UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN

Ingegneria Informatica

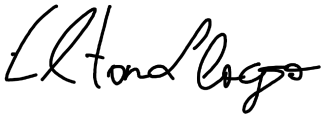
"TITOLO"

"A Comparison of Network Anomaly Detection Algorithms on Elasticsearch"

"Comparazione tra gli algoritmi di rivelazione delle anomalie di rete su Elasticsearch "

Relatore: Prof. / Dott: Damiano Varagnolo

Laureando/a: Elton Llacja



ANNO ACCADEMICO: 2023–2024

Data di laurea: 22/11/2024

Ai miei amati genitori: Elsa e Ben.

Ai miei cari fratelli: Jessica e Daniel.

"This is the highest wisdom that I own; freedom and life are earned by those alone who conquer them each day anew."

-Johann Wolfgang Von Goethe

Abstract

This thesis addresses the issue of anomaly detection in network security, with a particular focus on statistical-based models. The primary objective of this work was to develop an efficient anomaly detection system integrated with ElasticSearch to identify network anomalies in real-time. The research explores various anomaly detection techniques, including classification-based algorithms, statistical models, clustering models and proximity-based approaches, with a specific focus on the ARIMA model. The methodology involves the analysis of time-series data to forecast network behavior and detect deviations from expected patterns. The results demonstrate that statistical models, particularly ARIMA, offer a robust alternative to classification-based methods, providing reliable detection with fewer parameter dependencies. Finally, the thesis presents the implementation of the algorithm for the company, using a custom input data created in ElasticSearch to monitor network flow anomalies in real-time.

Italian Abstract

Questa tesi affronta il problema della rilevazione di anomalie nella sicurezza delle reti, con particolare attenzione ai modelli basati su metodi statistici. L'obiettivo principale di questo lavoro è stato sviluppare un sistema di rilevazione delle anomalie efficiente, integrato con ElasticSearch, per identificare le anomalie nelle reti in tempo reale. La ricerca esplora varie tecniche di rilevazione delle anomalie, inclusi algoritmi basati sulla classificazione, modelli statistici e approcci di clustering e di modelli di prossimità, con un focus particolare sul modello ARIMA. La metodologia impiegata prevede l'analisi dei dati di serie temporali per prevedere il comportamento della rete e rilevare deviazioni dai modelli attesi. I risultati dimostrano che i modelli statistici, in particolare ARIMA, offrono un'alternativa robusta ai metodi basati sulla classificazione, garantendo una rilevazione affidabile con minori dipendenze dai parametri. Infine, la tesi presenta l'implementazione dell'algoritmo per l'azienda, utilizzando un dato di input creato ad hoc in ElasticSearch per monitorare le anomalie nei flussi di rete in tempo reale.

Acknowledgements

I would like to express my deepest gratitude to my professor Damiano Varagnolo, whose invaluable guidance and insight have been fundamental to the success of this thesis. His foresight in understanding my aspirations and aligning this work with my passions has made this journey both meaningful and fulfilling. His support, encouragement, and expertise have not only contributed to the realization of this thesis but have also provided me with the tools to grow academically and personally.

For this, I am profoundly thankful.

I would also like to extend my thanks to the company Kirey Group, for offering me the opportunity to carry out this project within their organization. Their trust in my work and the resources they provided were essential for completing this thesis.

I would like to thank my friends who have been an important support throughout this journey. Their companionship, encouragement, and the moments we shared helped me to stay motivated and focused.

Lastly, I want to thank my family for their unwavering support throughout my academic journey. Their love, encouragement, and belief in me have been a constant source of motivation. I dedicate this thesis to them.

Contents

List of Figures

Acronyms

DoS Denial of Service

U2R User to Root

R2U Remote to User

R2L Remote to User

iForest Isolation Forest

iTree Isolation Tree

SVM Support Vector Machine

RAD Registry Anomaly Detection

DAG Directed Acyclic Graph

RNN Recurrent Neural Network

SOM Self-Organizing Maps

CLAD Cluster Label Anomaly Detection

ICD Inter-Cluster Distance

LOF Local Outlier Factor

ROC Receiver Operating Characteristic

AUC Area Under the Curve

KC KDD-cup

NK NSL-KDD

AL ADFA-LD

Chapter 1

Introduction

The rapid evaluation of information technology has deeply transformed the modern world, giving rise to an explosion of data collection and usage. While this proliferation offers extraordinary opportunities for insights and innovation, it also introduces critical challenges in safeguarding data integrity and ensuring security (2). Anomaly detection has become a key too in addressing these challenges, enabling the identification of unusual or abnormal patterns that may indicate significant events, such as system faults, cyber-attacks or fraudulent activities.

Anomaly detection techniques have been widely applied in various fields, form cybersecurity, where they help to identify network intrusions and denial of service attacks, to healthcare, finance and industrial applications. These methods aim to distinguish potentially harmful events from normal behavior, often leveraging advanced statistical models and machine learning techniques. For instance, anomalies in credit card transactions may indicate fraud, while deviations in medical data can signify health issues requiring urgent attention.

The importance of anomaly detection is particularly evident in network security, where administrators face increasingly sophisticated threats. Traditional signature-based detection systems have proven insufficient for identifying new and unknown attacks, as they rely on predefined patterns and require frequent updates. Anomaly detection algorithms address these limitations by modeling normal behavior and detecting deviations, making them especially effective for identifying novel intrusion attempts.

However, these techniques also face challenges, such as high false positive rates, where legitimate but unusual behaviors are flagged as anomalies.

In this context the temporal analysis of network data plays a critical role. By examining time-series metrics such as flow counts or packet rates, algorithms can identify significant deviations from expected patterns. Among the various approaches to anomaly detection, statistical methods like ARIMA (AutoRegressive Integrated Moving Average) have shown strong potential in modeling and predicting temporal trends. Arima offers a robust framework for understanding underlying patterns in time-series data and forecasting future behaviors, making it an important tool for identifying anomalies.

During my internship, I worked on a project to develop and implement an anomaly detection system for time-series data integrated with ElasticSearch, a platform widely used for real time data management. After a comprehensive review of potential approaches, ARIMA was selected as the algorithm of choice due to its statistical rigor and the company's specific operational needs. The decision was informed by its suitability for modelling temporal dynamics and its ability to handle seasonality and trends effectively.

1.1 Company Context

This thesis was conducted after a staging period at Kirey Group, a company specializing in IT consulting and advanced technological solution for digital transformation.
Till his foundation Kirey Group provides innovative solutions across various domains, including Cloud Computing, Cybersecurity, Infrastructure Management, Data Analytics, serving clients in sectors such as finance, telecommunications, and industry.
With a strong market presence and an expanding structure, Kirey Group positions itself as a strategic technology partner, supporting companies in their innovation and digitalization processes with a focus on quality and efficiency.

During the internship, I collaborated with the Monitoring and Consulting team, a business class specialized in monitoring and management of complex infrastructures using advanced technologies like ElasticSearch for real-time data analysis and observability. The primary goal of the internship project was to develop an advanced monitoring system, based on the real-time creation of a baseline capable of analyzing network data, identifying anomalies, and forecasting daily traffic trends. Unlike traditional monitoring systems, which are based on preconfigured models, this approach aims to build a dynamic baseline rely on real network data to detect anomalous events and improve IT resource

management.

The project focused on two main parts: the first algorithm aimed to construct the baseline to support input data and detect traffic anomalies, while the second was dedicated to forecasting network traffic over the next fifteen days. This system, oriented throw predictive analysis, wants to anticipate potential performance issues and optimize network resources, thus reducing the need for reactive interventions and enhancing the effectiveness of the monitoring services offered by the company.

The first phase of my project involved an extensive study of various anomaly detection algorithms, their categorization, and an analysis of their features to identify the most suitable approach for integration in ElasticSearch. This phase is essential to understand how different algorithms could contribute to achieving the core targets of the project: building a dynamic baseline for real-time anomaly detection and future forecasting trends.

(*)During the writing process of this thesis, I utilized ChatGPT as a support tool to review the text and address grammatical errors. ChatGPT provided suggestions for improving the clarity and coherence of the content while preserving the intended meaning. Its role was limited to linguistic and stylistic refinement, ensuring that the final document met high standards of quality in terms of language and presentation. All the scientific content, methodologies, and conclusions remain the result of my independent research and critical analysis.

1.1.1 Thesis Structure

The structure of this thesis is organized to address the main aspects of anomaly detection in network systems and to present the project developed during the internship. **Chapter 2** provides a comprehensive overview of anomaly detection, starting with its definition, followed by a detailed discussion of detection methods and an analysis of the primary types of network attacks that these techniques aim to identify. In **Chapter 3**, the focus shifts to the anomaly detection models, which are classified into five main categories: Classification-based, Statistical-based, Clustering-based, Distance-based, and Density-based methods. **Chapter 4** introduces the evaluation metrics used to compare the performance of anomaly detection algorithms and presents a comparative analysis based on the results obtained using these metrics.

Chapter 5 delves into the ARIMA statistical method, chosen by the company where the internship was conducted as the preferred model for the project. This chapter discusses the rationale behind this choice, highlighting the advantages of ARIMA over other models reviewed in the study. Finally, section 5.2 concludes the thesis by summarizing the work carried out, presenting the output of the ARIMA algorithm implemented for the internship project.

Chapter 2

Anomaly Detection

2.1 Anomaly Definition

Although an anomaly is defined by researchers in various ways based on its application domain, one widely accepted definition in that of Hawkins (17): 'An anomaly is an observation which deviates so much from other observations as to arouse that it was generated by a different mechanism'.

Anomalies are referred to as patterns in data that do not conform to a well-defined feature of normal patterns. They are produced by a variety of abnormal activities, for example frauds, cyber attacks, etc., which are significant for data analysts. According to (9), an important aspect of anomaly detection is to categorize them in various groups:

- Contextual anomaly: When a data instance behaves anomalously in a particular condition. It can be illustrated with temperature data by considering a scenario where a specific location typically experiences seasonal variations.

- Point anomaly: When a particular data instance deviates from the normal flow of the dataset. A realistic example of point anomaly occurs in monitoring bank transactions. if a customer who typically makes small purchases suddenly makes a single transaction of 5.000 euros this a significantly deviation from his usual behavior.

- Collective anomaly: When group of similar data instances behave anomalously in relation to the entire dataset. For example, in electrocardiograms (ECGs), a sustained period of low values, such as a prolonged decrease in heart rate, is considered abnormal and indicate a potential issue, rather than a single isolated low value.

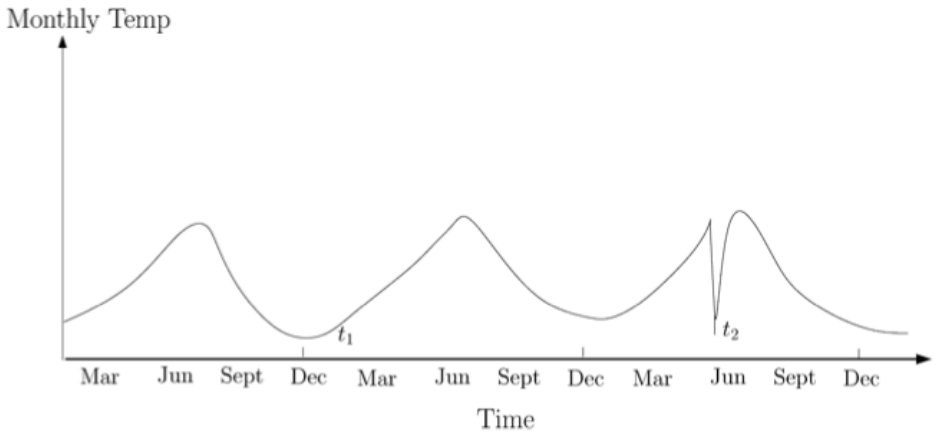


Figure 2.1: Chandola's example of context anomaly

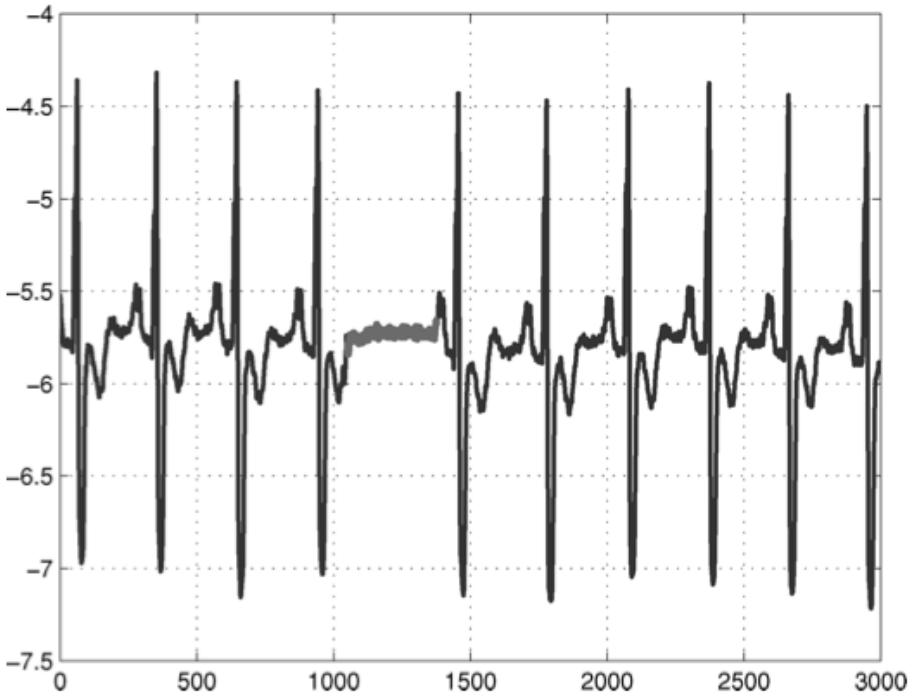


Figure 2.2: Collective anomaly example ECGs

2.2 Anomaly detection methods

Anomaly detection can be performed through various techniques, with the selection of the optimal method primarily carried by the input data's nature. Typically, anomaly detection methods operate on a dataset composed of instances that are defined by a set of attributes, also referred to as features, dimensions, fields or variables. When data consist of a single attribute, they are classified as univariate; however, when multiple attributes are present, the data are considerate multivariate. Although numerous anomaly detection techniques rely on independent data, wherein there are not assumed intrinsic relationship among instances, anomaly detection methods way also be effectively applied to datasets where instances are interdependent, such as data incorporating temporal components.

The nature of model's training data makes the first distinction among anomaly detection methods identified by Supervised, Semi-supervised and Unsupervised methods (16):

1. Supervised anomaly detection methods are applied when data samples within the dataset have been pre-labeled as "normal" or "anomalous" by subject-manner experts. This approach typically frames anomaly detection as a classification problem, where a predictive model is developed using labeled training data to classify each new instance as either normal or anomalous. The process of labeling is often time-intensive and resource-demanding, particularly because it frequently requires manual effort.

   Anomaly labeling generally demands a higher level of complexity due to the dynamic and variable nature of anomalous instances.

   Supervised model faces two main challenges: Class imbalance and Recall Priority.

   Due to the typically low proportion of anomalous instances compared to normal ones, class imbalance techniques, like oversampling anomalies, are applied to improve classifier efficiency. However, limited anomalous samples can still delay performance.

   Additionally, in many applications, prioritizing Recall is essential. Recall measures the model's effectiveness in identifying anomalies, calculated as the ration of true positives to the total instances in the target class.

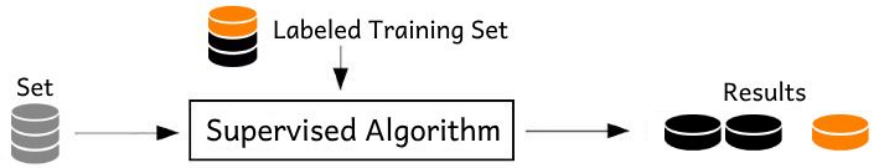


Figure 2.3: Supervised Algorithm

2. Semi-supervised methods require that only a portion of the learning dataset is labeled, typically including only normal instances, although some approaches may use anomalies as labeled data. Unlike supervised methods, where the entire dataset is labeled, semi-supervised models rely on a mixed dataset where only a low portion is labeled, while the majority remain undone.

   The goal of these models is to leverage the small labeled subset to identify similar patterns among unlabeled data. These changes were done to avoid the amount of cost in terms of time and resources, often requiring the involvement of experts or analysts to ensure label quality. Consequently, semi-supervised anomaly detection is widely utilized.

3. Unsupervised anomaly detection methods do not require labeled data at all and assume that the number of anomalies is significantly lower than the number of normal data. In scenarios where data labeled normal or outlier are not available, unsupervised learning models are utilized, which implicitly assume that normal data are “clusterable”, meaning that they follow a more frequent pattern compared to anomalies. The main advantage of this approach is the elimination of the need of labeling, so all the costs could be reduced. However, if the initial assumption is not met, the model may suffer from a high number of false negatives and false positives.

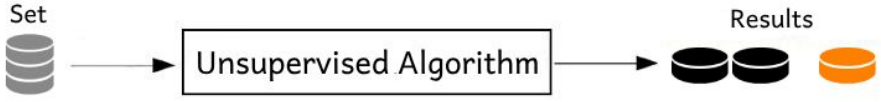


Figure 2.4: Unsupervised Algorithm

All the models discussed in Chapter 3 follow an unsupervised anomaly detection approach.

2.2.1 Output of anomaly detection techniques

One important matter in anomaly detection is how anomalies are represented in output. I consider the two following ways (9):

- Scores: Scoring-based anomaly detection techniques establish an anomaly score to each data instance, basically from 0 to 1and their decimals. Then the scores are ranked and the analyst decides the anomaly or uses a threshold to select them.
- Binary/label: basing to these techniques, outputs are considered in a binary manner, either anomalous or normal.

The binary label approach is considered more efficient in term of computationally because each data does not need to be assigned an anomaly score.

2.3 Types of network attacks

According to (2), the task of network security is to protect digital information by maintaining data confidentially and integrity, and ensuring the availability of resources. In simple terms, a threat/attack refers to anything which has harmful characteristics aimed to compromising a network (4). The poor design of a network, carelessness of its users and/or miss-configuration of its software or hardware can be vulnerable to attacks (22):

1. Denial of Service: (DoS) is a type of exploitation that misuses network or host resources, aiming to disrupt the normal operating environment and make a service unavailable. A common example of DoS attack is flooding a server with excessive connection requests, which prevents legitimate users from accessing a web service. Since carrying out a DoS attack doesn't require prior access to target system, it is considered particularly threatening.

2. Probe: This type of attack is used to collect information about a target network or host, primarily for reconnaissance. Reconnaissance attacks are a common method for identifying the types and numbers of devices connected to a network, as well as determining the software and applications running on a host. A probe attack is often the first step in compromising the target. Although these attacks do not directly cause effective damage, they are considered serious threats to organization as they may reveal valuable information that could be used to launch a harder attack.

3. User to Root: (U2R) is a type of attack that occurs when an intruder seeks unauthorized access to an administrative account to manipulate or misuse critical resources. By employing techniques such as social engineering or password sniffing, the attacker may initially gain access to a regular user account and then exploit one or more vulnerabilities to elevate privileges to those of a superuser.

4. Remote to User: (R2U or R2L) is a type of attack initiated when an intruder aims to obtain local user access on a targeted machine to gain the ability to send packets over its network. Commonly, the attacker uses automated scripts for password guessing through methods such as brute force or some sophisticated sniffing tools, enabling the aggressor to penetrate the system easily.

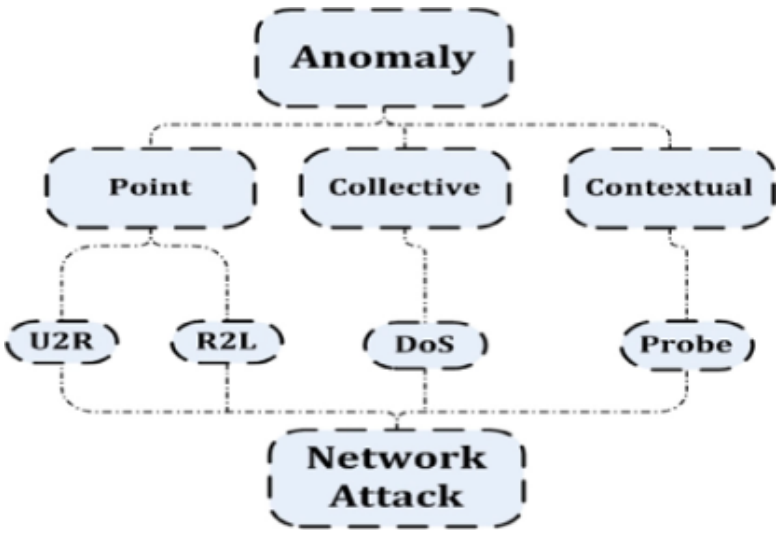


Figure 2.5: Different network attacks types

Chapter 3

Anomaly detection models

Anomaly detection is a practical technique used to identify behaviors or events that deviate significantly from normal data patterns. Numerous approaches and models have been developed to handle this problem effectively, each with its own strategies and applications in different contexts. Based on the literature review, I identified five main categories of models for anomaly detection, each grounded different theoretical and practical paradigms:

1. Classification-based models:These models focus on classifying data as either "normal" or "anomalous." Some of the most common models in this category include algorithms such as Isolation Forest (iForest), Support Vector Machines (SVM), Bayesian Networks, and Neural Networks.

2. Statistical-based models: Statistical models rely on probabilistic distributions to detect anomalies. These models assume that data follow a predefined statistical distribution and identify as anomalous those observations that significantly deviate from the distribution. This approach is useful when the data can be well modeled by known distributions, and it allows anomalies to be detected based on statistical measures such as mean, variance, or other statistics.

3. Clustering-based models: Clustering approaches are based on the idea that normal data tends to form natural clusters, while anomalies do not belong to any cluster. These models include both unsupervised and more advanced techniques. Clustering models are particularly useful when data are incompletely labeled or not labeled at all.

4. Proximity-based models: These models detect anomalies based on the distance or density of the data patterns. It can further divided into two subcategories:

    - Distance-based models: These models use the distance between observations to determine when a data point is far from or close to others, classifying points that are too distant from most others as anomalous.
    - Density-based models: These focus on the density of the data, for instance, the concentration of points in a given region. One example of this approach is the Local Outlier Factor (LOF), which measures how a point is, in terms of density, compared to its neighbors

These models are powerful tools for addressing the anomaly detection problem across various application domains, from computer systems to healthcare, and fraud detection. In the this chapter, each of these categories will be explored in detail, evaluating the advantages and limitations of each approach.

3.1 Classification-based models

3.1.1 Isolation Forest

The isolation forest (iForest) is an innovative classification-based algorithm and unsupervised anomaly detection method that differs fundamentally from traditional approaches. Developed specifically for anomaly detection task, iForest introduces the concept of isolation as a more effective and efficient means of identifying anomalies, as opposed to rely on distance or density measures. The algorithm was first introduced by (27) with the premise that anomalies, being sparse and distinct from normal data, can be isolated more easily than regular data points. This approach eliminates the need for predefined measures, making it particularly suited in term of complexity and high-dimensional datasets. As its core, iForest uses a tree-based structure, referred to as an isolation tree (iTree), which is built by recursively partitioning the dataset. In each partitioning iteration the algorithm randomly selects an attribute and split value within the range of that attribute, thereby dividing the data. This recursive process continues until each data point is isolated.

The fundamental idea is that “they will generally require fewer partitions to be isolated compared to normal data points”, because anomalies are “few and different”. Thus, anomalies tend to generate

shorter paths within the iTree, making the easier to identify. In practical terms, the length of the isolation path serves as ley indicator within the algorithm.

Anomalous data points, being more easily separable, will have a lower anomaly score, which is calculated based on the number of partitions required to isolate the point. A lower anomaly score indicates a higher probability that the data point is an anomaly. For example, an anomalous point x_o will typically require fewer partitions to be isolated, resulting in a shorter path, while a normal point x_i will require more partition, producing a longer path. This concept is illustrated in a several examples, figure 3.1 and figure 3.2, which highlight the differences in isolation paths between normal and anomalous data points.

One of the key strengths of the iForset algorithm is its computational efficiency. Unlike traditional methods based on distance or density, which require extensive computational resources and are often unsuitable for large datasets, iForest reduces the computational load by using a binary tree structure. This tree structure allows for linear time complexity and minimal memory requirements, making the algorithm highly scalable, even when working with large databases.

To build a forest of iTrees, the dataset is recursively partitioned into subsets until either a depth limit is reached or all instances are isolated in leaf nodes. The resulting tree structure is binary, with each node having zero or two children, forming a hierarchical partitioning system that allows for rapid isolation of anomalies. As proofed by (27), the high robustness, reducing false positives compared to non-specific anomaly detection methods, which often produce numerous false alarms and are less suitable for high-dimensional data.

Finally, the ranking of anomalies within a dataset is determined by sorting the data points according to their anomaly scores. This allows the algorithm to efficiently identify the most anomalous points, improving the overall detection process. As described by (7) and illustrated through practical examples, the iForest method offers one of the most effective and versatile solutions for anomaly detection in complex environments with vast datasets. Its ability to adapt to different data structures and its simplicity in both implementation and interpretation of results make it a valuable tool in the field of anomaly detection.

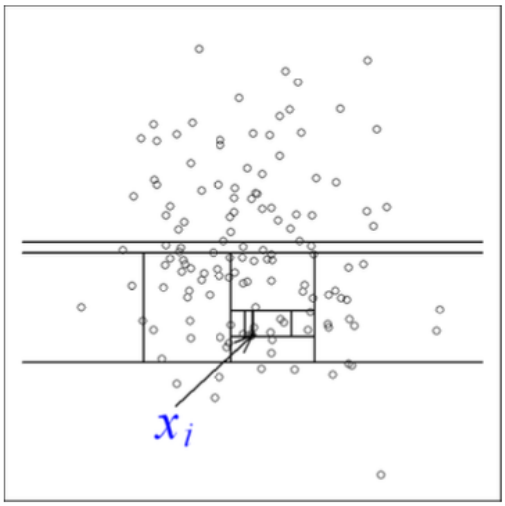


Figure 3.1: Isolating a normal point x_i

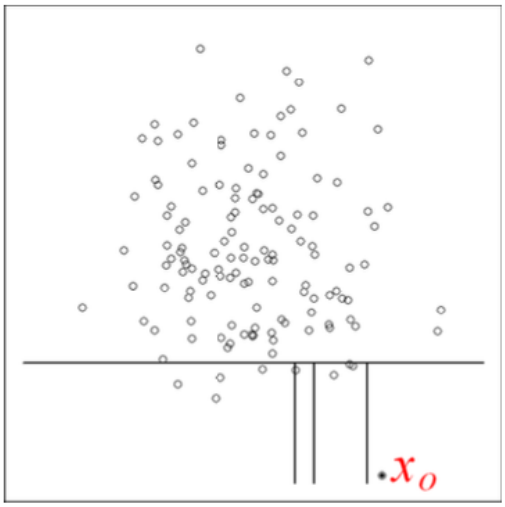


Figure 3.2: Isolating an anomalous point x_o

3.1.2 Support Vector Machine

The Support Vector Machine (SVM) is a classification algorithm designed to derive a hyperplane that maximizes the separating margin between positive and negative classes (13). This method is grounded in the principle of structural risk minimization from statistical learning theory, enabling it to provide robust classification capabilities. The standard SVM operates as a supervised learning technique, requiring labeled data to create a classification rule. However, SVM can also be adapted for unsupervised learning, in which it attempts to separate the entire training dataset from its origin. In this unsupervised variant, rather than focusing on separating two distinct classes, the SVM seeks to isolate data instances from the origin with maximum margin, solving an optimization problem to find the best hyperplane ((13) and (12)), as shown in figure 3.3. The Sequential Minimal Optimization algorithm, a well-known optimization method, is often used to solve this problem (31).

Further applications of SVM for anomaly detection involve techniques inspired by the One-Class SVM (OCSVM) model. For instance, (19) proposed the Registry Anomaly Detection (RAD) method, which applies a modified version of OCSVM in a supervised context to monitor Windows registry queries. Registry access is predictable during normal computer operations, as specific registry keys are commonly accessed by Windows programs. Deviations from typical registry activities, where certain programs and registry keys are used frequently, are considered anomalous. The OCSVM employed in the RAD system maps input data into a high-dimensional feature space using a kernel function and iteratively identifies the optimal hyperplane to distinguish between normal and abnormal registry activities.

Another adaptation of the SVM for anomaly detection was introduced by (20) through the Robust SVM (RSVM), which was specifically designed to manage noisy training data. In practice, training datasets frequently contain noise, which disrupts the primary assumption that all training samples are independent and identically distributed. This deviation often leads to a non-linear decision boundary in standard SVMs, reducing their ability to generalize across new data points. RSVM addresses this limitation by incorporating a class center averaging technique to create a smoother decision boundary, which automatically controls regularization. Consequently, RSVM not only enhances generalization by generating smoother decision surfaces but also reduces the number of support vectors, leading to faster computational performance.

More recently, advancements in SVM for anomaly detection have been demonstrated in a patented method for confident anomaly detection within computer network traffic (6). This approach applies the principles of SVM to monitor and detect anomalies in network traffic, providing a robust framework for managing real-time network security.

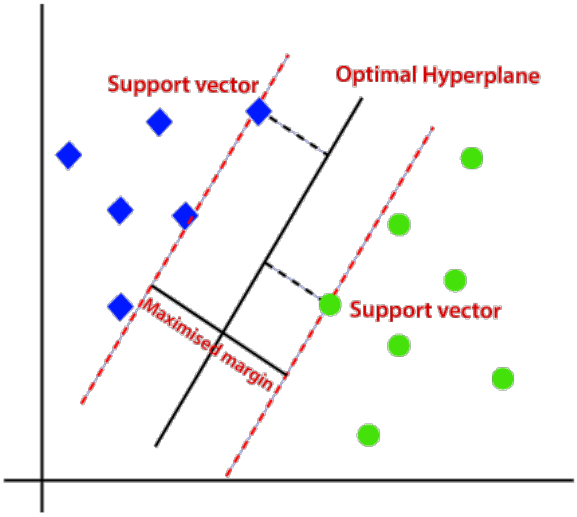


Figure 3.3: SVM example from www.javatpoint.com/machine-learning-support-vector-machine-algorithm

3.1.3 Bayesian Network

According to (3), A Bayesian network is a robust approach for modeling domains with inherent uncertainty, represented as a directed acyclic graph (DAG). In this structure, each node corresponds to a discrete random variable and holds a conditional probability table (CPT), which calculates the likelihood of the node being in a particular state. Within the network, a parent–child relationship is established, signifying that the state of a child node depends on its parent nodes. This setup allows Bayesian networks to classify events effectively, making them suitable for anomaly detection in network security (24).

Bayesian networks offer a structured means of addressing two common issues in anomaly detection systems that can lead to high false positive rates. Anomaly detection systems frequently employ multiple models to analyze different aspects of events. However, when aggregating results from these models to assess normality or abnormality, false positives can increase substantially. Furthermore, anomaly detection systems often struggle to distinguish unusual but legitimate behaviors, such as spikes in CPU or memory usage, from genuine anomalies, which can result in important contextual information being overlooked.

(24) proposed a Bayesian network-based method to address these challenges. Given an ordered stream of input events (S = e_1, e_2,...), the event classification system uses Bayesian networks to determine if an event is normal or abnormal by analyzing outputs from k models $(M = m_1, m_2, ..., m_k)$ along with possible additional information (I). Each model examines features of a given event and compares them with previously observed patterns. The decision rule in the event classification system (EC) is defined as:

$$EC(o_1, o_2, \ldots, o_k, I) = \begin{cases} \text{normal} & \text{if } \sum_{i=1}^{k} o_i \leq I \\ \text{anomalous} & \text{if } \sum_{i=1}^{k} o_i > I \end{cases} \tag{3.1}$$

3.1.4 Neural Network

Neural networks, recognized for their strong classification capabilities, have also shown potential in the field of network anomaly detection (3). Their extensive use across domains, such as image and speech processing, highlights their adaptability, even though with high computational demands. In network anomaly detection, neural networks are often combined with other techniques, including statistical models and their variations, to enhance detection accuracy. (18) introduced the concept of a Recurrent Neural Network (RNN) as a tool for identifying outlying or anomalous network traffic patterns. The RNN used in this study is a feed-forward, multi-layer perception with three hidden layers positioned between the input and output layers. Its primary goal is to replicate the input data pattern at the output layer with minimal error through training.

The architecture of an RNN is mathematically described by the function $S_k(I_{ki})$, which generates the output from unit i in layer k:

$$\theta = I_{ki} = \sum_{j=0}^{L_{k-1}} w_{kij} \cdot Z_{(k-1)\cdot j} \tag{3.2}$$

where I_{ki} represents the weighted sum of inputs to the unit, Z_{kj} is the output from the j-th unit of the k-th layer, and L_k is the number of units in the k-th layer.

The outlier factor (OF) is defined using the trained RNN as follows:

$$OF_i = \frac{1}{n} \sum_{j=1}^{n} (x_{ij} - o_{ij})^2 \tag{3.3}$$

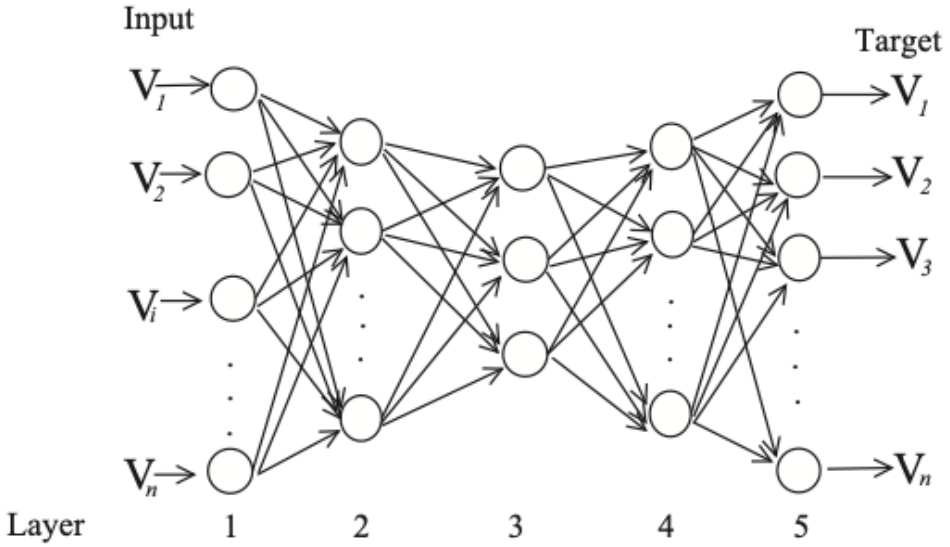


Figure 3.4: Schematic of replicator neural network, adapted from (18)

3.2 Statistical-based models

Statistical methods for anomaly detection, also known as model-based approaches, are rooted in principle that normal data clusters around high-probability regions of a statistical model, while anomalous data point, those that deviate significantly from expected patterns, reside in low-probability areas. This fundamental assumption provides the basis for identifying outliers, as these points contrast sharply with the distribution of normal data, often indicating unusual or potentially problematic behaviors.

Statistical anomaly detection typically comprises two main phases:

- Training phase: A statistical model of normal behavior is constructed based on a representative dataset. This establishes a baseline profile that captures expected data patterns.
- Testing phase: New data points are evaluated against this baseline. If a point deviates significantly from the normal profile, it is flagged as anomalous.

Statistical methods can be categorized into parametric and non-parametric approaches, each differing in their assumptions about data distributions and their treatment of statistical parameters:

- Parametric methods assume that data follows a specific probability distribution, often Gaussian and aim to estimate this distribution's parameters from the training data. For example ((5)), a Gaussian model calculates the mean and standard deviation of the dataset. Datapoints lying beyond a certain threshold, typically expressed as a multiple of the standard deviation, are classified as anomalies. This approach is widely used when the data distribution is known or can reasonably be approximated by common distributions.

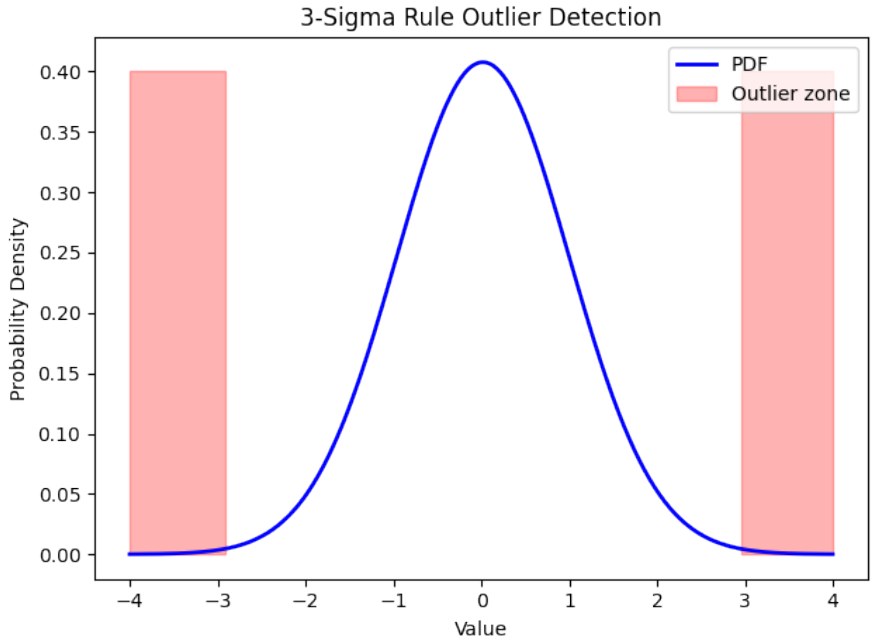


Figure 3.5: Outliers in Gaussian distribution

- Non-parametric methods do not assume any specific distribution for the data. These models directly analyze the data distribution, typically by examining the frequency of observations. One common approach is the histogram-based model, where the data is partitioned into bins, each representing an interval of values. The frequency of points within each bin is then analyzed, with low-frequency bins signaling potential anomalies.

Two notable applications of statistical methods in anomaly detection illustrate the effectiveness of these applications in real word setting:

1. (39): This model uses chi-square statistics to detect anomalies within information systems. The model first builds a profile of normal behavior and then evaluates deviations from this profile using a chi-square distance measure:

$$X^2 = \sum_{i=1}^{n} \frac{(X_i - E_i)^2}{E_i} \tag{3.4}$$

   where X_i is the observed value, E_i is the expected value and n represents the number of monitored variables. An anomaly is flagged when the chi-square distance surpasses a predefined threshold.

2. (25): Focusing on the detection of rare network attacks, this model assigns an anomaly score to each network request by analyzing three key parameters: request type, length, and payload distribution. The anomaly score is calculated as a weighted sum of these parameters:

$$AS = 0.3 \cdot AS_{type} + 0.3 \cdot AS_{length} + 0.4 \cdot AS_{payload} \tag{3.5}$$

3.3 Clustering in Anomaly Detection

Clustering is a widely used technique for uncovering patterns in high-dimensional, unlabeled data, making it especially useful for detecting anomalies and intrusions (37). Unlike traditional methods, that rely in predefined intrusion signatures, clustering algorithms can identify irregularities directly form the data, allowing for adaptive and signature free detection. This capacity to recognize anomalies based on inherent data structures highlights the versatility of clustering in security and anomaly detection research.

There are two primary approaches to clustering-based anomaly detection (3):

1. Unsupervised Clustering: The anomaly detection model is trained on unlabeled data that contains both normal and anomalous instances. Since anomalous data is expected to make up only a small fraction of the total dataset, clusters that are large and dense are typically assumed to represent normal data, while smaller, more dispersed clusters are considered indicative of potential attacks.

2. Semi-supervised: The model is trained exclusively on normal data to establish a baseline profile of standard behavior. Anomalous data is then detected based on deviations from this profile. Instances that do not fit within the defined normal clusters are flagged as potential outliers or intrusions. This method is particularly effective in highly controlled environments where normal behavior is well understood.

Clustering-based anomaly detection generally rests on three main assumptions, as described by (9):

1. Cluster Inclusion: Normal data aligns well within clusters, while anomalies fall outside established clusters. Algorithms using this assumption rely on standard clustering algorithms to identify points that do not fit well within any cluster, marking these as anomalous.

2. Cluster Centroid Proximity: In datasets containing both normal and anomalous instances, normal data is often closer to the cluster centroid, while anomalies are located farther from the center. Algorithms based on this assumption use a distance score to measure how far each instance is from the nearest centroid, with more distant points likely to be anomalies.

3. Cluster size and density: Normal data forms larger, denser clusters, while anomalies appear in smaller or less dense clusters. By setting a threshold for cluster size or density, instances within clusters that fall below this threshold are labeled as anomalies. This approach is highly effective in detecting anomalies across datasets with varying densities.

To support anomaly detection, clustering methods are implemented through various algorithms. Some popular choices include k-means, improved k-means, k-medoids and Expectation-Maximization (EM) clustering.

3.3.1 Unsupervised Clustering Techniques

- K-means Clustering: (29) applied k-means clustering to distinguish normal clusters from anomalous ones by measuring the proximity of each instance to cluster centroids. Instances close to a normal cluster centroid are considered normal, while those farther away are flagged as anomalous if they exceed a distance threshold. Anomalous clusters with fewer instances are labeled as potential attacks.

- Fixed-Width Clustering: (13) and (32) proposed clustering to anomalies in network traffic. This algorithm initiates an empty set of clusters, adding new clusters dynamically by measuring the distance between each instance and the nearest cluster centroid. If the distance is less than a specified width parameter W, the instance is added to the cluster; otherwise, a new cluster is created. Clusters with a high instance count are labeled as normal, while smaller clusters are anomalous.

- Cluster Labeling Based on Proportion: In this model, a parameter X determines the proportion of clusters designated as normal. The clusters with the largest number of instances are labeled as normal, and the rest are considered potentially anomalous. This method assumes that normal data is most common in the dataset, making smaller clusters more likely to represent attacks or unusual events.

3.3.2 Advanced Clustering Techniques

- Cluster Label Anomaly detection (CLAD): (3) proposed the CLAB approach, which assess clusters based on both density and inter-cluster distance (ICD). By defining density and IDC thresholds, CLAD is able to classify clusters as either normal or intrusive. For clusters with density and a close ICD to neighboring clusters, the data points are considered normal; otherwise, are flagged as potentially intrusive.

- Real-time clustering for Dynamyc data: (30) proposed a clustering-based technique tailored for evolving data environments. This approach uses feature weighting to improve the model's adaptability to changing data distributions, making it effective for real-time detection in network security contexts.

- X-means Clustering for collective anomalies: (1) implemented x-means clustering to identify group-based anomalies, such as Distributed Denial of Service (DDoS) attacks. This method improves on traditional k-means by automatically determining the optimal number of clusters. By capturing collective anomalies more effectively than traditional techniques, x-means is especially useful for high-dimensional datasets.

Clustering-based anomaly detection techniques provide a robust approach to identifying suspicious patterns in data. Unsupervised and semi-supervised clustering allow for flexible application across environments where labeled data may be sparse or unavailable. Traditional methods, such as k-means and fixed-width clustering, offer foundational approaches to anomaly detection, while advanced techniques like CLAD and x-means clustering provide the adaptability and precision required for complex datasets and real-time environments. Although computationally demanding in high-dimensional data settings, these clustering methods are essential tools in the ongoing effort to detect and respond to anomalous activities effectively.

3.4 Proximity-based methods

Proximity-based methods are based on the assumption that anomalies are located at a certain distance from their nearest neighbors. The effectiveness of these approaches largely depends on the distance metric used. Proximity-based anomaly detection techniques can be further divided into two main categories: distance-based models and density-based models.

3.4.1 Distance model

A distance-based model assesses an observation on its distance from neighboring points. If this distance exceeds a predefined threshold, the observation is classified as an anomaly. For instance, (23) define a distance-based anomaly (DB) as follows: *An object O in a dataset T is considered DB(p,D) outlier if at least fraction p of the objects in T lies greater than distance D from O.*
Distance-based algorithms apply this definition by evaluating a region with radius D around the instance O: if the number of points within this region does not meet a specific threshold, O is flagged as an anomaly.
These algorithms can be efficiently applied to large datasets, and they can be further classified into various categories each with different computational complexities:

- Index-based algorithms: For each instance O, these count the number of points within a region of radius D to determine if O is an anomaly. if at least $M + 1$ neighboring points are found (where $M = N(1 - p)$ represent the maximum allowable number of objects inside an outlier's region), O is considered non-anomalous. This type of algorithm has a worst-case complexity of $O(kN^2)$, making it one of the most efficient approaches for distance-based models

- Nested-loop algorithm: These achieve the same performance as index-based but avoid the cost of index construction by using an efficient block design.

- Cell-based algorithm: These methods partition the space into cells and analyze proximity within each cell. They exhibit linear complexity with respect to the total number of instances N, but exponential complexity concerning the dimension k, making them optimal for smaller dataset.

Distance-based methods thus offer a robust and adaptable framework for identifying anomalies, enabling the isolation of instances that deviate significantly from expected behavior within the analyzed dataset.

3.4.2 Density model

Density-based anomaly detection methods operate under the assumption that anomalies within a dataset are less common and thus are located in low-density regions (26). In this approach, an instance is considered anomalous if it is situated in an area where the local density is significantly lower than that of its neighbors. The density of the region surrounding a point is calculated based on two parameters: the distance defining the radius of the area considered and the number of points within that area. However, traditional density-based methods may be ineffective when dealing with datasets that contain regions with varying densities. For example, if a dataset has two clusters, C_1 and C_2, with different densities, a point in a low-density region such as C_1 might be misclassified as normal rather than anomalous simply because it reflects the local density. In such scenarios, density-based anomaly detection methods must account for local density differences to achieve accurate results.
To address this issue, (8) introduced the Local Outlier Factor LOF, which allows for differentiation between global and local anomalies. LOF assigns each point a score that measures the degree of "outliernees" relative to the density of its neighborhood. This approach enables the detection of not only global anomalies but also local ones, which are points that appear anomalous only in relation to neighboring data rather than the dataset as a whole. The concept of neighborhood is based on selecting the k nearest neighbors, which must be appropriately chosen to obtain an accurate estimate of the local density.
The calculation of LOF involves several steps:

1. For each point O, calculate the k-distance, i.e., the distance between O and its k-th nearest neighbor, thereby defining O's neighborhood within a sphere of radius equal to the k-distance.

2. Determine the "reachability distance" between O an another point P within the neighborhood, which is the maximum of P's k-distance and the actual distance $d(O, P)$. This step prevents low-density points from disproportionately influencing the LOF calculation.

3. Calculate the local reachability distance of O as the inverse of the average reachability distance relative to the k-nearest neighbors.

4. Finally, the LOF of O is obtained as the average of the ratios between O's local reachability

density of its neighbors. Points with a LOF value significantly greater than 1 are classified as anomalies, as their density is lower relative to the surrounding neighborhood.

The LOF approach is particularly advantageous in datasets with clusters of varying density, as it considers each point's local context. In a two-dimensional example, one might observe a low-density cluster C_1 and a higher-density cluster C_2. A simple distance-based algorithm might not detect a point p_1 as anomalous within C_1 due to its compatibility with the region's low density. However, with the LOF approach, points like p_1 and p_2 could be accurately identified as anomalies if their relative density is lower than that of their local neighbors.

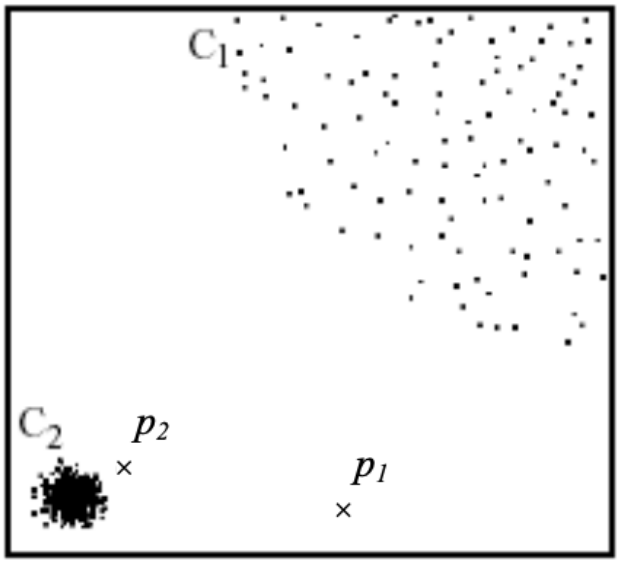


Figure 3.6: LOF example

Chapter 4

Selection of metrics

In anomaly detection, selecting and evaluating performance metrics is essential to verify the effectiveness of algorithms ant to compare different approaches (33). Typically, anomaly detection algorithms are evaluated using metrics based on binary labels that distinguish between anomalies (positive) and normal data (negative). According to (14) several key metrics are frequently adopted in the literature, including precision, recall, F1-score, accuracy, and ROC curve (AUC).

4.1 Metrics

- Precision: It measures the proportion of true positive TP among the sum of true positives TP and false positives FP. This metric evaluates the model's ability to minimize false positives, for example, instances that are incorrectly classified as positive. Precision is especially valuable when reducing the risk of misclassifying negative example is essential, as in applications where false alarm are costly.

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

- Recall: Also known as sensitivity, measures the model's ability to correctly identify positive data instances. It is calculated as the ratio of true positives TP to the sum of true positives TP and false negatives FN. This metric is critical in situations where minimizing FN is important, such as medical diagnostics where detecting all instances of a condition is essential.

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

- F1-score: It combines precision and recall into a single metric, calculated as their harmonic mean. It is particularly needful in cases involving imbalanced classes, as it balances he values of precision and recall, offering a more comprehensive assessment of the model's performance. Generally, the standard F1-score is used, but the parameter β can be modified to emphasize either precision or recall as needed.

$$F1_\beta = (1 + \beta^2) \cdot \frac{p \cdot r}{(\beta^2 \cdot p) + r}$$

For β = 1

$$F1score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

- Accuracy: Represent the proportion of correct predictions over the total predictions. It is calculated as the ratio of the sum of true positives and true negatives over the total number of predictions. However, accuracy may not always be suitable for imbalanced datasets, where one class is much more prevalent than the other, as it can appear high even when the model does not adequately detect the minority class.

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

- Area under the ROC curve (AUC-ROC): The ROC curve provides a graphical representation of the model's ability to discriminate between the two classes as the classification threshold varies. Specificity is plotted on the x-ax instead Sensitivity on y-ax. The overall effectiveness of the model is proportional to the area under the ROC curve (AUC), where a value close to 1 indicates excellent discrimination capacity (15).

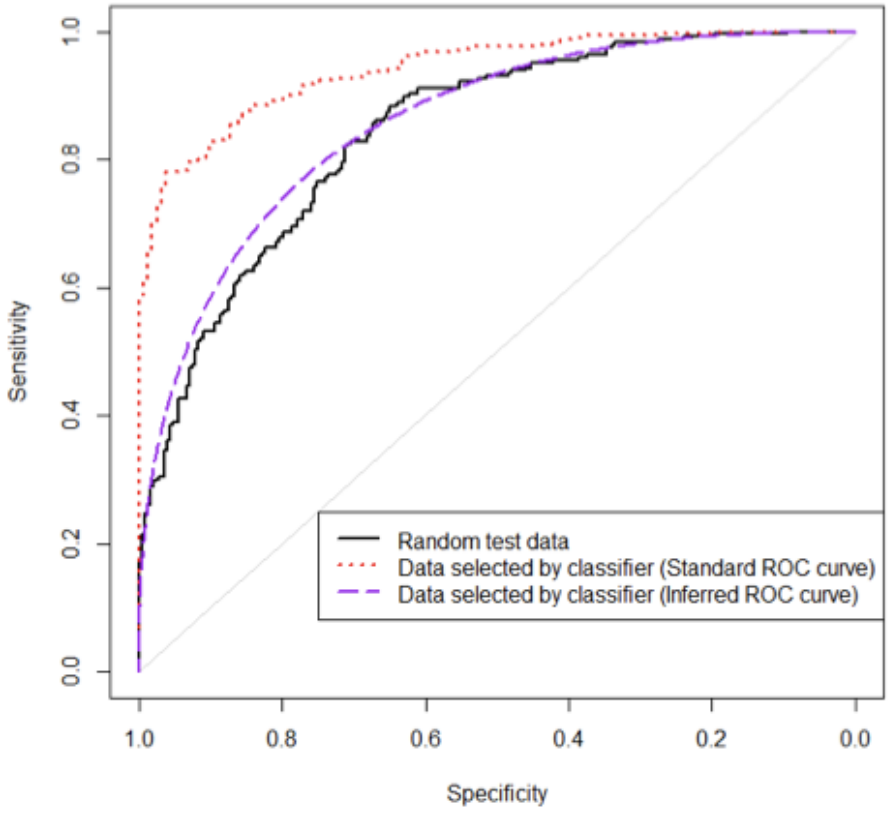


Figure 4.1: Example of ROC curve of (10)

4.2 Comparison of anomaly detection methods

The comparative analysis conducted by (14) focuses on evaluating various unsupervised anomaly detection algorithms across five databases: KC (34), NK (38), AL (11), IX (35), UN (28).

The twelve algorithms tested are grouped into six main families: classification-based, statistical-based, density-based, distance-based, clustering-based, and angle-based. In figure 4.2 the result of the metrics described above (section 4.1), such as precision, recall, accuracy, F1 score, AUC of the ROC curve, are presented based on these categories. Median scores are represented by the columns, while error bars pointed out the standard deviation of each result.

This study reveals that classification-based algorithms, like Isolation Forest and SVM, achieve the highest F1 scores, exceeding both statistical and density-based approaches. Although the last two categories mentioned have shown similar F1 scores, they rank slightly below the classification-based methods. The distance-based category also displays interesting features: despite having a lower median F1 score than classification method, they have a higher standard deviation, indicating that their performance varies significantly depending on parameter settings.

The clustering-based and angle-based methods exhibit the lowest F1 score, with angle-based demonstrating relatively lower F1 values at all.

Each algorithm has optimizable parameters that influence its final performance. For ROC curve and AUC metric, parameter tuning was conducted for each algorithm. The results indicate that classification ones are the most dependent on parameter choices, resulting in lower AUC performance compared to the other categories.

The observations suggest that classification-based methods are particularly effective for anomaly detection, providing high precision, recall, and accuracy. However, their dependency on parameter setting underscores the need for careful selection to achieve optimal outcomes. On the other side, density-based and statistical-based methods, while slightly less effective, offer a stable alternative.

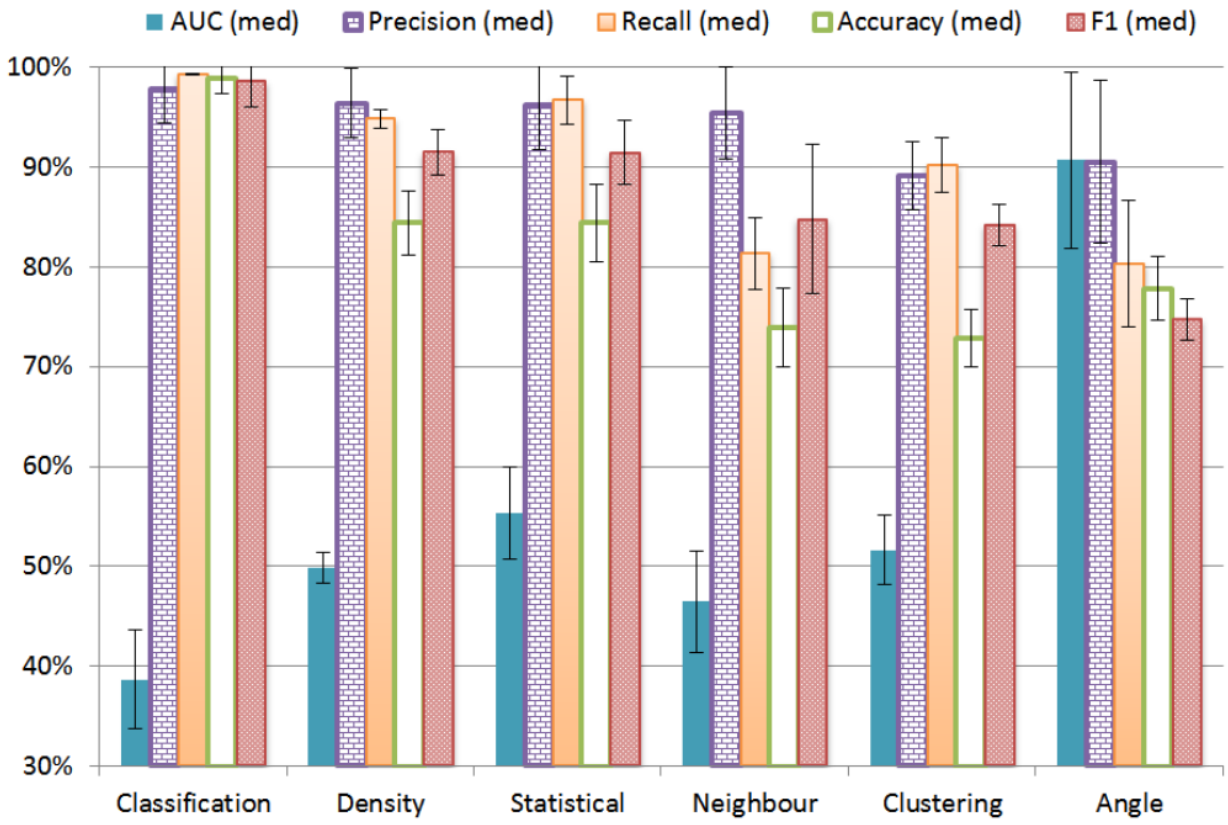


Figure 4.2: Results on all the datasets and all the attacks, grouped by algorithms families.

Algorithm	# Combinations	Family	AUC	Precision	Recall	Accuracy	F1
Isolation Forest [30]	8	Classification	37.2 ± 0.4	99.9 ± 0.3	99.3 ± 0.4	99.7 ± 0.3	99.6 ± 0.3
One-Class SVM [7]	1	Classification	53.4 ± 2.9	96.6 ± 3.2	99.3 ± 0.0	96.2 ± 3.2	98.0 ± 1.9
COF [41]	8	Density-Based	48.8 ± 1.7	93.6 ± 3.4	97.8 ± 0.1	91.7 ± 3.1	95.7 ± 2.0
ODIN [23]	8	Neighbour-Based	49.9 ± 1.7	96.6 ± 2.4	99.9 ± 0.4	89.8 ± 1.6	94.6 ± 1.1
HBOS [20]	1	Statistical	57.8 ± 5.5	92.6 ± 5.8	99.5 ± 4.3	89.2 ± 4.7	94.3 ± 4.8
rPCA [27]	1	Statistical	55.0 ± 4.0	97.5 ± 3.4	95.0 ± 1.0	83.1 ± 3.2	90.6 ± 2.0
LOF [9]	8	Density-Based	50.0 ± 1.3	96.6 ± 3.5	88.0 ± 1.1	81.3 ± 3.1	89.6 ± 2.1
LDCOF [4]	8	Clustering	49.9 ± 2.3	82.4 ± 1.8	94.4 ± 0.2	77.9 ± 1.5	87.4 ± 0.7
KNN [35]	8	Neighbour-Based	35.9 ± 6.7	91.9 ± 5.8	75.1 ± 3.4	71.4 ± 4.0	82.8 ± 4.3
K-Means [38]	8	Clustering	54.4 ± 8.9	95.7 ± 5.3	68.5 ± 2.8	65.6 ± 3.4	78.3 ± 3.5
ABOD [26]	8	Angle-Based	90.5 ± 7.8	69.2 ± 8.1	92.4 ± 8.3	90.0 ± 1.8	75.5 ± 10.2
FastABOD [26]	15	Angle-Based	86.4 ± 9.2	90.6 ± 7.8	77.4 ± 5.3	67.6 ± 3.2	74.7 ± 6.1

Figure 4.3: Metric scores (median ± std) for the 12 algorithms, ordered by F1 score

Chapter 5

ARIMA model and Conclusion

From the previous analyses, figures 4.2 and 4.3, it was determined that classification-based algorithms are the most effective for network anomaly detection. However, these algorithms suffer from a significant limitation: their performance heavily depends on the choice of parameters setting, which can vary widely depending on the specific problem approach. Consequently, it is not possible to define a ready-to-use, universal algorithm.

Given these constraints, the company opted for the adoption of a statistical-based algorithm. While statistical-based methods are second only to classification-based models in terms of effectiveness, they offer a key advantage: there is no need to predefine parameters at the outset.

This flexibility has enabled the development of a universal algorithm rooted in statistical methods, making it adaptable to a wide range of network anomaly detection scenarios.

5.1 Autoregressive Integrated Moving Average

According to (21), the ARIMA (Autoregressive Integrated Moving Average) algorithm is a widely model for the analysis and forecasting of time series data. Originally designed for economic trend extrapolation, ARIMA has proven to be a flexible solution, adaptable to anomaly detection applications in time series field. (36) described the three component as follows:

- Autoregression (AR): This component models data autocorrelation. If positive correlations remain ever after differencing, additional AR terms are necessary to capture them.

- Moving Average (MA): This is differencing. If it introduces negative correlations, as can occur in series with value jumps, the algorithm employs the MA component to adjust these spikes.
- Integration (I): Once the correct differencing order is identified, observations are reintegrated to outline the overall trend in the original series.

The ARIMA model is defined by three parameters: P, D, and Q, which represent:

- P: The number of autoregressive (AR) terms, indicating how many past values in the time series are used to predict the next one.
- D: The number of differencing operations required to make the time series stationary. Differencing involves calculating the differences between consecutive values in the series, a crucial step for simplifying the correlation patterns within the data.
- Q: The order of the moving average (MA), which adjust the model based on the past forecasting errors.

The construction of an ARIMA model generally starts with dividing the dataset into a training set (about 70%) and a test set (30%) to validate the model's accuracy. ARIMA forecasts future values based on the historical series data, aiming to minimize the discrepancy between the actual and predicted values.

An ARIMA(5,1,0) model, for example, features as autoregressive term of order 5, a differencing order of 1 (required to make the series stationary), and no moving average term.
During the training phase, the model is repeatedly re-fitted as new observation are joined to the dataset. After each forecast, the model updates previous data with newly observed values, progressively improving accuracy. The model then calculates the Root Mean Square Error (RMSE) to evaluate prediction accuracy.

Listing 5.1: Pseudocode for ARIMA Algorithm

```
# ARIMA
Inputs: series
Outputs: RMSE of the forecasted data
# Split data into 70% training and 30% testing data
1       size <- length(series) * 0.70
2       train <- series[0 to size]
3       test <- series[size to length(size)]
# Data structure preparation
4       history <- train
5       predictions <- empty
6       for each t in range(length(test)) do
7               model <- ARIMA(history, order=(p,d,q))
8                       model fit <- model.fit()
9                       hat <- model_fit.forecast()
10                      predictions.append(hat)
11                      observed <- test[t]
12                      history.append(observed)
13      end for
14      MSE = mean_squared_error(test, predicions)
15      RMSE = sqrt(MSE)
16      return RMSE
```

The provided pseudocode (36) outlines an algorithm designed to implement the ARIMA model for time series forecasting. The approach utilizes three parameters:

- p: Number of lag observations (autoregressive terms).
- d: Degree of differencing required to make the data stationary.
- q: Size of the moving average window.

This algorithm performs multistep out-of-sample forecasting with re-estimation, meaning that the ARIMA model is refitted after each step to incorporate newly observed data. The process ensures the model continuously updates to provide the best possible estimation.
Below is a step-by-step breakdown of the pseudocode:

- (Lines 1–3): The dataset is split into 70% training and 30% testing. The training set is used to build the ARIMA model, while the testing set evaluates its performance.
- (Lines 4–5): Two data structures are initialized:
  (a) History: Stores all observed values from the training set and test set iteratively.
  (b) Predictions: Stores predicted values for each step
- (Lines 6–12): An ARIMA model is built and fitted using the history dataset (Lines 7–8). The model generates a forecast hat, which is stored in the predictions list (Lines 9–10). The observed test value is appended to history to refine the model in the next iteration (Line 12). This loop ensures the model dynamically incorporates real-time information, simulating a real-world forecasting scenario.
- (Lines 14–16): The algorithm computes the Mean Squared Error (MSE) between predicted and actual test values. The Root Mean Square Error (RMSE), a common metric for evaluating forecast accuracy, is derived as the square root of the MSE.

5.2 Conclusion

5.2.1 Output of the project

In the initial phase of my project, I took inspiration from the pseudo-code algorithm 5.1 to enhance the ARIMA model using Python.

The company provided me with the access credential for ElasticSearch. Once connected, I could work with a designed dataset linked to a specific observer.IP. After a brief period spent familiarizing myself with the mechanism of ElasticSearch, I developed a query for that IP, utilizing an aggregation between the observer.ip and $date\_histogram$ with the $fixed\_interval$ parameter set to one day. this produced a $day - by - day$ time series visualization. The interval could also be adjusted to other increments, such as 30 minutes, two hours etc., allowing for flexibility in data granularity.

The output of ARIMA model I developed is displayed in figure 5.1

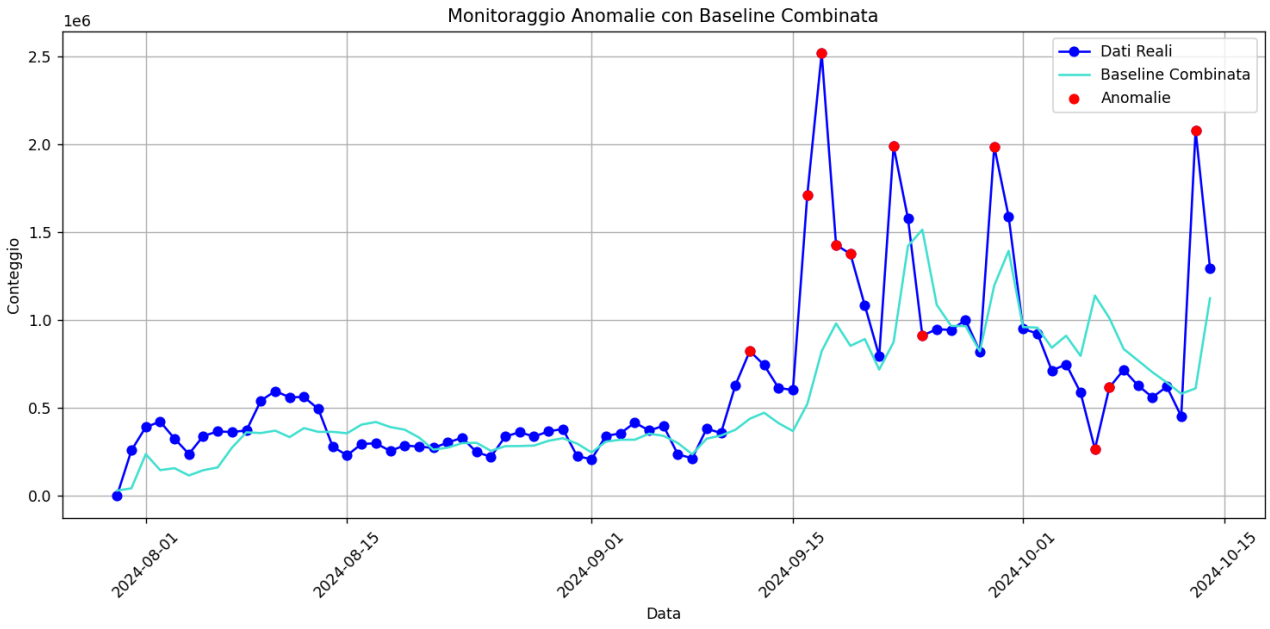


Figure 5.1: Results

Bibliography

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

[2] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. Anomaly detection in computer networks using machine learning techniques. *Journal of Network and Computer Applications*, 40:45–56, 2014.

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

[4] Mohiuddin Ahmed, Abdun Naser Mahmood, and Michael J. Maher. An efficient approach for complex data summarization using multiview clustering. In Jason J. Jung, Costin Badica, and Attila Kiss, editors, *Scalable Information Systems*, pages 38–47, Cham, 2015. Springer International Publishing.

[5] Frank J Anscombe. Rejection of outliers. *Technometrics*, 2(2):123–146, 1960.

[6] I. Balabine and A. Velednitsky. Method and system for confident anomaly detection in computer network traffic. Google Patents, 2015. Patent No. US20150254371A1.

[7] T. Barbariol, F. D. Chiara, D. Marcato, and G. A. Susto. A review of tree-based approaches for anomaly detection. In *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*, pages 149–185. Springer, 2022.

[8] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[9] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009.

[10] Jonathan Aaron Cook. Roc curves and nonrandom data. *Pattern Recognition Letters*, 85:35–41, 2017.

[11] Gideon Creech and Jiankun Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *2013 IEEE wireless communications and networking conference (WCNC)*, pages 4487–4492. IEEE, 2013.

[12] Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. 2000.

[13] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. *A Geometric Framework for Unsupervised Anomaly Detection*, pages 77–101. Springer US, Boston, MA, 2002.

[14] Filipe Falcão, Tommaso Zoppi, Caio Barbosa Viera Silva, Anderson Santos, Baldoino Fonseca, Andrea Ceccarelli, and Andrea Bondavalli. Quantitative comparison of unsupervised anomaly detection algorithms for intrusion detection. In *Proceedings of the 34th ACM/SIGAPP symposium on applied computing*, pages 318–327, 2019.

[15] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[16] J.W. Han, M. Kamber, and J. Pei. *Data Mining Concepts and Techniques*. Morgan Kaufmann Publishers, Waltham, 3rd edition, 2012.

[17] D. Hawkins. *Identification of Outliers*. Chapman and Hall, Kluwer Academic Publishers, Boston/Dordrecht/London, 1980.

[18] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, *Data Warehousing and Knowledge Discovery*, pages 170–180, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[19] Katherine Heller, Krysta Svore, Angelos D Keromytis, and Salvatore Stolfo. One class support vector machines for detecting anomalous windows registry accesses. 2003.

[20] Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, volume 6. Citeseer University Park, PA, USA, 2003.

[21] Mark Kasunic, James McCurley, Dennis Goldenson, and David Zubrow. An investigation of techniques for detecting data anomalies in earned value management data. *Software Engineering Institute: Pittsburgh, PA, USA*, 2011.

[22] K. Kendall. A database of computer attacks for the evaluation of intrusion detection systems. In *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX)*, pages 12–26, 1999.

[23] Edwin M Knorr, Raymond T Ng, and Vladimir Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.

[24] Christopher Krügel, Darren Mutz, William K. Robertson, and Fredrik Valeur. Bayesian event classification for intrusion detection. *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 14–23, 2003.

[25] Christopher Krügel, Thomas Toth, and Engin Kirda. Service specific anomaly detection for network intrusion detection. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 201–208, 2002.

[26] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the 2003 SIAM international conference on data mining*, pages 25–36. SIAM, 2003.

[27] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008.

[28] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015.

[29] Gerhard Münz, Sa Li, and Georg Carle. Traffic anomaly detection using k-means clustering. In *Gi/itg workshop mmbnet*, volume 7, 2007.

[30] Joshua Oldmeadow, Siddarth Ravinutala, and Christopher Leckie. Adaptive clustering for network intrusion detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 255–259. Springer, 2004.

[31] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. 1999.

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

[33] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[34] Saharon Rosset and Aron Inger. Kdd-cup 99: knowledge discovery in a charitable organization's donor database. *ACM SIGKDD Explorations Newsletter*, 1(2):85–90, 2000.

[35] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012.

[36] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pages 1394–1401. Ieee, 2018.

[37] Iwan Syarif, Adam Prugel-Bennett, and Gary Wills. Unsupervised clustering approach for network anomaly detection. In *Networked Digital Technologies: 4th International Conference, NDT 2012, Dubai, UAE, April 24-26, 2012. Proceedings, Part I 4*, pages 135–145. Springer, 2012.

[38] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009.

[39] Nong Ye and Qiang Chen. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and reliability engineering international*, 17(2):105–112, 2001.