1,1,2,1,1,0,1,2,0,"Fuzzers","yes"
.3421550673,2671238035,255,0.103949,0.043058,0.060891,52,45,0,0,3,1,3,3,1,3,0,0,0,3,3,0,"Fuzzers","yes"
1789,255,2687291769,3006365151,255,0.124701,0.057386,0.067315,60,45,0,0,4,1,1,1,4,0,0,0,1,4,0,"Fuzzers","yes"
.349031,255,177151243,1059375229,255,0.111537,0.057679,0.053858,54,57,0,0,4,1,1,1,4,1,0,1,4,0,"Fuzzers","yes"
740117,255,1398008940,881080843,255,0.18189,0.103433,0.078457,54,57,0,0,3,1,1,1,3,1,1,0,2,3,0,"Fuzzers","yes"
55,433146084,713507480,255,0.1341,0.068091,0.066009,53,45,0,0,2,1,1,1,2,0,0,0,1,2,0,"Fuzzers","yes"
3.67,255,505094852,3433063302,255,0.144251,0.06893,0.075321,111,45,0,0,6,1,4,3,3,6,0,0,0,3,7,0,"Fuzzers","yes"
1328,255,307047675,1832579755,255,0.08662,0.036545,0.050075,256,45,0,0,3,1,1,1,3,0,0,0,1,3,0,"Fuzzers","yes"
14309,255,1770033310,1606265084,255,0.12496,0.071502,0.053458,53,45,0,0,3,1,3,3,1,3,0,0,0,3,3,0,"Fuzzers","yes"
.165,255,1399874703,1117158503,255,0.112325,0.045494,0.066831,84,45,0,0,5,1,4,4,3,5,0,0,0,5,5,0,"Fuzzers","yes"
.3,0,"Fuzzers","yes"
58.302,255,2444965316,2596235514,255,0.084124,0.065221,0.018903,65,45,0,0,5,1,4,4,3,4,0,0,0,4,4,0,"Fuzzers","yes"
11258,255,3219100110,2572559002,255,0.100124,0.055093,0.045031,89,45,0,0,4,1,1,1,4,0,0,0,1,4,0,"Fuzzers","yes"
.143.089063,255,4084231089,1277456825,255,0.191173,0.101244,0.089929,58,57,0,0,3,1,1,1,2,1,1,0,1,2,0,"Fuzzers","yes"
7461,255,1481164948,3341150864,255,0.145343,0.075614,0.069729,52,45,0,0,4,1,2,2,1,4,0,0,0,2,4,0,"Fuzzers","yes"
141,255,560297479,3115030861,255,0.120855,0.071037,0.049818,77,45,0,0,4,1,1,1,4,0,0,0,1,4,0,"Fuzzers","yes"
153.017578,255,1726233151,1389133376,255,0.13292,0.084513,0.048407,58,57,0,0,2,1,1,1,2,1,1,0,3,2,0,"Fuzzers","yes"
3.67,255,505094852,3433063302,255,0.144251,0.06893,0.075321,111,45,0,0,6,1,4,3,3,6,0,0,0,3,7,0,"Fuzzers","yes"
3125,255,1428272289,3177964949,255,0.103845,0.047573,0.056272,61,45,0,0,4,1,2,2,1,4,0,0,0,2,4,0,"Fuzzers","yes"
1,0,"Fuzzers","yes"
305219,255,1918197902,1768643115,255,0.210426,0.115806,0.09462,864,44,0,0,5,1,2,2,2,5,0,0,0,2,5,0,"Fuzzers","yes"
1.62693,255,3886039819,2768942167,255,0.148089,0.084184,0.063905,53,57,0,0,2,1,1,1,1,2,0,0,0,4,2,0,"Fuzzers","yes"
5.153125,255,3422654858,1022150064,255,0.161498,0.090672,0.070826,789,55,0,0,3,1,1,1,1,3,0,0,0,2,3,0,"Fuzzers","yes"
```

Figure 4.2: Sequences of time series inputs for the fuzzers type of attack.



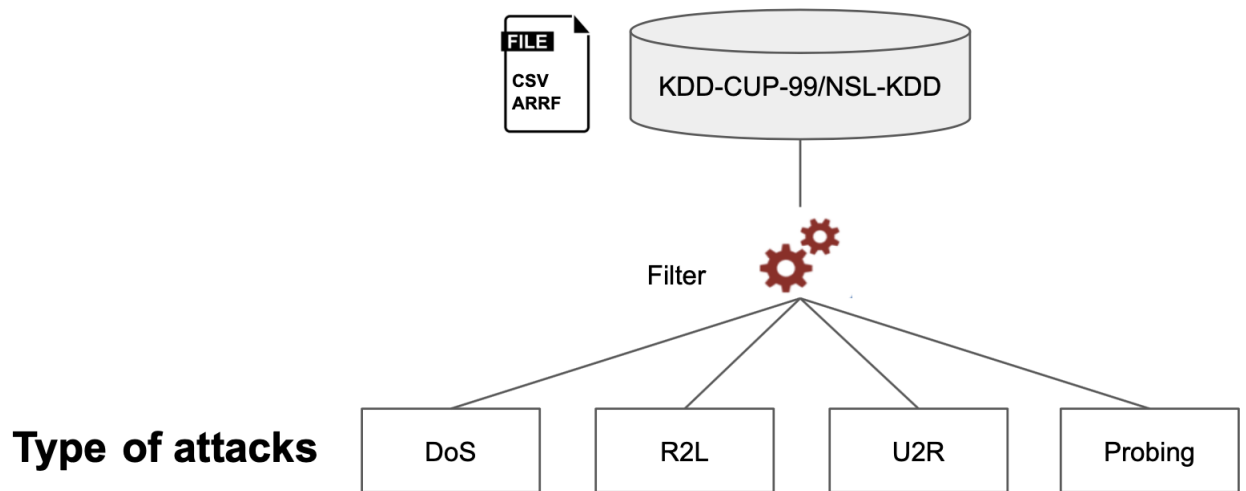


Figure 4.3: Creation of the sub-datasets, isolating single types of attacks for KDD-CUP-99 and NSL-KDD datasets.

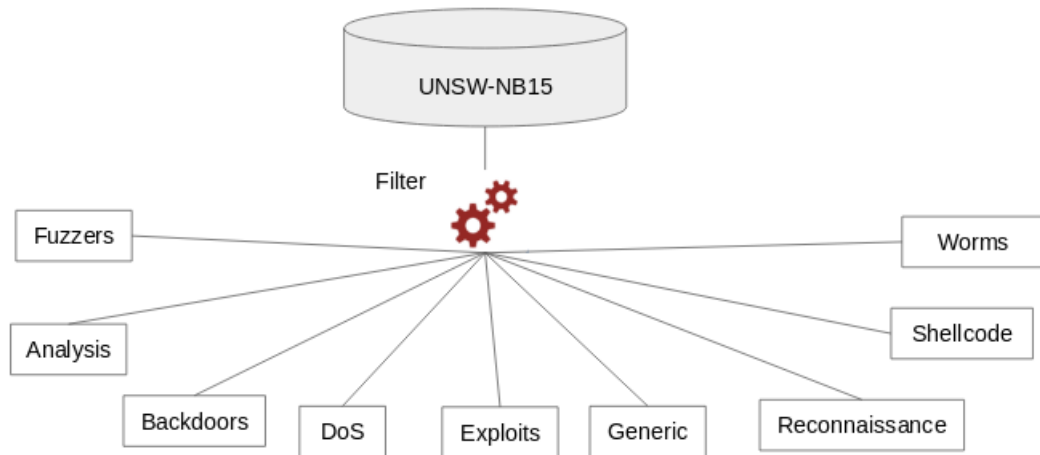


Figure 4.4: Creation of the sub-datasets, isolating single types of attacks for UNSW-NB15 dataset.

#### 4.1.2 The Detector Layer

The operator module includes a set of **anomaly detectors** which is basically the implementation of a set of algorithms for anomaly detection. An anomaly detector is assigned to a given data series, and evaluates if the current value of the selected data series is anomalous or expected following a set of given rules. More anomaly checkers can be created for the same data series: indicators data related to the same time instant is aggregated in a snapshot, given a snapshot as input, each anomaly checker produces an anomaly score. The individual outcomes of each selected anomaly detector is then combined to decide if an anomaly is suspected for the current snapshot. The structure of the detector layer is shown in the class diagram in 4.5. All anomaly detector implements the method `detect()` from `AlgorithmDetector` and are subclasses of its respective families, the attribute  $k$  means the number of clusters as a

input for an algorithm such as *KMeansDetector* or *t* for the number of trees when using *IsolationForestDetector*.

The class *DetectionConfig* is responsible for defining the configuration needed to perform the execution of an algorithm detector with the following parameters: the String describing the algorithm required by the user, the String describing the path to the input dataset, if the header of the file is present it's true and false otherwise, the String with the path for the output file and the number of attributes to select from the dataset in dimensionality reduction/feature selection:

```
1 // Create the detection configuration and detection mode with the given
  parameters
2 DetectionConfig detectionConfig = new DetectionConfig(algorithm, inputDir,
  isHeaderPresent, scoresDir, attributesNum);
```

After that the *detectionConfig* is used by the class *DetectionMode* as an input which will be responsible to execute the anomaly detectors and collection of information, these classes are shown in diagram in 4.6.

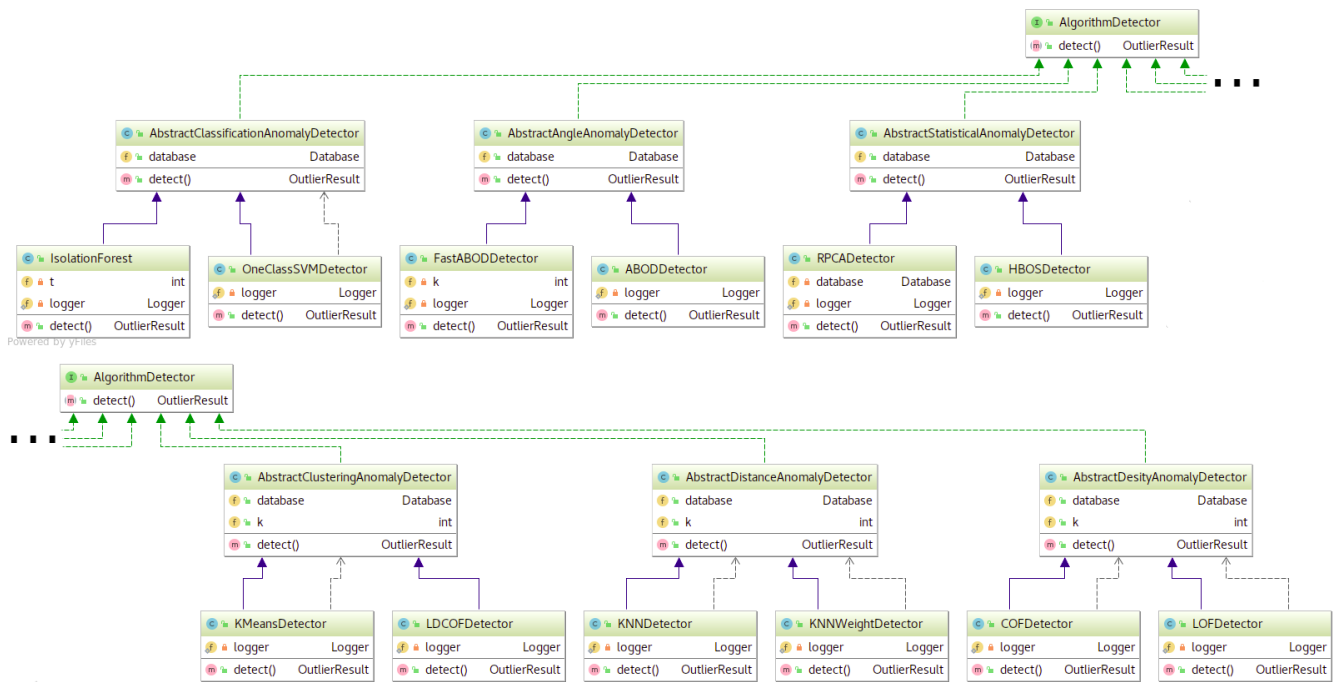


Figure 4.5: Class Diagram of Algorithm Detectors the Detector Module.

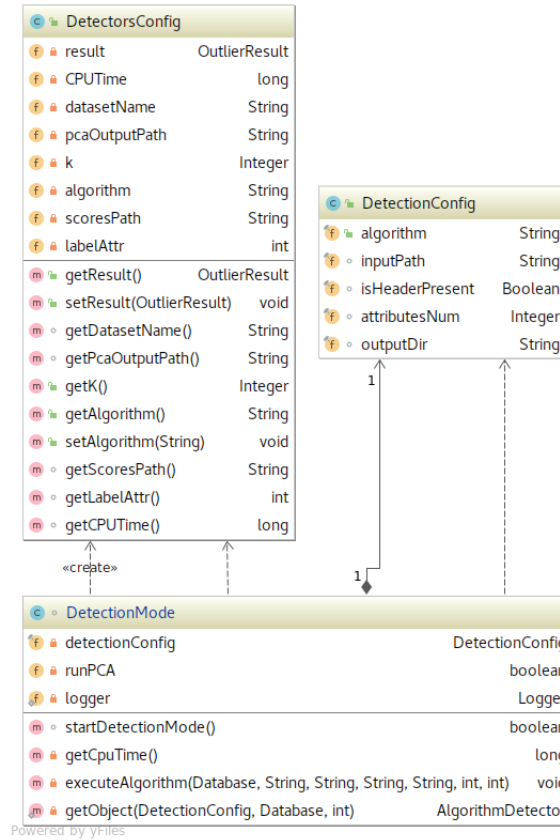


Figure 4.6: Class Diagram of the Algorithm Detectors Execution at Detector Layer.

### 4.1.3 The Evaluator Layer

In this layer, the output indicating how the anomaly detection algorithms performed in used and these metrics are mainly based on boolean *anomaly/-expected* labels assigned to a given data point. However, when providing an output, algorithms may not provide a *boolean* answer: instead, algorithms usually provide a numeric anomaly score, which indicates how anomalous a data point is about the others. In our study, we define such thresholds relying on the *Interquartile Range*, that is the difference of the two *quantiles*  $Q3$  and  $Q1$  that was extensively studied in [70], and adopted as a recommended practice when dealing with numerical data, as in [35]. Besides, we do not account for time-related metrics as the execution time and the detection time, because they are usually dependent on the hardware resources available for a given system and on the specific implementation of the algorithm.

## 4.2 Execution Sequence

The framework works as follows, starting at the *data layer*, the first part is to read the dataset, normally an *ARRF* or *CSV* file, filter the type of intrusion attacks as shown in section 3.2 to be detected, the optional part to perform the

feature selection attributes by reducing the features that don't change values over time and are not important for the evaluation as shown in the figure bellow:

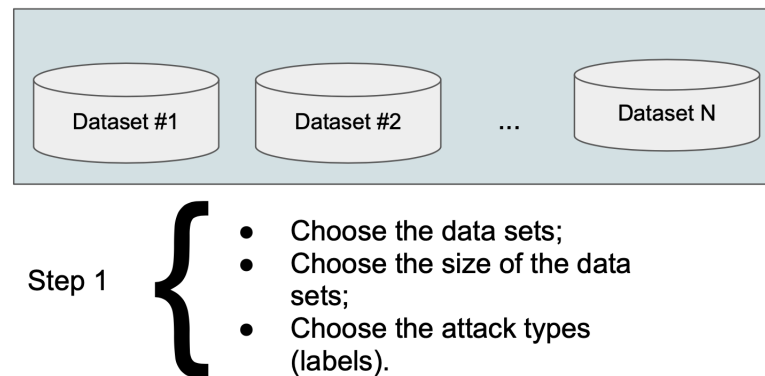


Figure 4.7: First step: *data layer*

The next step 4.8 is in the *detector layer*, run the anomaly detection algorithms, and generate the scores for each algorithm in respect to each of the datasets for the evaluation.

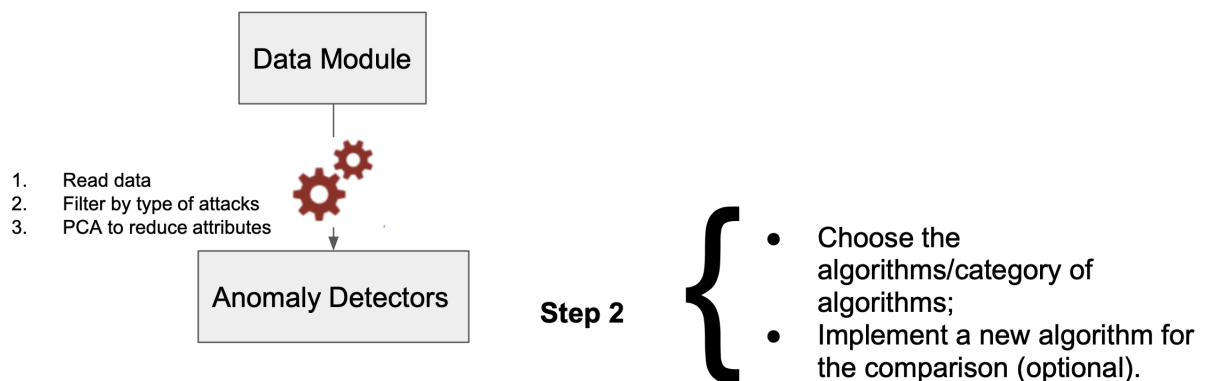


Figure 4.8: Second step: data layer and detector layer

Lastly, the third step 4.9 in the *evaluator layer* where the scores computed to metrics seen in section 3.3, in the **detector layer** these metrics will be saved into some collection in *MongoDB* [71] database for fast analytics, such as comparing metric scores. Ultimately, metric scores are used to plot ROC curves, and then to calculate the *Area under the Curve* (AUC).

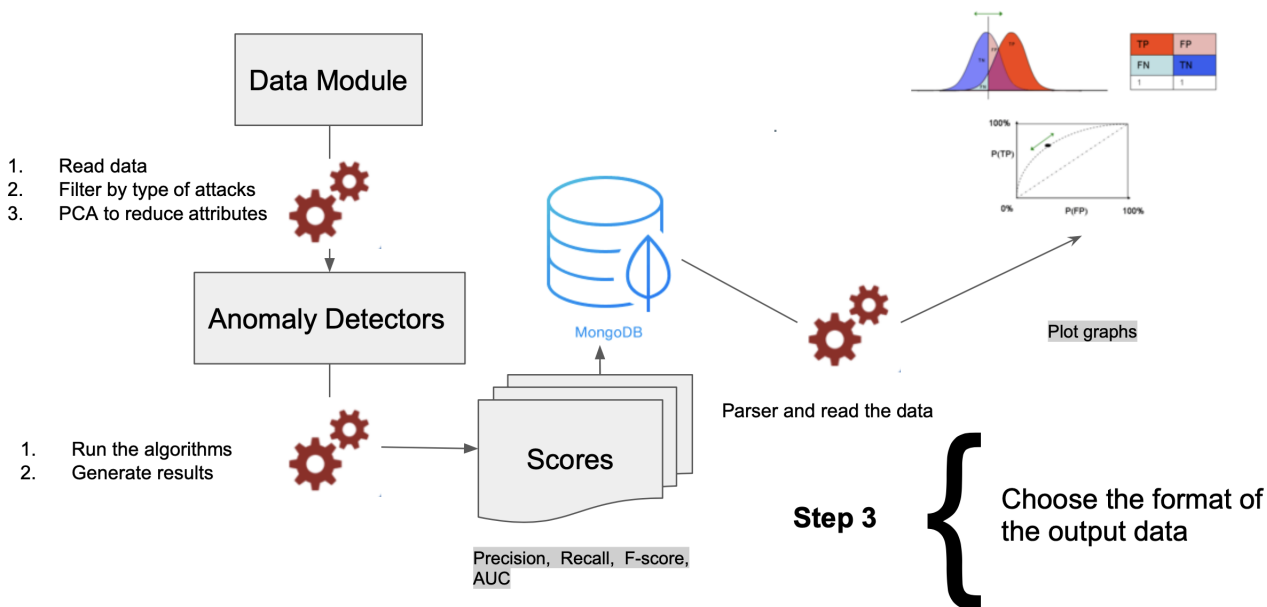


Figure 4.9: Final step. data layer, detector layer and evaluator layer

# 5

## Results and Discussion

For this first experiment it was retrieved available public implementations of the selected algorithms. *KMeans*, *kNN*, *ODIN*, *LOF*, *COF*, *ABOD*, *FastABOD* and *OneClass SVM* are extracted from the ELKI framework [53]. The other implementations of algorithms come from RapidMiner [72]. The 5 datasets are processed by converting - where needed - nominal variables to numerical to increase the amount of usable features, without affecting the semantics of the datasets.

Parameters tuning are employed to find an optimal setup of the algorithm. Tuning is performed by i) first, running different combinations of parameters; ii) then, comparing results for the different parameters. For example, we run KNN and KNN-dependent algorithms, i.e., ODIN, FastABOD, with  $k \in \{1, 2, 3, 5, 10, 20, 50, 100\}$ . The discussion in the following sections will also elaborate on the relevance of the choice of parameters.

The data generated during the execution of the algorithms was initially stored in CSV files, and successively in a *MongoDB* [71] database for fast analytics, such as comparing metric scores. Ultimately, metric scores are used to plot ROC curves, and then to calculate the *Area under the Curve* (AUC).

The algorithms, the database and the *Mongo* analytics are executed on three Intel Core i7, 32GB of RAM and 256GB of SSD storage servers. Overall, computing all the scores required approximately one month of 24 hours of execution. Due to known computational complexity problems [43], ABOD and FastABOD algorithms are ran only on a portion of the datasets. The portions are chosen considering the biggest subset of the dataset that do not escalate in heap memory errors, i.e., 4% for KC, 53% for NK, 24% for AL, 5% for IX, 6% for UN.

### 5.1 Experiment Comparison and Previous Results

Here is presented a previous comparison between algorithms relying on publicly available datasets and implementations.

Algorithm	# Setups	Family	AUC (ROC)	Precision	Recall	Accuracy	F1
Isolation Forest [40]	8	Classification	$37.2 \pm 0.4$	$99.9 \pm 0.3$	$99.3 \pm 0.4$	$99.7 \pm 0.3$	$99.6 \pm 0.3$
One-Class SVM [38]	1	Classification	$53.4 \pm 2.9$	$96.6 \pm 3.2$	$99.3 \pm 0.0$	$96.2 \pm 3.2$	$98.0 \pm 1.9$
COF [73]	8	Density-Based	$48.8 \pm 1.7$	$93.6 \pm 3.4$	$97.8 \pm 0.1$	$91.7 \pm 3.1$	$95.7 \pm 2.0$
ODIN [62]	8	Neighbour-Based	$49.9 \pm 1.7$	$96.6 \pm 2.4$	$99.9 \pm 0.4$	$89.8 \pm 1.6$	$94.6 \pm 1.1$
HBOS [34]	1	Statistical	$57.8 \pm 5.5$	$92.6 \pm 5.8$	$99.5 \pm 4.3$	$89.2 \pm 4.7$	$94.3 \pm 4.8$
rPCA [35]	1	Statistical	$55.0 \pm 4.0$	$97.5 \pm 3.4$	$95.0 \pm 1.0$	$83.1 \pm 3.2$	$90.6 \pm 2.0$
LOF [74]	8	Density-Based	$50.0 \pm 1.3$	$96.6 \pm 3.5$	$88.0 \pm 1.1$	$81.3 \pm 3.1$	$89.6 \pm 2.1$
LDCOF [72]	8	Clustering	$49.9 \pm 2.3$	$82.4 \pm 1.8$	$94.4 \pm 0.2$	$77.9 \pm 1.5$	$87.4 \pm 0.7$
KNN [3]	8	Neighbour-Based	$35.9 \pm 6.7$	$91.9 \pm 5.8$	$75.1 \pm 3.4$	$71.4 \pm 4.0$	$82.8 \pm 4.3$
K-Means [31]	8	Clustering	$54.4 \pm 8.9$	$95.7 \pm 5.3$	$68.5 \pm 2.8$	$65.6 \pm 3.4$	$78.3 \pm 3.5$
ABOD [43]	8	Angle-Based	$90.5 \pm 7.8$	$69.2 \pm 8.1$	$92.4 \pm 8.3$	$90.0 \pm 1.8$	$75.5 \pm 10.2$
FastABOD [43]	15	Angle-Based	$86.4 \pm 9.2$	$90.6 \pm 7.8$	$77.4 \pm 5.3$	$67.6 \pm 3.2$	$74.7 \pm 6.1$

Table 5.1: Metric scores (*median*  $\pm$  *std*) for the 12 algorithms, ordered by F1 score

### RQ1: Is there an algorithm (or a family) that performs better than the others?

Table 5.1 shows the results obtained by running all the 12 algorithms in our 5 datasets, ranked by F1 scores. We report the median and the standard deviation scores for each metric, and we aggregate the data by considering each algorithm on all the attacks and all the datasets separately. We observe that the first two algorithms belong to the *classification* family. In fact, both **Isolation Forest** and **OneClass SVM** show good scores for anomaly detection: Precision, Recall and Accuracy scores are above 96%. Opposed to classification algorithms, *angle-based* algorithms show poor results for the F1 scores on our datasets. This could be partially explained considering that training of such algorithms was performed by using just a portion of the datasets, negatively affecting their ability to characterize an expected behavior and highlight anomalies.

### RQ2: Is there an algorithm (or a family) that is less dependent on the choice of parameters?

This RQ is related to the choice of the optimal parameters of the algorithms, and it can be explained with the aid of Table 5.1 and Figure 5.1. In particular, the scores used to build the table and to plot the graph refer to the median scores related to the best parameter setup for a given algorithm. Each algorithm has its own parameters, e.g., the size of neighbourhood  $k$  in **KNN**, **ODIN** and **FastABOD**. Such parameters have to be explored to find an optimal setup, in order to use the algorithms at the best of their detection capability. To evaluate the impact of such parameters it repeats the experiments using the same algorithm, but with different parameters values. The number of possible setups are reported in Table 5.1. The scores obtained by using each of these setups are used to build the ROC Curve and the AUC Score. As expected, such score is low for classification algorithms (see Figure 5.1), which

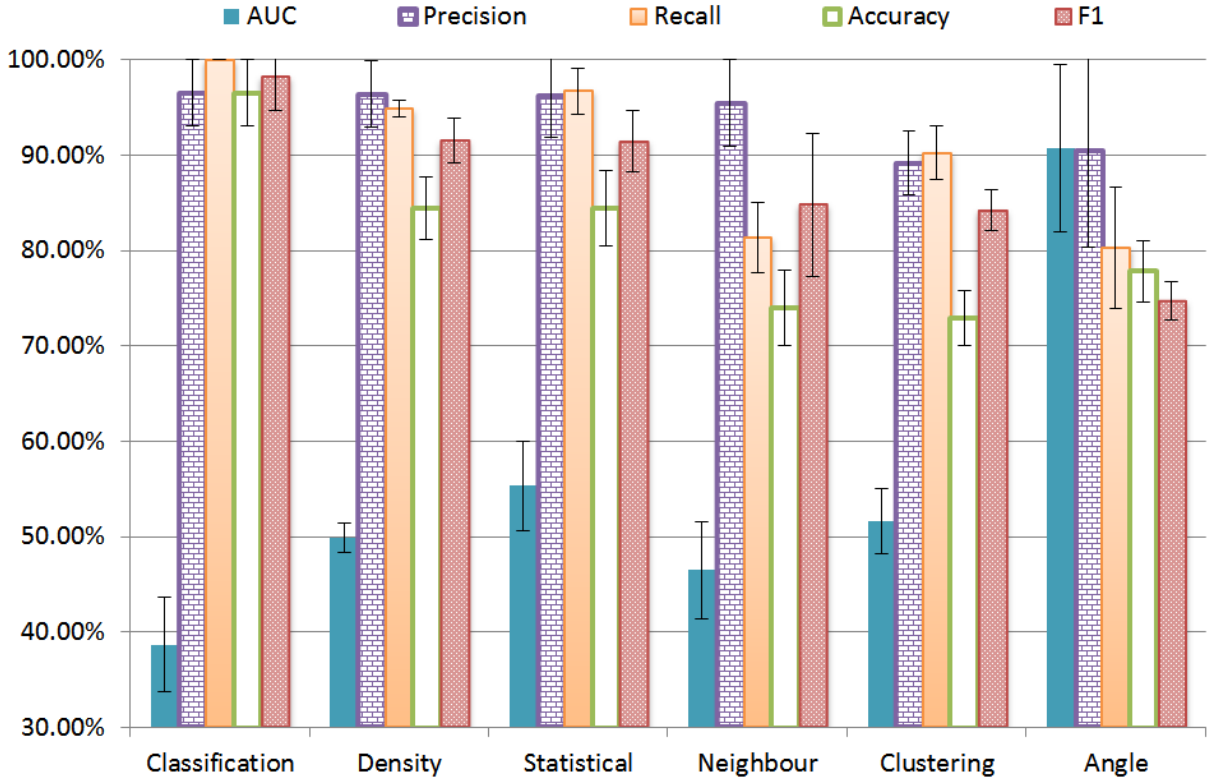


Figure 5.1: Results on all the datasets and all the attacks, grouped by algorithms families. Columns report median scores, while error bars depict the standard deviation

have several configurable parameters. This has strong consequences on our analysis: despite the algorithms shown a very low number of false positives and false negatives, i.e., accuracy and F1 are almost perfect, but classification algorithms strongly depend on their parameters values, and therefore cannot be always considered as an optimal solution.

Surprisingly, it turns out that angle-based scores are not heavily influenced by the choice of such parameters, showing very high AUC scores, i.e., an average of 85%, compared to the others which are instead mostly around 50%. This remarkable difference can be explained as a combination of two factors. First, and more important, such algorithms have few configurable parameters, and the way the ABOF is calculated makes them structurally 'robust' to a wrong choice. Second, such big difference could also be linked to the selection of possible parameters values: the AUC scores is high if changing the value of a parameter does not significantly impact on the outcome of the analysis.

### **RQ3: Is there an attack (or a category of attacks) that is more difficult to detect than the others?**

Differently from Table 5.1 and Figure 5.1, the results can be aggregated by considering the attacks, or the attack categories, as dimensions of the analysis. This allows discussing the median scores of our 12 algorithms, for the anomalies generated by each specific attack. Figure 5.2 shows Precision, Re-



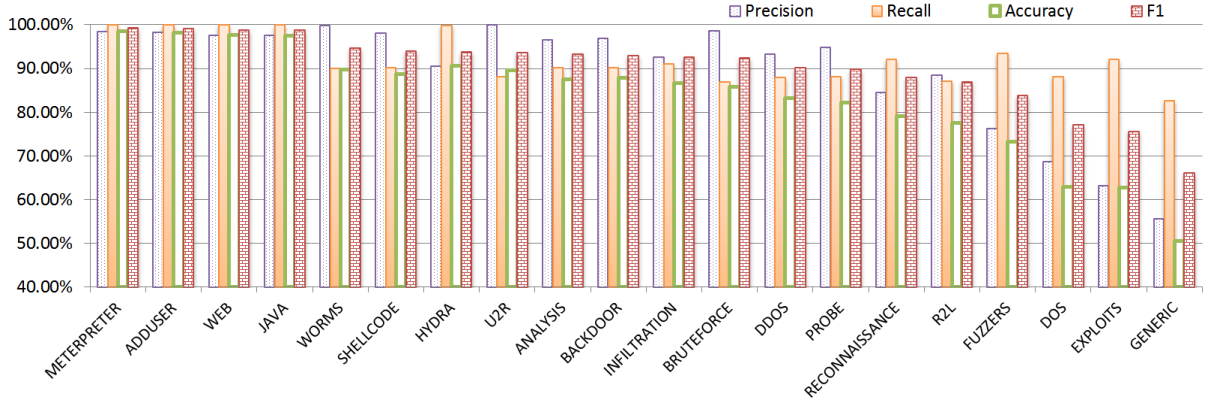


Figure 5.2: Median metric scores for all the datasets and algorithms, grouped by attacks.

call, Accuracy and F1 scores related to each of the attacks in the considered datasets. From left to right of Figure 5.2, it is possible to observe attacks with decreasing metric scores. On the right of the figure, we depict the attacks that turned out to be tricky to identify, and that generated the higher amount of false positives and false negatives. More in detail, it is possible to observe that anomalies generated by the *generic* and *exploits* attacks are difficult to detect. In fact, these attacks have heterogeneous characteristics: it is not trivial to define an expected behaviour, and consequently it is difficult to define what is not expected, i.e., anomalous. Another interesting observation regards *DoS* and *DDoS* attacks: Figure 5.2 shows that median scores for DDoS are higher than DoS, implying that the anomalies generated by DDoS attacks are detected with higher probability. When multiple malicious hosts are trying to flood the target system with network packets, the packets arrival rate raises significantly, and therefore it is noticeably different from the expected rate. On the other side, anomalies due to *Meterpreter*, *AddUser*, *Web* and *Java* attacks are detected with excellent Recall scores (100%), implying that no false negatives were generated. We can conclude that most of the *Application* attacks are on average easily detected by the algorithms considered in this study, while it is more difficult to detect generic and exploit attacks, probably because they introduce minimal - and not homogeneous - perturbations of the system and network behaviour.

#### RQ4: Is there a dataset that offers better detection scores than others?

The question is mostly related to the intrinsic difficulties of correctly defining an expected behaviour, that must be defined relying on the available data points of a given dataset. Since anomalies are supposed to be rare events, our conjecture is that in datasets with a low ratio of attacks it should be easier to define the expected behaviour and, consequently, to detect anomalies. Table 5.2 aggregates the results of all the algorithms for all the attacks contained in each dataset, partially confirming this conjecture. From the top of the table, we can see that the higher F1 scores are related to the AL and the IX datasets: ac-

Table 5.2: Metric scores (*median  $\pm$  std*) for the datasets considered in the study

<b>Dataset</b>	<b>AUC (ROC)</b>	<b>Recall</b>	<b>Precision</b>	<b>Accuracy</b>	<b>F1</b>
KDDCup99	$44.1 \pm 8.2$	$95.3 \pm 11.4$	$86.2 \pm 4.9$	$80.0 \pm 7.2$	$88.8 \pm 8.4$
NSLKDD	$48.7 \pm 8.6$	$87.6 \pm 3.2$	$88.2 \pm 2.5$	$75.3 \pm 2.6$	$84.6 \pm 2.2$
ISCX2012	$45.8 \pm 3.4$	$97.4 \pm 0.5$	$88.6 \pm 1.5$	$86.0 \pm 1.4$	$92.4 \pm 0.7$
ADFALD	$49.9 \pm 1.4$	$97.6 \pm 1.0$	$100.0 \pm 0.7$	$97.5 \pm 1.4$	$98.8 \pm 0.5$
UNSWNB15	$49.8 \pm 3.2$	$84.8 \pm 3.5$	$90.2 \pm 3.1$	$74.0 \pm 3.3$	$84.5 \pm 2.6$

cording to Table 3.1, these contain respectively 11.22% and 11.68% of attacks. However, this reasoning does not apply to the KC and NK datasets: KC has an higher ratio of attacks than NK, but it has better scores than NK. This may also be related to the specific attacks logged in the datasets: some of them may be easier to detect, because they generate point anomalies that significantly differ from the expectations. In addition, it is worth noticing that the AUC metric resulted in very similar and quite low scores (between 40% and 50%) across all the datasets, meaning that parameters of the algorithms should be tuned carefully before conducting anomaly detection on the datasets we selected for this study.

## 6

# Conclusion and Future Work

Generally, the vast majority of research works on anomaly detection, and on anomaly-based intrusion detection systems, propose a novel technique for intrusion detection and then compare it with a small set of different algorithms executing on a single dataset. Valuable examples are cited, especially [43], [34], [62]. This structure is very effective in presenting a novel algorithm and showing how it performs compared to a few existing ones. However, the experimental comparison of the target algorithm with algorithms from state of the art is often limited to proof-of-concept samples or a few target datasets. A question that is often left open is if an algorithm that is proven effective for a given case study or dataset has a similar behavior when applied to a different - although similar - context, or when it is evaluated using a different metric.

An extensive evaluation of algorithms by considering different categories of attacks, target systems (datasets), and scoring metrics would be beneficial. Being aware that the "silver bullet" algorithm, or rather an algorithm which always performs better than others, does not exist (yet?), we think that an in-depth comparison among *anomaly detection algorithms for intrusion detection* is needed to understand which (family of) algorithm is recommended when dealing with a specific class of attacks and systems.

The results suggest that many works may be built on this baseline, contributing to anomaly-based intrusion detection. In particular, our current and future works aim at understanding if some algorithms may cover large sets of attacks, while the uncovered attacks may be detected by using other algorithms in conjunction with the primary strategy. Then, we will investigate if specific algorithms or algorithms families are particularly effective in detecting either point, contextual, or collective anomalies. The analysis may be complemented by deriving which kind of anomaly our (category of) attacks are more likely to generate.

Therefore this work aims at providing a framework for a quantitative comparison of anomaly-based intrusion detectors for critical distributed systems, which may be used by researchers and practitioners when designing and assessing anomaly-based intrusion detection systems. The previous results of this work considering the quantitative comparison of the anomaly detection algorithms in intrusion detection systems has a publication in [75].

Future works would include improving the possible weakness of the methodology for comparing anomaly detection algorithms by considering sliding win-

dow algorithms and even supervised and semi-supervised ones, automating more parts of the framework, and establish possible guidelines to suggest algorithms to build intrusion detection systems.

---

# Bibliography

- [1] M. R. Lyu, ed., *Handbook of Software Reliability Engineering*. Hightstown, NJ, USA: McGraw-Hill, Inc., 1996.
- [2] T. Verwoerd and R. Hunt, "Intrusion detection techniques and approaches," *Computer communications*, vol. 25, no. 15, pp. 1356–1365, 2002.
- [3] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *ACM Sigmod Record*, vol. 29, pp. 427–438, ACM, 2000.
- [4] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 833–844, ACM, 2012.
- [5] Y. Himura, K. Fukuda, K. Cho, and H. Esaki, "An automatic and dynamic parameter tuning of a statistic-based anomaly detection algorithm," in *Proceedings of the 2009 IEEE International Conference on Communications, ICC'09*, (Piscataway, NJ, USA), pp. 1003–1008, IEEE Press, 2009.
- [6] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of pca for traffic anomaly detection," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, pp. 109–120, June 2007.
- [7] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International Conference, Co-NEXT '10*, (New York, NY, USA), pp. 8:1–8:12, ACM, 2010.
- [8] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PloS one*, vol. 11, no. 4, p. e0152173, 2016.
- [9] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "An experimental evaluation of novelty detection methods," *Neurocomputing*, vol. 135, pp. 313–327, 2014.
- [10] K. L. Ingham and H. Inoue, "Comparing anomaly detection techniques for http," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 42–62, Springer, 2007.

- 
- [11] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proceedings of the 2003 SIAM Int. Conference on Data Mining*, pp. 25–36, SIAM, 2003.
- [12] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.
- [13] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *In Proceedings of the Int. Conference on Machine Learning*, Citeseer, 2000.
- [14] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: system log analysis for anomaly detection," in *Software Reliability Engineering (ISSRE), 2016 IEEE 27th International Symposium on*, pp. 207–218, IEEE, 2016.
- [15] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.
- [16] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.
- [17] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001)*, pp. 5–8, 2001.
- [18] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, pp. 18–28, Feb. 2009.
- [19] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.
- [20] L. Cherkasova, K. Ozonat, N. Mi, J. Symons, and E. Smirni, "Anomaly? application change? or workload change? towards automated detection of application performance anomaly and change," in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pp. 452–461, June 2008.
- [21] O. Jung, S. Bessler, A. Ceccarelli, T. Zoppi, A. Vasenev, L. Montoya, T. Clarke, and K. Chappell, "Towards a collaborative framework to improve urban grid resilience," in *2016 IEEE International Energy Conference (ENERGYCON)*, pp. 1–6, April 2016.
- [22] J. Theiler and M. Cai, "Resampling approach for anomaly detection in multispectral images," *Proc SPIE*, vol. 5093, 04 2003.
- [23] N. Abe, B. Zadrozny, and J. Langford, "Outlier detection by active learning," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06*, (New York, NY, USA), pp. 504–509, ACM, 2006.

- 
- [24] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *SIGMOD Rec.*, vol. 29, pp. 427–438, May 2000.
  - [25] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," in *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, (Washington, DC, USA), pp. 430–433, IEEE Computer Society, 2004.
  - [26] Y. Liao and V. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," an earlier version of this paper is to appear in the proceedings of the 11th unix security symposium, san francisco, ca, august 2002," *Comput. Secur.*, vol. 21, pp. 439–448, Oct. 2002.
  - [27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, pp. 93–104, May 2000.
  - [28] J. Tang, Z. Chen, A. W.-C. Fu, and D. W.-L. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD '02, (London, UK, UK), pp. 535–548, Springer-Verlag, 2002.
  - [29] K.-H. G. P. B. Papadimitriou, S. and Faloutsos, "Loci: Fast outlier detection using the local correlation integral," Tech. Rep. IRP-TR-02-09, Intel Research Laboratory, 2002.
  - [30] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, (New York, NY, USA), pp. 59–68, ACM, 2004.
  - [31] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek, "A framework for clustering uncertain data," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1976–1979, 2015.
  - [32] M. Amer and M. Goldstein, "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer,"
  - [33] F. J. Anscombe, "Rejection of outliers," *Technometrics*, vol. 2, no. 2, pp. 123–146, 1960.
  - [34] M. Goldstein and A. Dengel, "Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm," 2012.
  - [35] R. Kwitt and U. Hofmann, "Unsupervised anomaly detection in network traffic by means of robust pca," in *Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on*, pp. 37–37, IEEE, 2007.
  - [36] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.



- 
- [37] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2Nd Edition)*. New York, NY, USA: Wiley-Interscience, 2000.
- [38] M. Amer, M. Goldstein, and S. Abdennadher, “Enhancing one-class support vector machines for unsupervised anomaly detection,” in *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, pp. 8–15, ACM, 2013.
- [39] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.
- [40] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *Data Mining, 2008. ICDM’08. Eighth IEEE International Conference on*, pp. 413–422, IEEE, 2008.
- [41] H.-P. Kriegel, M. S. Hubert, and A. Zimek, “Angle-based outlier detection in high-dimensional data,” in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’08*, (New York, NY, USA), pp. 444–452, ACM, 2008.
- [42] C. Piao, Z. Huang, L. Su, and S. Lu, “Research on outlier detection algorithm for evaluation of battery system safety,” vol. 6, pp. 830402–830402, 08 2015.
- [43] H.-P. Kriegel, A. Zimek, *et al.*, “Angle-based outlier detection in high-dimensional data,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 444–452, ACM, 2008.
- [44] V. Garcia-Font, C. Garrigues, and H. Rifà-Pous, “A comparative study of anomaly detection techniques for smart city wireless sensor networks,” *Sensors*, vol. 16, no. 6, p. 868, 2016.
- [45] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, “The 1999 darpa off-line intrusion detection evaluation,” *Computer networks*, vol. 34, no. 4, pp. 579–595, 2000.
- [46] J. T. Force and T. Initiative, “Security and privacy controls for federal information systems and organizations,” *NIST Special Publication*, vol. 800, no. 53, pp. 8–13, 2013.
- [47] T. Casey, “Threat agent library helps identify information security risks,” 2007.
- [48] “Open web application security project (owasp).” [https://www.owasp.org/index.php/Main\\_Page](https://www.owasp.org/index.php/Main_Page). Accessed: 2018-02-05.
- [49] N. Nostro, A. Bondavalli, and N. Silva, “Adding security concerns to safety critical certification,” in *Software Reliability Engineering Workshops (ISSREW), 2014 IEEE International Symposium on*, pp. 521–526, IEEE, 2014.

- 
- [50] J. Mirkovic and P. Reiher, "A taxonomy of ddos attack and ddos defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.
- [51] N. Gruschka and M. Jensen, "Attack surfaces: A taxonomy for attacks on cloud services," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pp. 276–279, IEEE, 2010.
- [52] T. F. Lunt, "A survey of intrusion detection techniques," *Computers & Security*, vol. 12, no. 4, pp. 405–418, 1993.
- [53] "Elki data mining." <https://elki-project.github.io/>. Accessed: 2018-04-30.
- [54] Y. Zhang, N. Meratnia, and P. Havinga, *A taxonomy framework for unsupervised outlier detection techniques for multi-type data sets*. No. Paper P-NS/TR-CTIT-07-79 in CTIT Technical Report Series, Netherlands: Centre for Telematics and Information Technology (CTIT), 11 2007.
- [55] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Comput. Netw.*, vol. 51, pp. 3448–3470, Aug. 2007.
- [56] "1999 darpa intrusion detection evaluation data set." <https://www.ll.mit.edu/ideval/data/1999data.html>. Accessed: 2018-04-30.
- [57] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proceedings of the 6th International Conference*, p. 8, ACM, 2010.
- [58] A. Shiravi, H. Shiravi, M. Tavallaei, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.
- [59] B. Marlin, *Missing data problems in machine learning*. PhD thesis, 2008.
- [60] S. Rosset and A. Inger, "Kdd-cup 99: knowledge discovery in a charitable organization's donor database,"
- [61] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of outlier detection: Measures, datasets, and an empirical study continued," in *Lernen, Wissen, Daten, Analysen 2016*, ceur workshop proceedings, 2016.
- [62] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using k-nearest neighbour graph," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 3, pp. 430–433, IEEE, 2004.
- [63] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pp. 1–6, IEEE, 2009.

- 
- [64] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *Wireless Communications and Networking Conference (WCNC), 2013 IEEE*, pp. 4487–4492, IEEE, 2013.
- [65] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), 2015*, pp. 1–6, IEEE, 2015.
- [66] "Ixia." <http://www.ixiacom.com/products/perfectstorm>. Accessed: 2017-02-05.
- [67] J. Fürnkranz and P. A. Flach, "Roc 'n' rule learning—towards a better understanding of covering algorithms," *Machine Learning*, vol. 58, no. 1, pp. 39–77, 2005.
- [68] R. Johnson and B. Foote, "Designing reusable classes," *Journal of Object-Oriented Programming*, vol. 1, p. 22textendash35, 06 1988.
- [69] M. E. Fayad, D. C. Schmidt, and R. E. Johnson, *Building Application Frameworks: Object-oriented Foundations of Framework Design*. New York, NY, USA: John Wiley & Sons, Inc., 1999.
- [70] X. Wan, W. Wang, J. Liu, and T. Tong, "Estimating the sample mean and standard deviation from the sample size, median, range and/or interquartile range," *BMC medical research methodology*, vol. 14, no. 1, p. 135, 2014.
- [71] "Mongodb for giant ideas." <https://www.mongodb.com/>. Accessed: 2018-04-30.
- [72] "Rapidminer - anomaly detection." <https://github.com/Markus-Go/rapidminer-anomalydetection>. Accessed: 2018-04-30.
- [73] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 535–548, Springer, 2002.
- [74] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *ACM sigmod record*, vol. 29, pp. 93–104, ACM, 2000.
- [75] F. Falcão, T. Zoppi, C. B. V. Silva, A. Santos, B. Fonseca, A. Ceccarelli, and A. Bondavalli, "Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19*, (New York, NY, USA), pp. 318–327, ACM, 2019.