People’s Democratic Republic of Algeria
Ministery of Higher Education and Scientific Research
Mohamed El Bachir El Ibrahimi University of Borj Bou Arréridj
Faculty of Mathematics and Informatics

Informatics Department

DISSERTATION
Presented in fulfillment of the requirements of obtaining the degree
Master in Computer Science
Specialty: Networks and Multimedia

THEME
Anomaly Detection for Network Security

Presented by:

BELKAALOUL Abdelkoudous

SANAA El Hassen Abdeldjalil

Publicly defended on: dd/mm/yyyy

In front of the jury composed of:

President: …………………………………..

Examiner: ………………………………

Supervisor :………………………………

2023/2024

Dedication

To my mother and father, for their endless love, support, and encouragement.

To my brother Abdeladhim, for always believing in me and inspiring me to strive for greatness.

To my sister Nedjma, for her unwavering support and understanding.

To my niece Shahad, the daughter of my brother.

To the soul of my dear friend Aymen.

To Mr. Benaouda Nadjib, for his invaluable guidance and mentorship.

To Bellucci, for being a source of inspiration and motivation throughout this journey.

And to all my friends, your friendship, laughter, and support have been invaluable.

I dedicate the fruits of these years of study.

Abdelkoudous

Dedication

First and foremost, I thank our God who has given us patience and strength throughout our project.

I dedicate this work to:

My dear mother, whose presence by my side has always been my source of strength to face various obstacles.

My father, who taught me, supported me, and guided me.

My brothers and sister, for their constant encouragement and support.

My friends and the teachers of the Computer Science department, for their invaluable assistance and guidance.

Abdeldjalil

Acknowledgment

At the end of this work, we thank Allah who has given us the strength and patience to overcome all the difficulties during these long years of study.

We would like to express our warmest thanks to our teacher and supervisor, Mrs. Djamila MOHDEB, for her support and valuable dedicated advice, her constant encouragement, and her assistance throughout the entire period of work. During our studies, we have benefited from your clear and precise teaching.

Our thanks to all the members of the jury, who honored us by carefully studying our work.
We also thank all the professors of the Computer Science department at Mohamed El Bachir El Ibrahimi University of Bordj Bou Arréridj.

Finally, we thank our parents, all our relatives, and friends for their support and encouragement.

Abstract

Network security is increasingly challenged by sophisticated cyber threats, necessitating advanced methods for anomaly detection. In this project, we developed an anomaly detection application specifically designed for cybersecurity datasets.

Our contribution includes a Python-based application that integrates both supervised and unsupervised anomaly detection techniques, leveraging statistical, clustering, and machine learning approaches. The application is capable of analyzing both pre-existing and synthetic datasets, providing comprehensive anomaly detection and actionable insights for enhancing cyber defenses.

We evaluated the application through a detailed case study in network security, applying it to real-world scenarios. The results demonstrate the effectiveness of our application in identifying anomalies and potential threats within network traffic. The flexibility in selecting various anomaly detection methods ensures adaptability to diverse cybersecurity datasets, underscoring the practical relevance and robustness of our approach.

Keywords: anomaly detection, network security, cybersecurity, supervised learning, unsupervised learning, machine learning, statistical methods, clustering, Python, synthetic datasets, cyber threats, data analysis.

Résumé

La sécurité des réseaux est de plus en plus confrontée à des menaces cybernétiques sophistiquées, nécessitant des méthodes avancées de détection d'anomalies. Dans ce projet, nous avons développé une application de détection d'anomalies spécifiquement conçue pour les ensembles de données de cybersécurité.

Notre contribution comprend une application basée sur Python qui intègre à la fois des techniques de détection d'anomalies supervisées et non supervisées, en utilisant des approches statistiques, de clustering et d'apprentissage automatique. L'application est capable d'analyser à la fois des ensembles de données existants et synthétiques, fournissant une détection complète des anomalies et des informations exploitables pour renforcer les défenses cybernétiques.

Nous avons évalué l'application à travers une étude de cas détaillée en sécurité réseau, en l'appliquant à des scénarios réels. Les résultats démontrent l'efficacité de notre application pour identifier les anomalies et les menaces potentielles dans le trafic réseau. La flexibilité dans le choix des différentes méthodes de détection d'anomalies garantit une adaptabilité aux divers ensembles de données de cybersécurité, soulignant la pertinence pratique et la robustesse de notre approche.

Mots-clés: détection d'anomalies, sécurité réseau, cybersécurité, apprentissage supervisé, apprentissage non supervisé, apprentissage automatique, méthodes statistiques, clustering, Python, ensembles de données synthétiques, menaces cybernétiques, analyse de données.

ملخص

يتعرض أمن الشبكات بشكل متزايد لتحديات من التهديدات السيبرانية المعقدة، مما يستدعي وجود أساليب متقدمة لاكتشاف الشوائب .في هذا المشروع، قمنا بتطوير تطبيق لكشف الشوائب مصمم خصيصًا لمجموعات البيانات الخاصة بأمن المعلومات.

تشمل مساهمتنا تطبيقًا مبنيًا على لغة البرمجة بايثون يدمج كل من تقنيات الكشف عن الشوائب التي تم توجيهها وغير الموجهة، باستخدام الأساليب الإحصائية وتقنيات التجميع وتعلم الآلة .يستطيع التطبيق تحليل كل من مجموعات البيانات الحالية والمزيفة، مما يوفر كشفًا شاملاً للشوائب ورؤى قابلة للتنفيذ لتعزيز الدفاعات السيبرانية.

قمنا بتقييم التطبيق من خلال دراسة حالة مفصلة في أمن الشبكات، حيث تم تطبيقه على سيناريوهات واقعية .تظهر النتائج فعالية تطبيقنا في تحديد الشوائب والتهديدات المحتملة داخل حركة المرور على الشبكة .يضمن مرونة اختيار مختلف أساليب كشف الشوائب قابلية التكيف مع مجموعات البيانات السيبرانية المتنوعة، مما يؤكد على الأهمية العملية والقوة التحليلية لنهجنا.

الكلمات المفتاحية :كشف الشوائب، أمن الشبكات، الأمن السيبراني، التعلم الإشرافي، التعلم غير الإشرافي، التعلم الآلي، الأساليب الإحصائية، التجميع، بايثون، مجموعات بيانات مزيفة، التهديدات السيبرانية، تحليل البيانات.

Table of contents

Abbreviations list

HBOS: Histogram-based Outlier Score

LOF: Local Outlier Factor

OMSVM: One Class Support Vector Machine

CBLOF: Cluster based Local Outlier Factor

KNN: K-Nearest Neighbour

IForest : isolation forest

Spyder : Scientific Python Développent Environnement.

List of figures

List of tables

List of algorithms

General Introduction

In today's digital age, network security is a critical concern for organizations worldwide. As cyber threats become more sophisticated and pervasive, traditional security measures are often insufficient to detect and mitigate these evolving threats. Anomaly detection has emerged as a vital technique in the realm of cybersecurity, capable of identifying unusual patterns or behaviors that may indicate malicious activities. This thesis focuses on the development and application of an anomaly detection system tailored specifically for network security.

1. Context of the Study

The rapid expansion of the internet and the increasing complexity of network infrastructures have made them prime targets for cyber-attacks. Organizations are constantly at risk of data breaches, malware infections, and other forms of cyber threats that can compromise sensitive information and disrupt operations. Traditional security systems, which rely on predefined rules and signatures, struggle to keep up with the dynamic nature of modern cyber threats. Anomaly detection offers a promising solution by identifying deviations from normal behavior, potentially uncovering previously unknown threats.

2. Problem Statement

Despite the advancements in anomaly detection techniques, there remains a significant challenge in effectively applying these methods to network security. Existing solutions often lack the flexibility to adapt to the diverse and dynamic nature of network traffic. Additionally, the integration of supervised and unsupervised learning methods within a single framework is rarely achieved, limiting the robustness and effectiveness of current systems. This thesis addresses these gaps by developing a comprehensive anomaly detection application that leverages a wide range of techniques and is specifically designed for cybersecurity datasets.

3. Objective of the Study

The primary objective of this project is to create an anomaly detection application that combines both supervised and unsupervised techniques, drawing from statistical, clustering, and machine learning approaches. Our application aims to analyze both pre-existing and synthetic cybersecurity datasets, providing a robust and flexible tool for identifying anomalies and potential threats within network traffic. By offering comprehensive anomaly detection and actionable insights, the application aims to enhance cyber defenses and facilitate proactive measures against cyber threats.

4. Methodology

The development of our anomaly detection application follows a systematic approach that incorporates loading and preparing data tasks for both historical and synthetic datasets, integrating supervised and unsupervised anomaly detection methods, including statistical analysis, clustering algorithms, and machine learning models, into a unified application, implementing and testing, and finally conducting a detailed case study in network security to demonstrate the practical application of the system.

5. Organization of the Thesis

The remainder of this thesis is organized into four chapters:

- Chapter 1 provides an overview of the fundamental concepts in anomaly detection.
- Chapter 2 focus on applications of anomaly detection techniques in network security.
- Chapter 3 details the design and architecture of our anomaly detection application.
- Chapter 4 presents the implementation details of our application, including the user interface and functionality. We also focus on the case study conducted to evaluate our application then provide a comprehensive analysis of the results, compare the performance of different anomaly detection methods, and discuss the practical implications of our findings.

Chapter 01: Anomaly Detection

1.1. Introduction

Anomaly detection is an essential task for identifying atypical behaviors in data. This chapter provides an overview of the fundamental concepts in anomaly detection. It covers the definitions, types, and importance of anomaly detection in various domains. The aim is to gain an understanding of the basic process of identifying deviations from normal behavior, laying the groundwork for more advanced discussions.

1.2. Definition of Anomaly

An anomaly, in the context of data analysis and machine learning, refers to a data point or a subset of data points that significantly deviates from the expected pattern or behavior of the majority of the data [1]. Anomalies are often indicative of rare or unusual events, and they can signal important insights such as system malfunctions, fraud, or changes in underlying patterns.

1.3. Anomaly Detection

Anomaly detection, also known as **outlier detection**, is a technique used in data analysis and machine learning to identify data points or patterns that deviate significantly from the norm or expected behavior [2]. These deviations are often referred to as **anomalies** or **outliers** and could indicate unusual events, errors, or potential fraud in the data.

Anomaly detection has applications across various industries, including finance (for fraud detection), manufacturing (to identify defects or equipment malfunctions), cybersecurity (to detect unusual network activity), and healthcare (to identify abnormal patient conditions). It plays a crucial role in improving data quality, enhancing decision-making, and optimizing machine learning performance.

1.4. Key Characteristics of Anomalies

- **Deviation**: Anomalies deviate from the normal behavior or distribution of the data.
- **Rarity**: They occur infrequently within the dataset.
- **Unexpectedness**: Anomalies are not anticipated based on the existing data trends.

1.5. Importance of Anomaly Detection Task

Anomaly detection plays a crucial role in data analysis and machine learning for several reasons:

- Detecting anomalies helps identify unusual patterns, outliers, or unexpected events within a dataset.
- Detecting anomalies helps improve data quality by addressing errors, noise, or inconsistencies in the data.
- Anomalies can impact decision-making processes. By identifying them, organizations can make informed decisions.
- Anomaly detection is vital for identifying suspicious or malicious behavior. It helps detect network intrusions, unauthorized access, or unusual system activity. Cybersecurity systems rely on anomaly detection to protect against threats.
- Anomalies in financial data can signal potential risks. Detecting them helps manage risk exposure, whether in stock market trading, credit scoring, or insurance underwriting.
- In Natural Language Processing (NLP), detecting unusual language patterns can help identify spam emails, fake reviews, or abnormal chatbot interactions.

1.6. Aspects of the Anomaly Detection Problem

In this section, we will explore various aspects of the anomaly detection problem. Each aspect plays a crucial role in designing and implementing effective anomaly detection systems.

1.6.1. Nature of the Data

The nature of the data is a crucial aspect in anomaly detection. Data can be structured, semi-structured, or unstructured, and their quality and quantity can influence the accuracy of anomaly detection models. Data characteristics such as dimensionality and distribution also play an important role [4].

1.6.2. Types of Anomalies

There are several types of anomalies: point anomalies, contextual anomalies, and collective anomalies. Each of these anomalies requires specific detection techniques tailored to their unique characteristics [2].

❖ Point Anomalies

Point anomalies, refer to individual data points that deviate significantly from the majority of data points in a dataset. These anomalies are isolated and can be identified by their deviation in value from the expected range or pattern of the data [2].

For example, consider a dataset that records the daily temperature of a city over a year. The temperature usually ranges between -10°C to 35°C. If one day the recorded temperature is 100°C, this data point would be considered a point anomaly because it significantly deviates from the typical temperature range.

❖ Contextual Anomalies

Contextual anomalies, also known as conditional anomalies, are data points that significantly deviate from expected behavior only within a specific context or condition. Unlike point anomalies, contextual anomalies may appear normal in a general context but become abnormal when examined in relation to additional contextual attributes such as time or location [2].

For example, a temperature of 30°C might be normal in the summer but abnormal in the winter.

Detecting contextual anomalies is particularly useful in fields such as environmental monitoring, fraud detection, and network management, where data behavior can vary depending on the context [2].

❖ Collective Anomalies

Collective anomalies occur when a set of data points, considered together, significantly deviates from the expected behavior, even if the individual data points might appear normal on their own. These anomalies are identified by analyzing groups of data rather than isolated points, allowing for the detection of abnormal behaviors at the dataset level.

For example, a series of small financial transactions made within a short period may seem normal individually, but when taken together, they could indicate fraudulent behavior. Detecting collective anomalies is particularly important in fields such as cybersecurity, fraud detection, and industrial system monitoring, where abnormal patterns may only emerge when a group of data is considered as a whole [2].

1.6.4. Output of Anomaly Detection Techniques

In anomaly detection, the manner in which anomalies are represented in the output is crucial. Typically, anomalies are represented in one of two ways [8]:

- ❖ **Scoring method** : Scoring-based anomaly detection techniques allocate an anomaly score to each data point. These scores are then ordered, enabling an analyst to identify anomalies either by direct selection or by setting a threshold.
- ❖ **Binary Classification**: Binary classification techniques output results in a simple binary format: either normal or anomalous.

1.6.5. Anomaly Detection Process

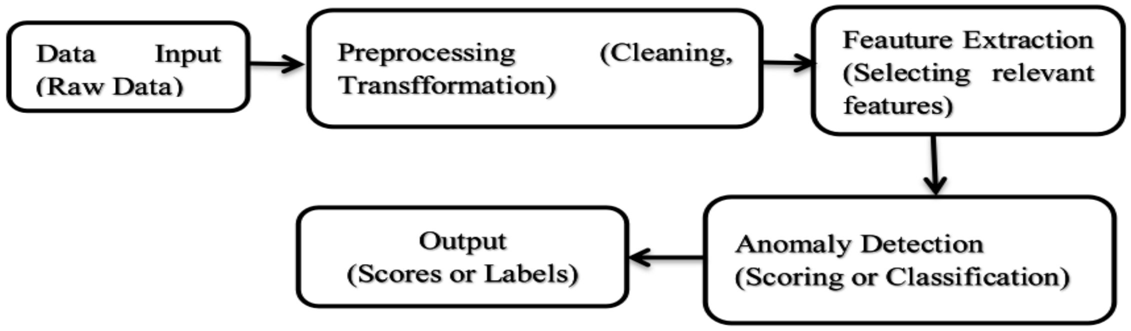

Data Input (Raw Data)
Preprocessing (Cleaning, Transfformation)
Feauture Extraction (Selecting relevant features)
Anomaly Detection (Scoring or Classification)
Output (Scores or Labels)

Figure 1 Anomaly detection process

1. **Data Input**: The raw data collected from various sources (e.g., sensor data, transaction logs, user activity).
2. **Preprocessing**: The raw data is cleaned and transformed to remove noise and handle missing values, making it suitable for analysis.
3. **Feature Extraction**: Relevant features are selected or extracted from the preprocessed data to help the anomaly detection algorithms.
4. **Anomaly Detection**: Algorithms are applied to detect anomalies. This step can involve scoring each data point or classifying them as normal or anomalous.
5. **Output**: The results of the anomaly detection are represented as either anomaly scores or binary labels (normal or anomalous).

1.7. Challenges of Anomaly Detection

Anomaly detection encounters several challenges that can impede its effectiveness across various applications. These challenges have been extensively discussed in academic literature and include:

- **Scalability**: Anomaly detection algorithms must be capable of handling large-scale datasets efficiently to meet the demands of modern data-driven applications [9].
- **Imbalanced Data**: Anomalies are often rare events compared to normal instances, leading to imbalanced datasets. Dealing with imbalanced data requires specialized techniques to prevent biased models and ensure accurate anomaly detection [10].
- **Complexity of Anomalies**: Anomalies can exhibit intricate patterns and behaviors, making them challenging to detect using conventional methods. Advanced anomaly detection techniques capable of capturing complex anomalies have been proposed by researchers such as Pang et al. (2016) in their work on detecting complex anomalies in sensor data [11].
- **Interpretability**: Understanding and interpreting detected anomalies are crucial for taking appropriate actions. However, some anomaly detection algorithms may produce complex or opaque results, making it challenging for users to interpret and trust the outcomes [12].

1.9. Conclusion

By delving into fundamental anomaly detection principles, we've established a solid foundation for understanding its applications across various domains. These concepts lay the groundwork for more specialized discussions, preparing us to explore anomaly detection's role in network security.

Chapter 02: Anomaly Detection and Network Security

2.1. Introduction

Focusing on the specific application of anomaly detection within network security, this chapter explores the unique challenges and requirements of detecting anomalies in network traffic. It discusses common threats, key detection techniques, and the significance of anomaly detection methods in safeguarding network integrity and preventing cyber attacks.

2.2. Cybersecurity Fundamentals

2.2.1. Definition of Cybersecurity

Cybersecurity refers to any technology, measure, or practice aimed at preventing cyberattacks or mitigating their impact. Its primary goal is to protect individuals' and organizations' systems, applications, computing devices, sensitive data, and financial assets against various threats [13].

Cybersecurity in the context of network security refers to the practice of protecting the integrity, confidentiality, and availability of information as it is transmitted, processed, and stored within and across computer networks. This involves implementing a range of technologies, processes, and policies designed to defend against unauthorized access, misuse, disruption, modification, or destruction of network resources, data, and services.

2.2.2. Cybersecurity Goals

- **Confidentiality**: Confidentiality ensures that sensitive information is accessible only to authorized individuals.
- **Integrity**: Ensuring data integrity means that information remains accurate, unaltered, and trustworthy unless modified by authorized individuals.

- **Availability**: Cybersecurity aims to ensure that systems and services are available when needed.
- **Traceability**: Traceability involves tracking and auditing system activities to identify any unauthorized or suspicious behavior. It helps in investigating incidents and maintaining accountability.

2.2.3. Cyberattack

A cyberattack is a deliberate attempt by individuals or organizations to breach the information system of another individual or organization. These attacks aim to steal, alter, or destroy data, disrupt operations, or gain unauthorized access to computer systems and networks. Cyberattacks can target a wide range of entities, including individuals, companies, governments, and critical infrastructure.

2.2.4. Network Attacks

Network attacks are unauthorized actions on the digital assets within an organizational network. Malicious parties execute these attacks to alter, destroy, or steal private data.

2.2.5. Types of Network Attacks

There are two main types of network attacks:

❖ **Passive Network Attacks**

In passive attacks, malicious actors gain unauthorized access to networks, monitor traffic, and steal private data without altering it. Essentially, they eavesdrop on sensitive information.

Examples include eavesdropping on network communications or stealing data without leaving any noticeable trace.

❖ **Active Network Attacks**

Active attacks involve modifying, encrypting, or damaging data within the network. Common examples of active network attacks include:

- **DoS (Denial-of-Service) and DDoS (Distributed Denial-of-Service) Attacks**: Overwhelm system resources to the point where it cannot respond to legitimate service requests.
- **Man-in-the-Middle (MITM) Attacks**: Intercept data transmitted between two parties, compromising privacy and integrity.
- **SQL Injection Attacks**: Exploit unmoderated user data inputs to manipulate databases.
- **Phishing Attacks**: Deceptive emails or websites trick users into revealing sensitive information.
- **Malware Attacks**: Introduce malicious software (e.g., Trojan horses) into the network.
- **URL Spoofing Attacks**: Manipulate URLs to deceive users.
- **Web Application Vulnerability Attacks**: Exploit weaknesses in web applications.
- **Drive-by Attacks**: Automatically infect devices when users visit compromised websites.
- **Eavesdropping Attacks**: Intercept and monitor network traffic.

2.3. Role of Anomaly Detection in Network Security

Anomaly detection is a crucial component in network security for identifying unusual patterns or behaviors that may indicate the presence of malicious activities, potential threats, or network intrusions. Key roles include:

- **Early Threat Detection**
  - Proactive Defense: Anomaly detection systems can identify abnormal behaviors in real-time, allowing for early detection of potential threats before they cause significant damage.
  - Zero-Day Attack Mitigation: By detecting deviations from normal patterns, anomaly detection can help identify zero-day attacks that exploit unknown vulnerabilities.
- **Complementing Signature-Based Detection**
  - Enhanced Security: While signature-based detection systems rely on known patterns of malware or attack signatures, anomaly detection can identify novel or unknown threats.
  - Comprehensive Coverage: Combining both anomaly and signature-based detection provides a more robust security posture.
- **Behavioral Analysis**
  - User and Entity Behavior Analytics (UEBA): Anomaly detection systems can monitor user behavior, identifying activities that deviate from established norms, such as unusual login times or accessing atypical resources.
  - Network Traffic Analysis: Continuous monitoring of network traffic to identify unusual patterns that may indicate data exfiltration, lateral movement, or communication with malicious servers.
- **Incident Response and Forensics**
  - Rapid Response: Quick identification of anomalies enables faster incident response, containment, and remediation of threats.
  - Investigative Insights: Anomaly detection provides valuable insights for forensic analysis, helping to understand the nature and extent of an attack.

❖ **Reducing False Positives**

- Adaptive Learning: Modern anomaly detection systems use machine learning algorithms that adapt to evolving network conditions and user behaviors, reducing the number of false positives and improving detection accuracy.

❖ **Compliance and Auditing**

- Regulatory Requirements: Many regulations and standards (e.g., GDPR, HIPAA) require continuous monitoring and anomaly detection to ensure the security and privacy of sensitive data.
- Audit Trails: Anomaly detection systems maintain logs of detected anomalies, which are useful for compliance audits and reporting.

2.4. Types of Network Anomalies

- **Network Traffic Anomalies**: Detect unusual spikes, drops, or patterns in network traffic volume, protocols, or ports.
- **User Behavior Anomalies**: Identify abnormal login times, access requests, or data transfers.
- **System Resource Anomalies**: Monitor CPU, memory, and disk usage for unexpected spikes.
- **Application-Level Anomalies**: Detect irregularities in application logs or transactions.
- **Protocol Violations**: Identify non-compliant or suspicious network protocol behavior.

2.5. Anomaly Detection for Network Security: Use Cases

- **Intrusion Detection**: Anomaly detection helps identify unauthorized access attempts, port scans, or brute-force attacks.
- **Insider Threat Detection**: Detects abnormal behavior by employees or contractors.

- **Malware Detection**: Identifies unusual patterns associated with malware activity.
- **Fraud Detection**: In financial systems, it detects fraudulent transactions.
- **Industrial Control Systems (ICS) Security**: Monitors deviations in critical infrastructure networks.

2.6. Approaches of Anomaly Detection

Anomaly detection in network security employs a variety of techniques to identify unusual patterns or behaviors that may indicate security threats. These techniques can be broadly categorized into statistical methods, machine learning approaches, rule-based systems, and hybrid models.

2.6.1. Statistical Approaches

Statistical techniques look for anomalies by applying the concepts of probability and statistical reasoning (i.e. comparing data points to the overall distribution). They offer measurable metrics of normalcy and deviation and are particularly effective at objectively identifying outliers in a variety of data sets. Some of the key statistical techniques used are [8]:

- **Descriptive Statistics**: In descriptive statistics, normal behavior is defined and outliers are identified by using summary statistics such as mean, median, standard deviation, and interquartile range.
- **Hypothesis Testing**: Creating and evaluating hypotheses in order to ascertain whether an observed data point substantially deviates from the expected distribution is known as hypothesis testing. Methods include the Grubbs test and Z-scores.
- **Box Plots:** A box plot (see Figure 1), sometimes called a box-and-whisker plot, is a visual depiction of a dataset's distribution. Essential summary statistics including the median, quartiles, and outliers are shown. The interquartile range (IQR) of the data is represented by a box in the plot, and the median is shown by a line inside the box. The

range of the data outside the IQR is represented by whiskers that extend from the box; optional points or dots are used to indicate outliers [18].

- **HBOS** : HBOS (Histogram-based Outlier Score) is an anomaly detection method used to find outliers in a dataset. Every feature in the dataset is given a histogram, and the probability density of the data inside these histograms is used to determine the outlier scores. When compared to normal data points, HBOS assumes that outliers have lower probability densities [19].
- **Principal Component Analysis (PCA)**: By highlighting anomalies and identifying important patterns based on departures from the principal components, PCA reduces dimensionality to identify key patterns and highlight anomalies based on deviations from the principal components.
- **Density Estimation**: Determines areas of low probability, which point to anomalies, by estimating the probability density function of the data. Among the methods is kernel density estimation.
- **Time Series Analysis**: Time series analysis examines patterns in temporal data and looks for anomalies when these patterns deviate. approach variations from the anticipated time-series behavior, such as the ARIMA model.

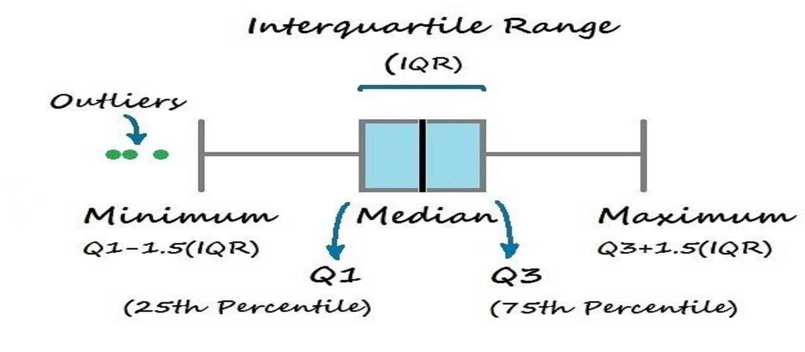


Figure 2: Box plot

2.6.2. Clustering Approaches

A clustering-based anomaly detection method makes use of the idea of assembling related data points into clusters. Anomalies are found by comparing the data points to the normal patterns found within their clusters after they have been grouped.

Clustering approach for anomaly detection encompasses several models. The most frequently employed methods are:

- **K-Means**: K-Means is a clustering algorithm used to partition a dataset into K clusters. The algorithm iteratively assigns each data point to the nearest cluster centroid and then recalculates the centroid of each cluster based on the data points assigned to it. This process continues until the centroids no longer change significantly or until a maximum number of iterations is reached. K-Means aims to minimize the sum of squared distances between data points and their respective cluster centroids. Points far from their assigned cluster centroid are considered anomalies.

- **DBSCAN**: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) identifies clusters based on dense regions of data points and labels points in sparse regions as noise (outliers). Outliers are explicitly identified as points not belonging to any dense cluster.

- **Local Outlier Factor (LOF)**: LOF measures the local deviation of a data point's density compared to its neighbors. It compares the Local Reachability Density (LRD) of a point to that of its neighbors. A LOF score close to 1 indicates that the LRD around the point is similar to its neighbors, suggesting it's not an outlier.

2.6.3 Rule-Based Systems

Rule-based approaches for anomaly detection rely on predefined rules or heuristics derived from expert knowledge or historical data to identify unusual or suspicious behavior. These systems rely on Boolean logic, thresholds, and pattern matching to detect anomalies based on specified criteria.

Examples

- ✓ Alert if more than five failed login attempts occur within 10 minutes. (**Predefined Rules**)
- ✓ If a user accesses the system from multiple geographically distant locations within a short period, mark it as suspicious. (**Heuristics**)
- ✓ Alert if an IP address attempts to access more than three distinct servers AND the access occurs outside business hours. (**Boolean Logic**)
- ✓ Raise an alert if CPU usage exceeds 90% for more than 10 minutes. (**Threshold**)
- ✓ Identify traffic patterns resembling known DDoS attack signatures. (**Pattern Matching**)

While rule-based systems are simple, transparent, and computationally efficient, they require regular updates to handle evolving threats and can suffer from high false positive and negative rates. Despite these challenges, rule-based methods remain a fundamental tool in the anomaly detection toolkit.

2.6.4. Other Techniques

Additional famous methods for detecting anomalies include Isolation Forests, Neural Networks, Support Vector Machines (SVM).

❖ Isolation Forests

Isolation Forests, or IForests, utilize a group of binary trees known as isolation trees (iTrees) to isolate instances within a dataset. This method recursively partitions the data through random feature selection and random split values. IForests are particularly efficient because they focus on isolating data points rather than relying on similarity or distance measures. The underlying principle is that anomalies are few and different, making them easier to isolate. This method encodes variable-length sequences into a single, fixed-size vector, facilitating anomaly detection [21].

❖ **Support Vector Machines (SVM)**

Support Vector Machines (SVM) are a set of supervised learning methods used for classification, regression, and outlier detection. For anomaly detection, SVM is typically used in its one-class form, known as One-Class SVM (OCSVM) [25].

One-Class Support Vector Machine (One-Class SVM) operates by learning a decision boundary that encapsulates the normal data points within a high-dimensional feature space. During training, the One-Class SVM aims to separate the normal data from the origin with maximum margin, assuming that the majority of the training data represents normal behavior. Points that lie outside this learned boundary during inference are considered anomalies. This method is effective for identifying outliers in scenarios where the training data primarily consists of normal instances.

❖ **Autoencoders**

Autoencoders are neural networks used for anomaly detection, consisting of an encoder and a decoder. The encoder compresses input data into a lower-dimensional latent space, while the decoder reconstructs the original data from this representation. During training, the autoencoder learns to minimize the reconstruction error using normal data. The key assumption is that the model will effectively reconstruct normal data but struggle with anomalies, resulting in higher reconstruction errors. Anomalies are identified based on these elevated reconstruction errors, distinguishing them from normal data points.

2.3. Conclusion

Through an exploration of network security challenges and anomaly detection techniques, we've underscored the critical importance of robust detection methods in safeguarding networks. This understanding sets the stage for further examination of practical implementations in our developed system.s based on different approaches

Chapter 03: Methodology and System Design

3.1. Introduction

This chapter introduces our methodology for developing an anomaly detection application for network security. It details the system's functionalities, including data manipulation, visualization, and the integration of various detection techniques. Additionally, it outlines the performance evaluation metrics implemented to assess the effectiveness of the system in identifying network anomalies.

3.2. Project Description

In our project, we've developed an innovative anomaly detection application tailored specifically for cybersecurity datasets. Our Python-based application incorporates both supervised and unsupervised anomaly detection techniques, drawing from statistical, clustering, and machine learning approaches. It is designed to analyze **pre-existing security datasets** as well as generate **synthetic datasets** for experimentation and evaluation purposes.

The primary objectives of our application are as follows:

- ✓ **Comprehensive Anomaly Detection**: Our application thoroughly analyzes cybersecurity datasets to identify deviations from normal patterns, regardless of whether the data is **historical** or **synthetic**, employing a spectrum of anomaly detection methods.
- ✓ **Insightful Analysis for Proactive Defense**: Providing security analysts with actionable insights derived from anomaly detection algorithms, our application facilitates proactive measures to strengthen cyberdefenses and mitigate potential threats.

The core components of our system, encapsulated in Figure 3, outline the systematic approach we employ to achieve these objectives. Notably, our application offers flexibility in selecting from a diverse array of anomaly detection methods, ensuring adaptability to the nuances of cybersecurity datasets, particularly those pertaining to network security.

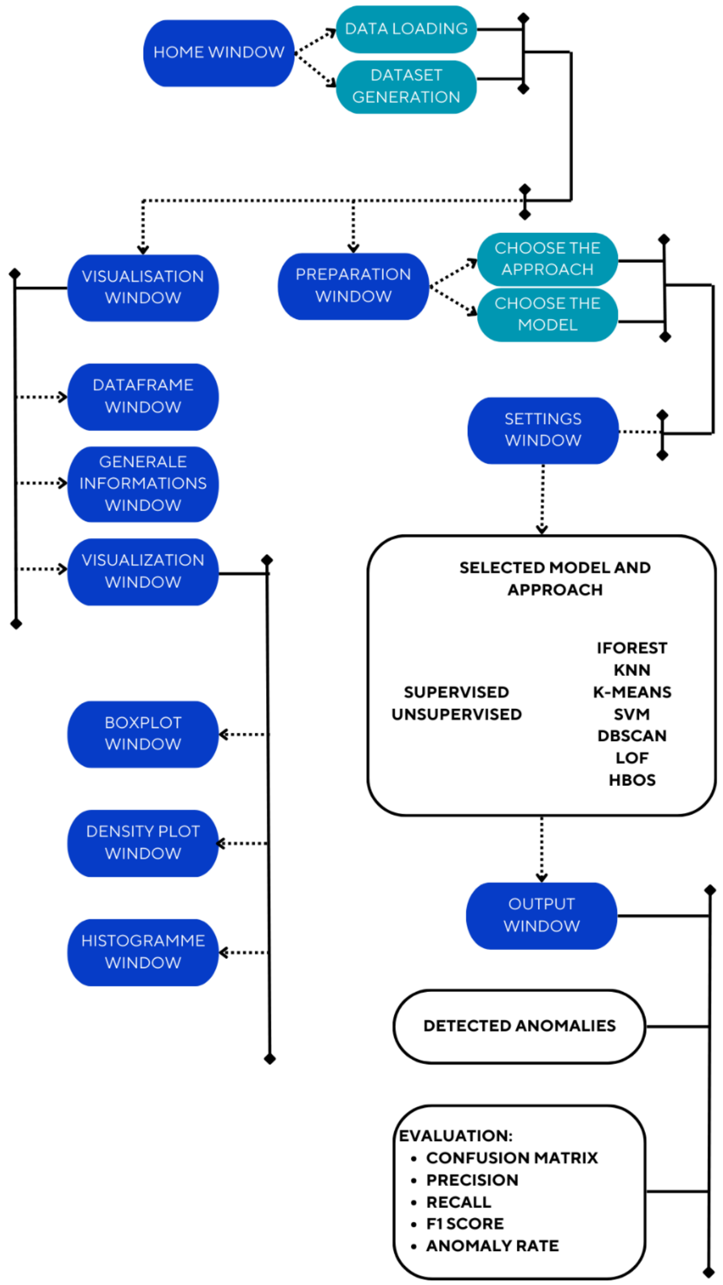


Figure 3: Architecture of our anomaly detection application

3.4. System Components and Functionality

Our application is equipped with a range of features designed to enhance network security through various anomaly detection techniques. This section outlines the key components and functionalities that enable effective analysis, dataset generation, and anomaly detection within the network security domain.

3.4.1. Data Loading

Cybersecurity datasets encompasses diverse types of data crucial for detecting and mitigating security threats in digital environments. These datasets include network traffic data, system logs, user behavior records, security alerts, malware samples, web server logs, DNS activities, authentication details, and threat intelligence feeds [28].

Our application allows the use of existing datasets from network logs and also provides the functionality to generate custom datasets. These custom datasets can be specifically tailored to meet the needs of the network security domain.

3.4.2. Data Cleaning and Preprocessing

Data in databases can contain various types of errors such as typographical errors, missing information, noisy data, or inconsistent data. The erroneous part of the processed data can be replaced, modified, or deleted. The cleaning process identifies erroneous data and either automatically corrects them using a computer program or presents them to a human for manual modifications.

Figure 4: Data Processing Workflow

3.4.3 Data Visualization and Exploration

Data Visualization is defined as the visual and interactive exploration of data. For anomaly detection, visualizations help to quickly and easily detect outlier points that deviate from normal behavior even when dealing with large volumes of data.

Our application allows users to manipulate data using traditional methods such as adding, deleting, updating, searching, and editing. Additionally, it provides charting capabilities to visualize data and detect anomalies. The following visualization charts are present in the application:

❖ **Box Plot**

A box plot (box and whiskers plot) is a simple chart composed of a rectangle with two lines extending from it to represent certain elements of the data [18].

- The central value of the chart is the **median**
- The edges of the rectangle represent the **quartiles**
- The ends of the "whiskers" are calculated using **1.5 times the interquartile range** (the distance between the 1st and 3rd quartiles).

Therefore, the inside of the box contains 50% of the observations. Values outside the whiskers are represented by points. If an observation is outside the whiskers, then it is considered an outlier (anomaly) [18].

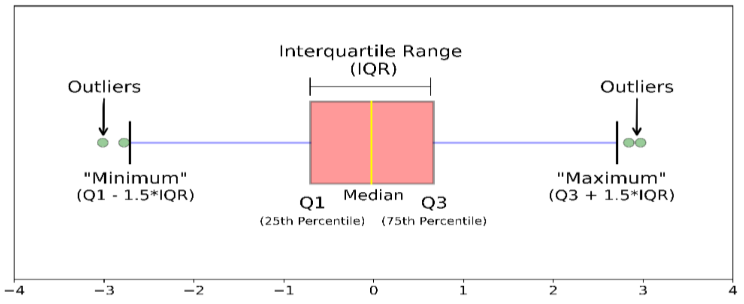


Figure 2: Box Plot

❖ Histogram

Histograms are a visual tool used to check for normality. They group data into bins and provide a count of the number of observations in each bin. By examining the shape of the bins, one can quickly determine if an attribute follows a Gaussian (normal), skewed, or even exponential distribution. This can also help in detecting possible outliers or extreme values [29].

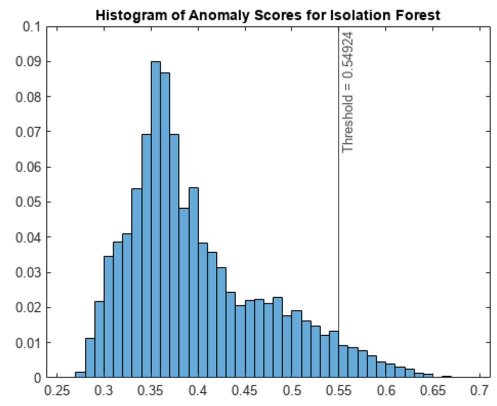


Figure 3: Histogram

❖ Density Plot

Density plots are used to observe the distribution of a variable within a dataset. They plot the values of a selected column as evenly distributed distributions. Peaks in a density plot indicate where values are concentrated within the range [30].

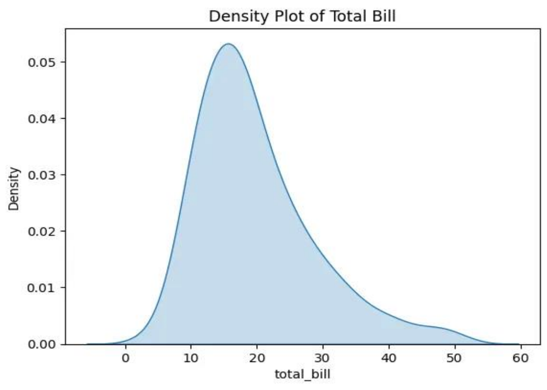


Figure 4: Density plot

3.4.4 Anomaly Detection

Our application offers a versatile platform for anomaly detection, accommodating both supervised and unsupervised approaches. Leveraging supervised methods, it harnesses labeled data to train models and detect anomalies with precision. Simultaneously, it empowers users with unsupervised techniques, allowing for anomaly detection in unlabeled datasets, offering flexibility and adaptability in cybersecurity analysis.

3.4.4.1. Machine Learning Approach for Anomaly Detection

❖ **Supervised Approach**

Supervised learning uses labeled data to train a model to predict outcomes based on input features. The model learns patterns by minimizing the error between its predictions and the known target values during training. The steps of this approach for our study are illustrated in the following algorithm:

Algorithm 1: Supervised Approach

```
      Input: «  Cybersecurity datasets  »
      Output: Detected anomalies
      Begin
1         Load cybersecurity dataset.
2         Split the dataset onto training set
          and test set.
3         Train the model on the training set.
4         Test the model to the test data.
5         Evaluate the model.
      End
```

❖ **Unsupervised Approach**

Unsupervised learning involves learning without a supervisor. It involves extracting classes or groups of individuals with common characteristics. The steps of this approach for our study are illustrated in the following algorithm:

Algorithm 2: Unsupervised Approach

```
      Input: Cybersecurity dataset
      Output : Detected Anomalies
      Begin
1         | Import the cybersecurity dataset.
2         | Apply an unsupervised anomaly detection model to
          | the dataset.
3         | Evaluate the model.
      End
```

3.4.4.2. Integrated Anomaly Detection Techniques

Our application incorporates a diverse array of advanced anomaly detection techniques from diverse anomaly detection approaches, offering users multiple options for analyzing cybersecurity data. The integrated methods include:

- ✓ Histogram-based Outlier Score (HBOS)
- ✓ K-Nearest Neighbors (KNN)
- ✓ K-Means Clustering (Kmeans)
- ✓ Local Outlier Factor (LOF)
- ✓ Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
- ✓ One-Class Support Vector Machine (One Class SVM)
- ✓ Isolation Forest (IForest)

Each technique provides unique strengths and capabilities for identifying anomalies in complex datasets and can be implemented in a supervised or unsupervised setting.

The pseudocode for statistical method HBOS is illustrated in Algorithm 3 [31].

```
Algorithm 3: HBOS(data)
    Input: data - dataset
    Output: scores - anomaly scores for each data point

    Initialize histograms for each feature
    For each feature in data:
        Create a histogram
        Calculate the density for each bin

    For each data point in data:
        Initialize score = 1
        For each feature value in the data point:
            Find the corresponding bin in the histogram
            Multiply score by the density of the bin

    Return scores
```

Pseudocodes for classification method KNN and clustering method KMeans are illustrated in Algorithm 4 and Algorithm 5 respectively:

```
Algorithm 4: KNN(data, k)
    Input: data - dataset, k - number of
    neighbors
    Output: scores - anomaly scores for
    each data point

    Initialize distance matrix

    For each data point i in data:
        For each data point j in data:
            Calculate distance between i and j
            Store distance in distance matrix
    For each data point i in data:
        Sort distances to all other points
        Select k nearest distances
        Calculate score as the average
        distance to k nearest neighbors

    Return scores
```

```
Algorithm 5: KMeans(data, k)
    Input: data - dataset, k - number of
    clusters
    Output: scores - anomaly scores for each
    data point

    Initialize k centroids randomly
    Repeat until convergence:
        Assign each data point to the nearest
        centroid
        Update centroids as the mean of
        assigned points

    For each data point in data:
        Find the distance to the nearest
        centroid
        Set score as the distance

    Return scores
```

Pseudocodes for clustering methods LOF and DBSCAN are illustrated in Algorithm 6 and Algorithm 7 respectively:

```
Algorithm 6: LOF(data, k)
  Input: data - dataset, k - number of neighbors
  Output: scores - anomaly scores for each data point

  Initialize distance matrix and reachability distances

  For each data point i in data:
    For each data point j in data:
      Calculate distance between i and j
      Store distance in distance matrix

  For each data point i in data:
    Sort distances to all other points
    Select k nearest distances
    Calculate reachability distance for k nearest neighbors

  For each data point i in data:
    Calculate local reachability density
    Calculate LOF score as the average ratio of local reachability densities

  Return scores
```

```
Algorithm 7: DBSCAN(data, eps, minPts)
  Input: data - dataset, eps - radius for neighborhood, minPts - minimum number of points to form a cluster
  Output: labels - cluster labels for each data point

  Initialize cluster label = 0
  Initialize labels for each point as unvisited

  For each data point i in data:
    If i is not visited:
    Mark i as visited
    Retrieve neighbors within eps radius
      If neighbors < minPts:
        Label i as noise
      Else:
        Increment cluster label
        Expand cluster from i with neighbors

  Return labels
```

Pseudocodes for One-Class SVM and IForests are illustrated in Algorithm 8 and Algorithm 9 respectively:

```
Algorithm 8: OneClassSVM(data, nu)
  Input: data - dataset, nu - parameter for outlier fraction
  Output: labels - anomaly labels for each data point

  Train One-Class SVM model with data and nu
  For each data point in data:
    Predict label using the trained model

  Return labels
```

```
Algorithm 9: IForest(data, nTrees, sampleSize)
  Input: data - dataset, nTrees - number of trees, sampleSize - subsample size for each tree
  Output: scores - anomaly scores for each data point

  Initialize forest of isolation trees
  For each tree in nTrees:
    Sample data points without replacement (sampleSize)
    Build isolation tree with sampled data

  For each data point in data:
    Calculate average path length from all trees
    Convert average path length to anomaly score

  Return scores
```

3.4.5 Performance Evaluation

Our application includes a comprehensive suite of performance evaluation metrics to assess the effectiveness of anomaly detection techniques. These metrics provide detailed insights into the accuracy and reliability of the models. The implemented metrics include the confusion matrix, accuracy, recall, precision, and F1 score, allowing for a thorough evaluation of anomaly detection performance.

❖ **Confusion matrix**

A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning algorithm. Each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class (or vice versa) [34]. The confusion matrix is made of the following values:

- ✓ **TP (True Positive)**: The number of correctly predicted positive instances. These are the cases where the model correctly identifies an anomaly.
- ✓ **TN (True Negative)**: The number of correctly predicted negative instances. These are the cases where the model correctly identifies normal instances (non-anomalies).

- ✓ **FP (False Positive)**: The number of incorrect positive predictions. These occur when the model incorrectly identifies a normal instance as an anomaly (also known as a Type I error).

- ✓ **FN (False Negative)**: The number of incorrect negative predictions. These occur when the model fails to identify an actual anomaly and incorrectly classifies it as normal (also known as a Type II error).

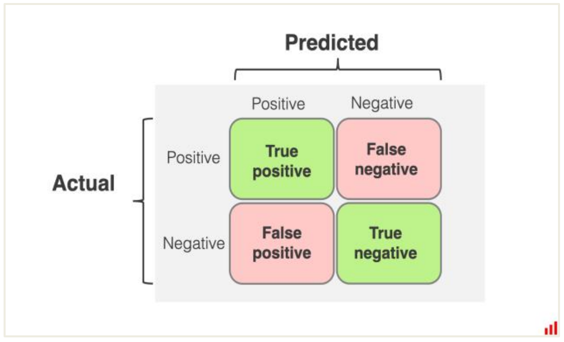


Figure 5: Confusion matrix

These values form the basis for calculating various performance metrics, such as accuracy, precision, recall, and F1 score. These metrics are defined as follows:

❖ **Precision**

Precision is defined as the ratio of true positive results to the total number of positive results predicted by the model. In other words, precision measures the accuracy of the positive predictions [35]. It is given by the formula:

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

High precision indicates that the model produces a large number of correct positive predictions compared to the number of incorrect positive predictions.

❖ **Recall**

Recall is defined as the ratio of true positive results to the total number of actual positive instances. In other words, recall measures the ability of a model to identify all relevant instances within a dataset [35]. Mathematically, recall is given by the formula:

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

High recall indicates that the model correctly identifies a large proportion of the actual positive cases.

❖ **F1 Score**

The **F1 Score** is the harmonic mean of precision and recall. It provides a single measure that balances both the precision and the recall of the model. The F1 score is particularly useful when the distribution of classes is imbalanced [35]. Mathematically, it is given by the formula:

$$F1Score = \frac{2 * Precision * recall}{Precision + recall}$$

The F1 score ranges from 0 to 1, with 1 indicating perfect precision and recall, and 0 indicating the worst performance.

3.5. Conclusion

The detailed exploration of our anomaly detection application reveals its potential to enhance network security through advanced detection techniques and performance evaluation metrics. This practical demonstration exemplifies the real-world application of anomaly detection methodologies, offering a valuable tool for cybersecurity professionals.

Chapter 04: Implementing and Testing

4.1. Introduction

In this chapter, we provide a detailed account of the implementation of our developed anomaly detection application for network security. We explore the programming environments utilized, showcase the user interfaces of the application, and provide a comprehensive demonstration of its functionalities. Additionally, we present a case study in network security, applying our developed application to real-world scenarios. Through this case study, we aim to evaluate and compare the performance of the anomaly detection techniques implemented in the application.

4.2. Programming Environment and Tools

4.2.1. Hardware

The hardware environment in which our anomaly detection system was implemented is characterized by:

Table 1: Hardware Specifications

Workstation	**Characteristics**
PC	Laptop
Operating System	Windows 11
RAM	8.00 GO
Processor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz 2.50 GHz
System type	64-bit operating system, x64 processor

4.2.2. Programming Language

For the implementation of the anomaly detection application for network security, we have chosen the Python language version 3.11.5.

Python is a general-purpose, high-level, interpreted programming language that is simple to learn. It is used for web development, artificial intelligence, machine learning, operating systems, mobile application development, video games, and much more. It has an ordered structure and straightforward grammar. Because of this, it's a great option for a variety of projects, from straightforward online apps to full operating systems [36].

4.2.3. Programming Environment

The code editor Spyder (Scientific Python Development Environment) was used to create this project.

Spyder is a cross-platform, open-source Python development environment that works with GNU/Linux, Mac OS, and Windows. It incorporates a number of libraries, including IPython, Matplotlib, and NumPy. Though Spyder is focused on scientific computing, it may be used as an environment for developing any kind of application [37].

4.2.4. Employed Python Packages

- **Pandas:** a Python data manipulation library. Another wordplay on the phrase "Python Data Analysis" appears in its name [38].
- **NumPy:** A Python numerical computing toolkit that supports huge, multi-dimensional arrays and matrices and offers a number of mathematical operations that may be performed on them [39].
- **Matplotlib:** a Python package for making animated, interactive, and static visualizations. It has an interface similar to MATLAB and is frequently used to create charts and plots [40].
- **Sklearn:** A robust Python machine learning framework that offers easy-to-use capabilities for data mining and analysis. Numerous techniques for clustering, regression, classification, dimensionality reduction, and other tasks are included [42].

- **PyOD:** A comprehensive and scalable Python framework for identifying outliers in multivariate data is the Python Outlier Detection (PyOD) toolbox. It has around forty algorithms for detecting anomalies [44].
- **CustomTkinter:** CustomTkinter is a personalized or enhanced version of the Tkinter library in Python, offering additional functionalities or optimizations tailored to specific requirements or preferences [45].
- **PandasTable:** PandasTable is a Python library that offers a graphical interface for interacting with Pandas DataFrames, facilitating data exploration and manipulation tasks within a visual environment [46].

These packages together provide a robust foundation for implementing anomaly detection algorithms and analyzing data in Python.

4.3. Anomaly Detection Application for Network Security

Our application is designed to facilitate the detection of anomalies within network security datasets. It leverages advanced anomaly detection algorithms to identify irregular patterns that may indicate errors, fraud, or other significant events. The user-friendly interface ensures a smooth experience, allowing users to easily load data, configure settings, and visualize results.

In the following, we will illustrate the application with figures and explain in detail each of its functionalities.

4.3.1. Home Window

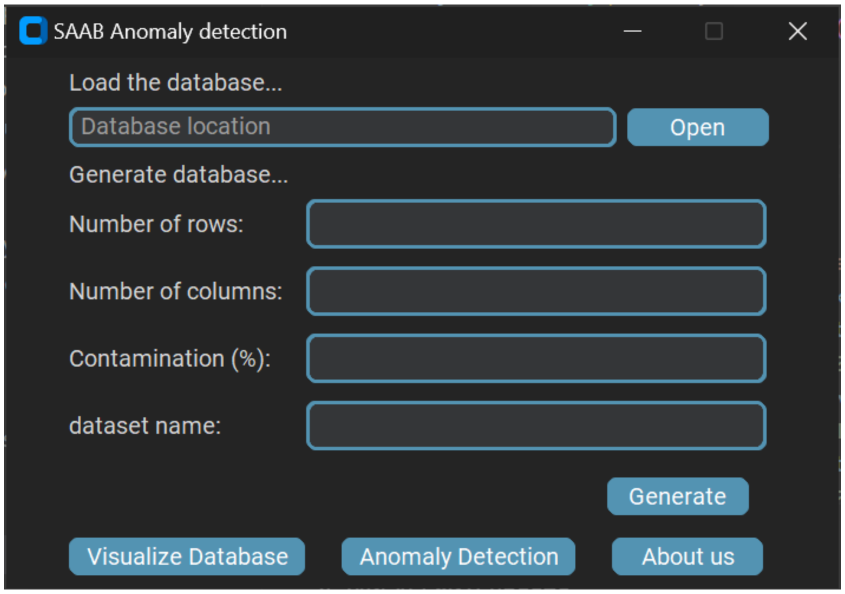


Figure 6: Home window

The Home window defines a GUI using Tkinter for the anomaly detection application. It allows users to load existing databases, generate new ones, visualize loaded databases, and perform anomaly detection. Users can specify parameters for database generation and initiate actions like loading databases, generating databases, visualizing them, and detecting anomalies and about us button.

4.3.2. Data Visualization Window

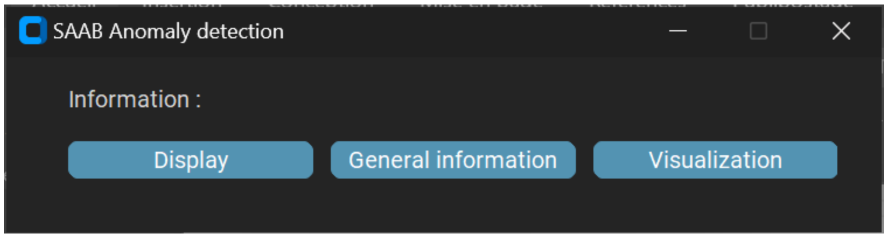
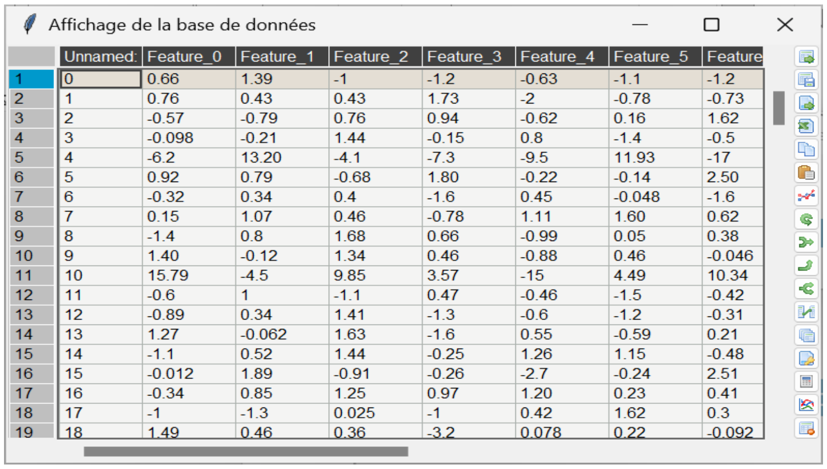


Figure 7: Visualization window

This window serves as a comprehensive tool for exploring datasets and visually identifying anomalies. It offers three main functionalities:

- ✓ **DataFrame Display Button (Dispaly)**: This button allows users to view the dataset in a tabular format, providing a detailed look at the individual data points and their corresponding features. Users can scroll through the data to examine specific entries and gain insights into the dataset's structure.

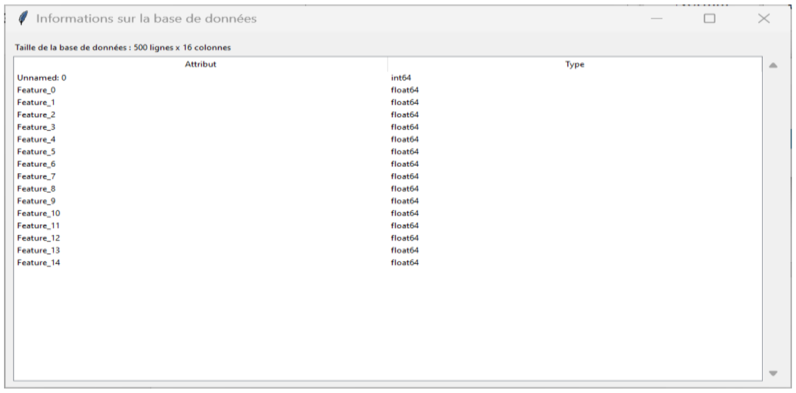


Affichage de la base de données

	Unnamed:	Feature_0	Feature_1	Feature_2	Feature_3	Feature_4	Feature_5	Feature
1	0	0.66	1.39	-1	-1.2	-0.63	-1.1	-1.2
2	1	0.76	0.43	0.43	1.73	-2	-0.78	-0.73
3	2	-0.57	-0.79	0.76	0.94	-0.62	0.16	1.62
4	3	-0.098	-0.21	1.44	-0.15	0.8	-1.4	-0.5
5	4	-6.2	13.20	-4.1	-7.3	-9.5	11.93	-17
6	5	0.92	0.79	-0.68	1.80	-0.22	-0.14	2.50
7	6	-0.32	0.34	0.4	-1.6	0.45	-0.048	-1.6
8	7	0.15	1.07	0.46	-0.78	1.11	1.60	0.62
9	8	-1.4	0.8	1.68	0.66	-0.99	0.05	0.38
10	9	1.40	-0.12	1.34	0.46	-0.88	0.46	-0.046
11	10	15.79	-4.5	9.85	3.57	-15	4.49	10.34
12	11	-0.6	1	-1.1	0.47	-0.46	-1.5	-0.42
13	12	-0.89	0.34	1.41	-1.3	-0.6	-1.2	-0.31
14	13	1.27	-0.062	1.63	-1.6	0.55	-0.59	0.21
15	14	-1.1	0.52	1.44	-0.25	1.26	1.15	-0.48
16	15	-0.012	1.89	-0.91	-0.26	-2.7	-0.24	2.51
17	16	-0.34	0.85	1.25	0.97	1.20	0.23	0.41
18	17	-1	-1.3	0.025	-1	0.42	1.62	0.3
19	18	1.49	0.46	0.36	-3.2	0.078	0.22	-0.092

Figure 8: Dataset display window

- ✓ **Dataset Information Button (General Information)**: Clicking on this button provides users with general information about the dataset.



Informations sur la base de données

Taille de la base de données : 500 lignes x 16 colonnes

Attribut	Type
Unnamed: 0	int64
Feature_0	float64
Feature_1	float64
Feature_2	float64
Feature_3	float64
Feature_4	float64
Feature_5	float64
Feature_6	float64
Feature_7	float64
Feature_8	float64
Feature_9	float64
Feature_10	float64
Feature_11	float64
Feature_12	float64
Feature_13	float64
Feature_14	float64

Figure 9: General information window

- ✓ **Data Visualization Button (Visualization)**: This button opens up a suite of visualization tools for exploring the data graphically. Users can choose from various visualization types, including box plots, histograms, and density plots. Additionally, they have the option to select a specific feature (column) from the dataset to visualize. This feature is particularly useful for identifying anomalies visually. Users can spot unusual patterns, outliers, or distributions that deviate from the norm, indicating potential anomalies within the dataset.

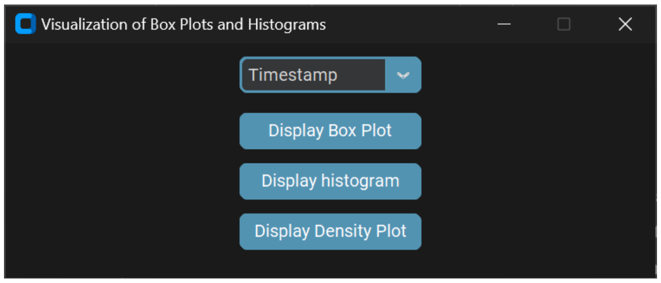


Figure 10: Data Visualization plots window

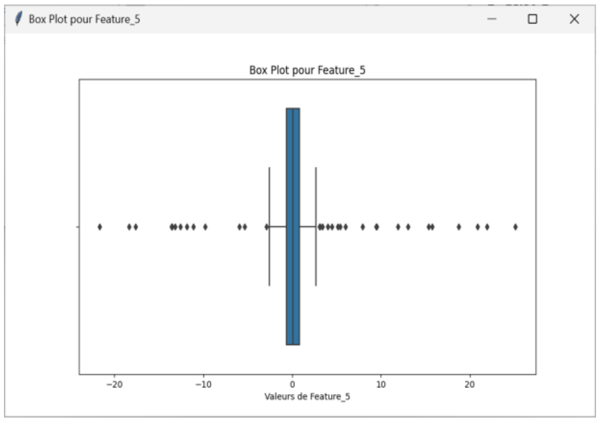


Figure 11: Box plot window

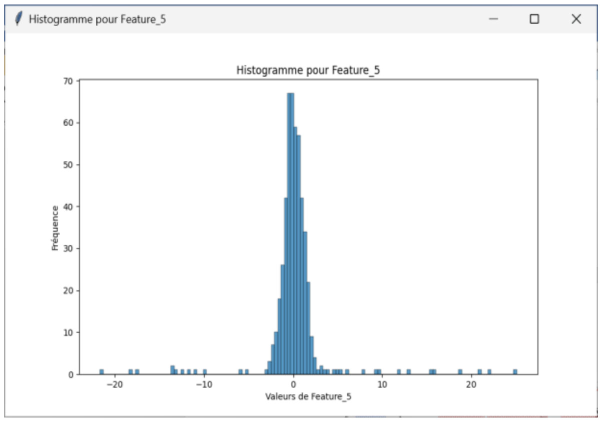


Figure 12: Histogram window

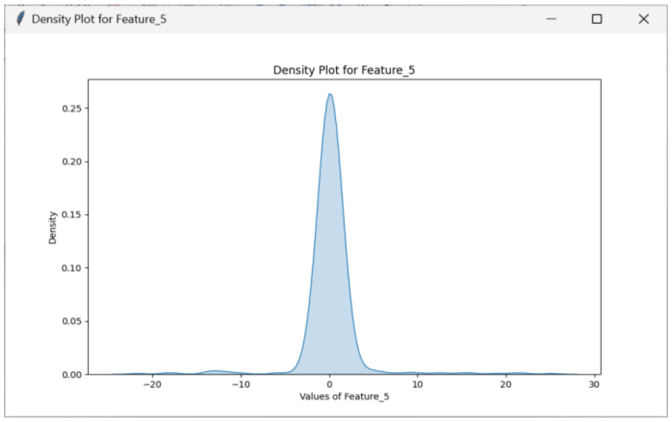
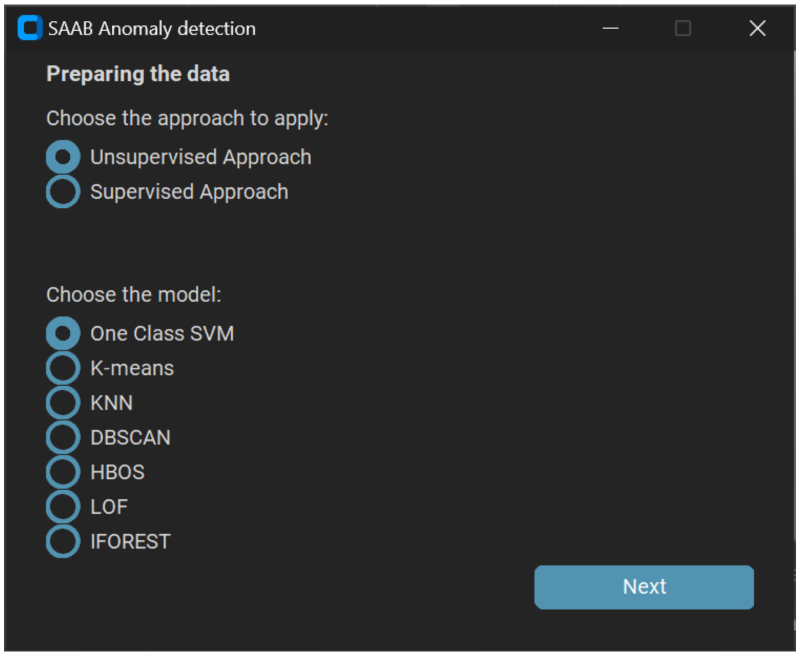


Figure 13: Density plot window

4.3.3. Anomaly Detection Window


SAAB Anomaly detection

Preparing the data

Choose the approach to apply:

[✓ Unsupervised Approach]
Supervised Approach

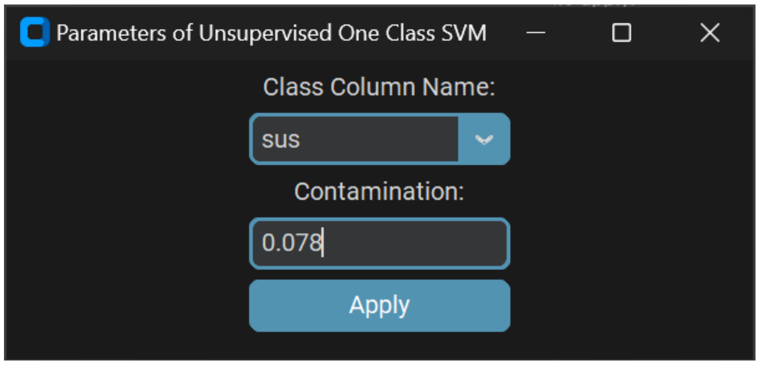
Choose the model:

[✓ One Class SVM]
K-means
KNN
DBSCAN
HBOS
LOF
IFOREST

Figure 14: Anomaly detection window

This window serves as a central hub for configuring the approach and method for anomaly detection. It comprises two selection groups along with a "Next" button for advancing to the parameter configuration window. Here's a breakdown of its components:

- ✓ **Approach Selection Group**: This group allows users to choose the approach or methodology for anomaly detection (supervised or unsupervised).
- ✓ **Method Selection Group**: Within the chosen approach, users can further refine their selection by specifying the method or algorithm to be used for anomaly detection.
- ✓ **Next Button**: Clicking the "Next" button submits the selected approach and method configurations. It serves as a trigger to proceed to the next stage of the anomaly detection process, opening the parameter window (see Figure 18) specific to the chosen method. This window allows users to fine-tune the parameters associated with the selected algorithm, ensuring optimal performance for anomaly detection.


Parameters of Unsupervised One Class SVM

Class Column Name:

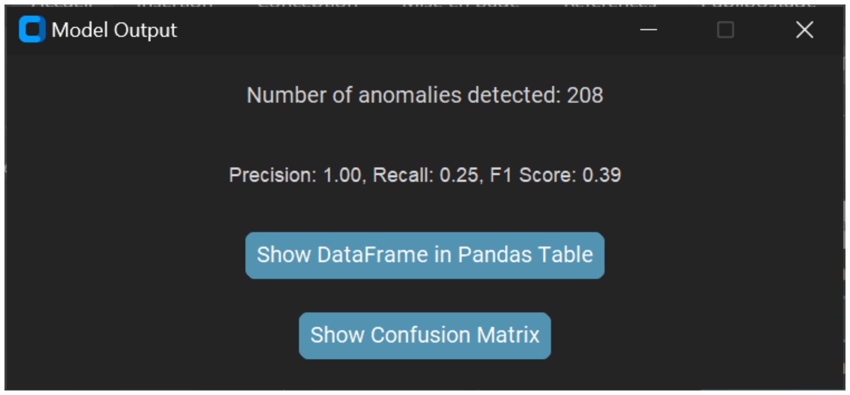
sus

Contamination:

0.078

Figure 15: Settings window

After selecting the approach and method, and configuring the parameters, users proceed to the results window (see Figure 19) upon clicking "Apply" This window provides a summary of the anomaly detection outcomes and options for detailed analysis. It includes:


Model Output

Number of anomalies detected: 208

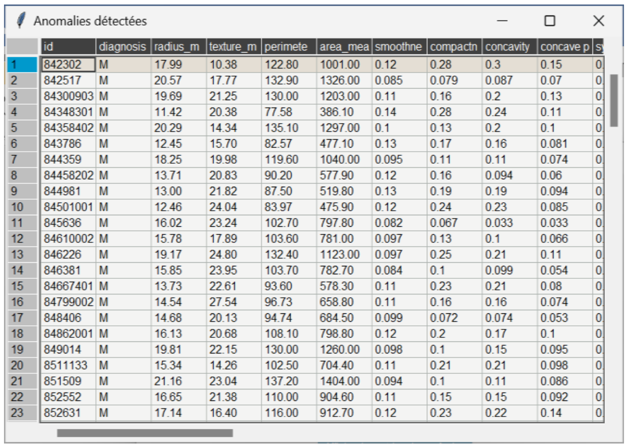
Precision: 1.00, Recall: 0.25, F1 Score: 0.39

Figure 16: Results window

- ✓ **Number of Detected Anomalies**: Displays the total count of identified anomalies.
- ✓ **Evaluation Report**: Shows key metrics like F1 score and recall support for assessing method performance.

The results window include also two other action buttons:

- ✓ **Show Anomalies (Show DataFrame in Pandas Table)**: Opens a window displaying the detected anomalies in a DataFrame format (see Figure 20).
- ✓ **Show Confusion Matrix (Show Confusion Matrix)**: Opens a window showing the confusion matrix for detailed performance analysis (see Figure 21).



Anomalies détectées

	id	diagnosis	radius_m	texture_m	perimete	area_mea	smoothne	compactn	concavity	concave p	sy
1	842302	M	17.99	10.38	122.80	1001.00	0.12	0.28	0.3	0.15	0
2	842517	M	20.57	17.77	132.90	1326.00	0.085	0.079	0.087	0.07	0
3	84300903	M	19.69	21.25	130.00	1203.00	0.11	0.16	0.2	0.13	0
4	84348301	M	11.42	20.38	77.58	386.10	0.14	0.28	0.24	0.11	0
5	84358402	M	20.29	14.34	135.10	1297.00	0.1	0.13	0.2	0.1	0
6	843786	M	12.45	15.70	82.57	477.10	0.13	0.17	0.16	0.081	0
7	844359	M	18.25	19.98	119.60	1040.00	0.095	0.11	0.11	0.074	0
8	84458202	M	13.71	20.83	90.20	577.90	0.12	0.16	0.094	0.06	0
9	844981	M	13.00	21.82	87.50	519.80	0.13	0.19	0.19	0.094	0
10	84501001	M	12.46	24.04	83.97	475.90	0.12	0.24	0.23	0.085	0
11	845636	M	16.02	23.24	102.70	797.80	0.082	0.067	0.033	0.033	0
12	84610002	M	15.78	17.89	103.60	781.00	0.097	0.13	0.1	0.066	0
13	846226	M	19.17	24.80	132.40	1123.00	0.097	0.25	0.21	0.11	0
14	846381	M	15.85	23.95	103.70	782.70	0.084	0.1	0.099	0.054	0
15	84667401	M	13.73	22.61	93.60	578.30	0.11	0.23	0.21	0.08	0
16	84799002	M	14.54	27.54	96.73	658.80	0.11	0.16	0.16	0.074	0
17	848406	M	14.68	20.13	94.74	684.50	0.099	0.072	0.074	0.053	0
18	84862001	M	16.13	20.68	108.10	798.80	0.12	0.2	0.17	0.1	0
19	849014	M	19.81	22.15	130.00	1260.00	0.098	0.1	0.15	0.095	0
20	8511133	M	15.34	14.26	102.50	704.40	0.11	0.21	0.21	0.098	0
21	851509	M	21.16	23.04	137.20	1404.00	0.094	0.1	0.11	0.086	0
22	852552	M	16.65	21.38	110.00	904.60	0.11	0.15	0.15	0.092	0
23	852631	M	17.14	16.40	116.00	912.70	0.12	0.23	0.22	0.14	0

Figure 17: Detected anomalies window

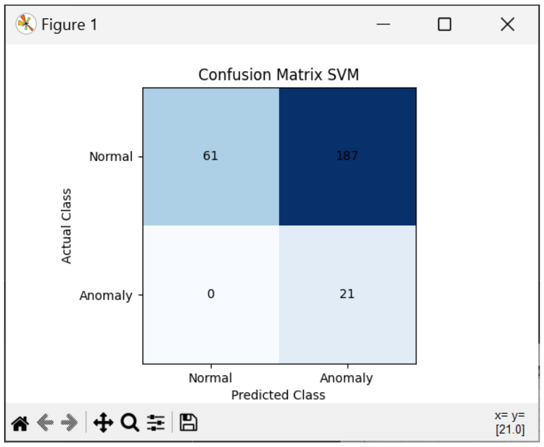


Figure 18: Confusion matrix window

4.3.4. «About Us» Window

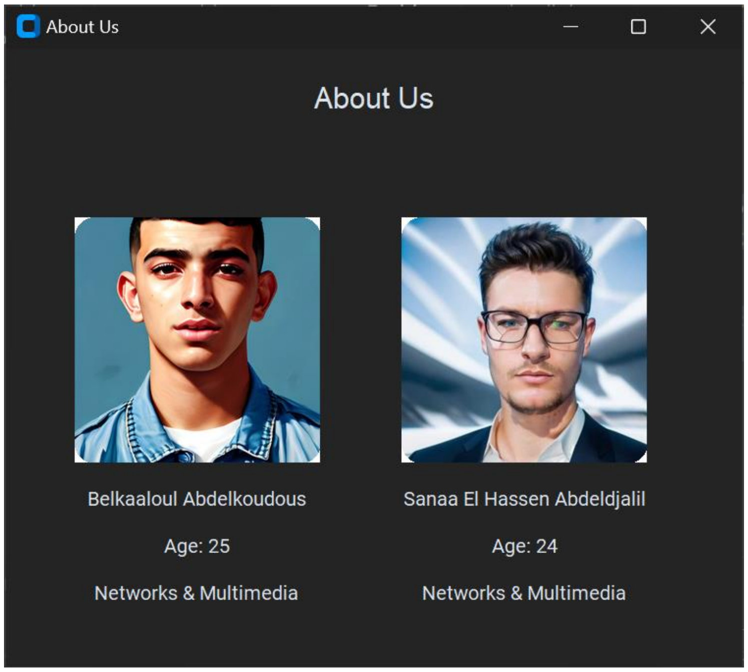


Figure 19: About Us window

About Us page. Here, we introduce team members who have developed the application.

4.4. Case Study: Anomaly Detection for Network Security

In this section, we present a case study using our anomaly detection application, focusing on identifying anomalies within the BETH network security dataset. This case study specifically targets malicious activities and potential threats within network traffic. This practical example demonstrates the effectiveness and relevance of our application in a real-world network security context.

4.4.1. Dataset Description

BETH is a public cybersecurity dataset for anomaly detection and out-of-distribution analysis. Featuring real "anomalies" collected using a novel tracking system, the dataset comprises over eight million data points tracking 23 hosts. Each host's activity includes benign behavior and, at most, a single attack, allowing for cleaner behavioral analysis. In addition to being one of the most modern and extensive cybersecurity datasets available, BETH facilitates the development of anomaly detection algorithms on heterogeneously structured real-world data, with clear downstream applications. This dataset contains several attributes [28] .

- ✓ **Timestamp:** The specific date and time when the data point was recorded, indicating when the network activity occurred.
- ✓ **SourceIP:** The IP address of the source host that initiated the network communication or DNS query.
- ✓ **DestinationIP:** The IP address of the destination host or server that received the network communication or DNS query.
- ✓ **DnsQuery:** The domain name or address that was queried in the DNS request.
- ✓ **DnsAnswer:** The response provided by the DNS server to the query, which may include the IP address or other resource records.
- ✓ **DnsAnswerTTL:** The Time-To-Live (TTL) value for the DNS answer, specifying how long the response should be cached before it expires.

- ✓ **DnsQueryNames:** A list of domain names involved in the DNS query, often including aliases or canonical names.
- ✓ **DnsQueryClass:** The class of the DNS query, typically specifying the protocol family (e.g., IN for Internet).
- ✓ **DnsQueryType:** The type of DNS query, indicating the type of resource record being requested (e.g., A for address, MX for mail exchange).
- ✓ **NumberOfAnswers:** The number of answers returned in the DNS response, indicating how many resource records were included.
- ✓ **DnsResponseCode:** The code returned by the DNS server indicating the result of the query (e.g., NOERROR for successful queries, NXDOMAIN for non-existent domains).
- ✓ **DnsOpCode:** The operational code of the DNS query, indicating the type of query (e.g., QUERY for standard queries, UPDATE for updates to DNS records).
- ✓ **SensorId:** An identifier for the sensor or device that collected the data point, useful for tracing the source of the data.
- ✓ **sus:** A binary indicator of whether the data point is suspicious (1 for anomaly, 0 for normal), based on heuristic rules or initial analysis.
- ✓ **evil:** A binary indicator of whether the data point is confirmed malicious (1 for anomaly, 0 for normal), based on more stringent validation or external threat intelligence.

The other characteristics of the dataset are presented in the table below :

Table 2: The characteristics of BETH dataset

Dataset	**BETH Cybersecurity Dataset**
Number of rows	8004918

Number of columns	15
Type of data	Float, int, String
DataFrame size	81.1GB

4.4.2. Data Visualization

❖ Box Plot

The box plot in our application serves to visually detect outliers. Data points outside the minimum and maximum values are considered anomalies. For instance, Figure 22 displays box plots for two of our dataset features: "SourceIP" and "DestinationIP". These plots clearly indicate the presence of anomalies in both columns.

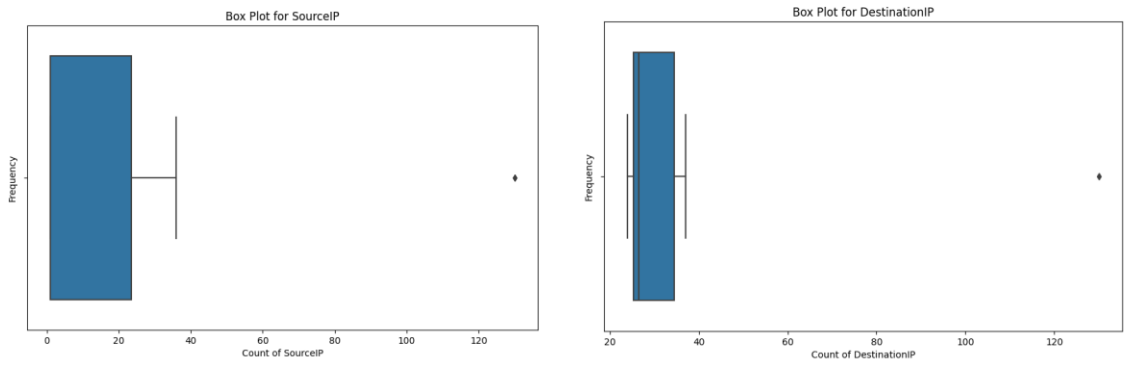


Figure 20: Box plot of SourceIP and DestinationIP

❖ Histogram and density plot

Histograms and density plots serve the same purpose. Figure 23 displays the histograms and density plots for the same two columns in our database: "SourceIP" and "DestinationIP". The abnormal data in these columns is highlighted by their separation from the normal data.

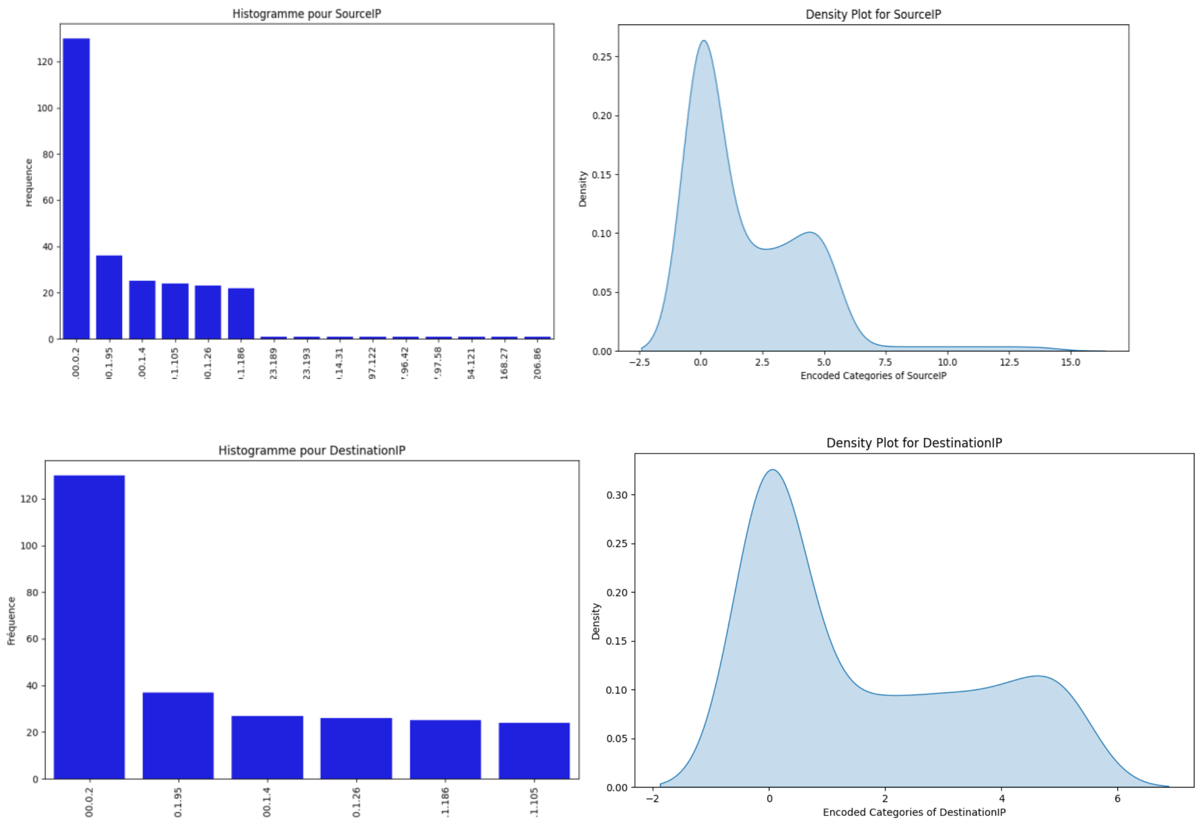


Figure 21: Histograms and density plots of SourceIP and DestinationIP

4.4.3. Data Cleaning and Preprocessing

For the anomaly detection task, the "BETH" database does not require many modifications. The two preprocessing steps we applied are replacing «Null» values with numerical values, which has no effect on anomaly detection, and converting non-numeric data into numeric data.

4.4.4. Implementing Anomaly Detection Techniques

The table below represents the parameters of the different methods we implemented for anomaly detection:

Table 3: The chosen parameters for anomal detection methods

Algorithm	Parameters	Description	Default Values
IFOREST	Contamination	The proportion of outliers in the dataset.	In our study, we have set the contamination to be 0.078
HBOS	Random_state	Used to set the seed value for the random number generator	Random_state = 'NONE'
DBSCAN	Eps	This parameter specifies the maximum distance between two samples for them to be considered as part of the same neighborhood	Eps = 0.5
	min_samples	This parameter determines the minimum number of samples required to form a dense region	min_samples = 5
K-MEANS	n_clusters	This parameter specifies the number of clusters to form	n_clusters = 8
KNN **LOF** **SVM**	Contamination	The proportion of outliers in the dataset.	In our study, we have set the contamination to be 0.078
LOF	n_neighbors	This parameter specifies the number of neighbors to consider for each sample when calculating its local density	n_neighbors = 20

4.4.5. Results

❖ **Displaying performance using the application : Example of the confusion matrix of IFOREST model**

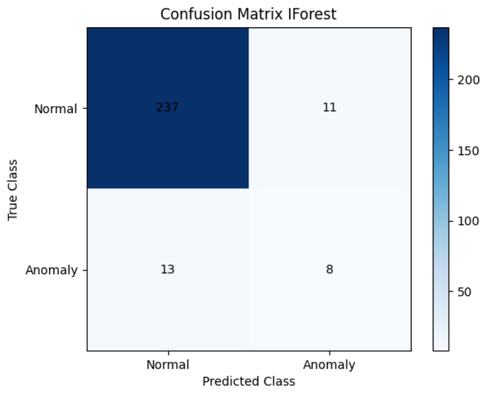


Figure 22 Confusion matrix for IFOREST following unsupervised approach

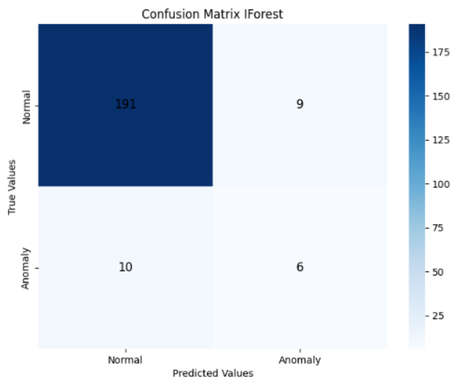


Figure 23 Confusion matrix for IFOREST following supervised approach

Taking the confusion matrix of the IFOREST model following unsupervised approach as an example. This metric clearly shows that:

- **Total records:** 269
- **KNN model predictions:**
    - Normal Class: 250 detected
    - Anomaly Class: 19 detected

- **Correct predictions:**
    - True Positives (Normal Class): 237
    - True Negatives (Anomaly Class): 8

- **Classification errors:**
    - **False Positives**: 11 (Anomaly Class predicted as Normal Class)
    - **False Negatives**: 13 (Normal Class predicted as Anomaly Class)

To further clarify, here are the terms used in the confusion matrix:

- **True Positives (TP):** The model correctly predicted the positive class.
- **False Positives (FP):** The model incorrectly predicted the positive class when it is actually a negative class.
- **True Negatives (TN):** The model correctly predicted the negative class.
- **False Negatives (FN):** The model incorrectly predicted the negative class when it is actually a positive class.

Taking the confusion matrix of the IFOREST model following supervised approach as an example. This metric clearly shows that:

- **Total records:** 206
- **KNN model predictions:**
    - Normal Class: 201 detected
    - Anomaly Class: 15 detected
- **Correct predictions:**
    - True Positives (Normal Class): 191
    - True Negatives (Anomaly Class): 6

- **Classification errors:**
    - **False Positives**: 9 (Anomaly Class predicted as Normal Class)
    - **False Negatives**: 10 (Normal Class predicted as Anomaly Class)

❖ **Summary of the results of all anomaly detection models**

To specify which algorithm is the most performant for outlier detection, we employed the following evaluation metrics: Precision, Recall, and F1 Score.

The table below summarizes all the results obtained using both supervised and unsupervised learning approaches.

Table 4: Performance of anomaly detection models

Method	**Supervised**			**Unsupervised**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
IFOREST	0.95	<u>0.95</u>	**0.95**	0.95	<u>0.96</u>	**0.95**
HBOS	0.10	0.38	0.16	0.95	<u>0.96</u>	**0.95**
KNN	0.40	0.38	0.39	0.95	<u>0.96</u>	**0.95**
K-MEANS	0.75	0.19	0.30	0.93	0.73	0.82
LOF	0.95	<u>0.95</u>	**0.95**	0.95	<u>0.96</u>	**0.95**
SVM	0.94	<u>0.96</u>	**0.95**	1.00	0.25	0.39
DBSCAN	0.91	0.79	0.84	0.95	<u>0.96</u>	**0.95**

4.4.6 Discussion

Isolation Forest achieves high precision, recall, and F1-Score in both supervised and unsupervised settings, indicating its robustness and effectiveness in anomaly detection. It consistently outperforms other models across all metrics, making it a promising choice for anomaly detection tasks. This can be justified for several reasons. First, its intuitive concept leverages the idea that anomalies are "few and different" from normal data points, allowing it to isolate them quickly. Second, randomized partitioning of features and thresholds helps uncover diverse outliers. Third, IForest is scalable, handling large datasets efficiently. Fourth, its robustness ensures good performance across various contexts. Finally, its simplicity contributes to its effectiveness, making it a reliable choice for anomaly detection tasks.

HBOS performs poorly in supervised settings but shows competitive performance in unsupervised settings, particularly in terms of F1-Score. Its limited effectiveness in supervised mode may be due to its reliance on histogram-based modeling, which might not capture the complexities of labeled data well.

KNN exhibits moderate performance in unsupervised settings and low performance in supervised settings, with precision, recall, and F1-Score values in the mid-range. It is notable for its simplicity and ease of implementation, but may not excel in scenarios with highly complex or non-linear data distributions.

K-MEANS demonstrates relatively weaker performance compared to other models, particularly in terms of recall and F1-Score. As expected, K-MEANS, designed primarily for clustering, struggles with anomaly detection tasks where anomalies may not form distinct clusters.

LOF shows consistently high performance across all metrics in both supervised and unsupervised settings, indicating its effectiveness in identifying anomalies. It utilizes the local density deviation of data points, making it robust and versatile for various datasets and anomaly types.

SVM performs well in terms of precision, recall, and F1-Score in both settings, but exhibits a higher error rate compared to IForest and LOF. SVM's effectiveness may vary depending on the dataset and kernel selection, but it generally provides competitive performance for anomaly detection tasks.

Lastly, DBSCAN achieves good performance in unsupervised settings but shows a slight decrease in supervised mode, particularly in precision and F1-Score. DBSCAN's ability to identify global outliers makes it effective, but its performance may vary depending on the dataset's characteristics.

In general, Isolation Forest, Local Outlier Factor, and Support Vector Machine stand out as effective options for both supervised and unsupervised anomaly detection tasks. They demonstrate strong performance in various scenarios. On the other hand, K-means and Histogram-based Outlier Score show relatively weaker performance in comparison. When selecting an algorithm, it’s crucial to take into account the specific attributes of the dataset and the desired balance between precision, recall, and computational efficiency.

4.5 Conclusion

In conclusion, the detailed implementation of our anomaly detection application highlights its effectiveness in addressing network security anomaly detection problem. Through the presented case study, we have demonstrated the application's capability to detect potential threats in real-world scenarios. The comparison of performance metrics underscores the importance of selecting appropriate anomaly detection techniques tailored to specific network security needs. Overall, our application stands as a valuable tool for enhancing cybersecurity defenses and protecting against evolving threats.

General Conclusion

In this thesis, we addressed the critical challenge of enhancing network security in the face of increasingly sophisticated cyber threats. Recognizing the limitations of traditional security measures, we focused on the development and application of an anomaly detection system tailored specifically for network security.

The primary objective of our project was to develop a robust anomaly detection application that combines supervised and unsupervised techniques, drawing from statistical, clustering, and machine learning approaches. Through systematic methodology, we meticulously designed, implemented, and tested our application, ensuring its effectiveness and versatility in analyzing both historical and synthetic cybersecurity datasets.

The case study conducted in network security served as a crucial validation of our application's capabilities. We demonstrated its effectiveness in identifying anomalies and potential threats within network traffic, highlighting its practical relevance and significance in real-world scenarios.

Looking ahead, our findings underscore the importance of continued research and innovation in anomaly detection for network security. As cyber threats continue to evolve, it is imperative to develop adaptive and robust solutions that can effectively safeguard network infrastructures.

In conclusion, this thesis makes a significant contribution to the field of cybersecurity by presenting a comprehensive anomaly detection system specifically tailored for network security. The application provides actionable insights and proactive defense mechanisms, aiming to strengthen cyber defenses and mitigate potential threats in an ever-changing digital landscape.

References

[1] Foorthuis, R. (2021). On the nature and types of anomalies: a review of deviations in data. International journal of data science and analytics, 12(4), 297-331.

[2] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.

[3] Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. Signal processing, 99, 215-249.

[4] Aggarwal, C. C. (2016). Outlier analysis second edition.

[5] Blanchard, G., Lee, G., & Scott, C. (2010). Semi-supervised novelty detection. The Journal of Machine Learning Research, 11, 2973-3009. Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. Journal of Network and Computer Applications, 60, 19-31.

[6] Blanchard, G., Lee, G., & Scott, C. (2010). Semi-supervised novelty detection. The Journal of Machine Learning Research, 11, 2973-3009.

[7] Erfani, S. M., Rajasegarar, S., Karunasekera, S., & Leckie, C. (2016). High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recognition, 58, 121-134.

[8] Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial intelligence review, 22, 85-126.

[9] Agrawal, D., Aggarwal, C. C., & Prasad, V. (2013). A framework for scalable and efficient anomaly detection. *Proceedings of the VLDB Endowment*, 6(3), 197-208.

[10] Liu, X. Y., Wu, J., & Zhou, Z. H. (2008). Exploratory undersampling for class-imbalance learning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 39(2), 539-550.

[11] Huang, C., Li, Y., Loy, C. C., & Tang, X. (2016). Learning deep representation for imbalanced classification. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5375-5384).

[12] Liao, Y., Vemuri, V. R., & Chun, S. A. (2019). Interpretability in deep learning models: A survey of methods, application, and opportunities. *Artificial Intelligence Review*, 52(1), 87-113.

[13] Cornish, Paul (ed.) (2021).The Oxford Handbook of Cyber Security, Oxford Handbooks (2021; online edn, Oxford Academic, 8 Dec. 2021), https://doi.org/10.1093/oxfordhb/9780198800682.001.0001, accessed 2 June 2024.

[13] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. AI magazine, 17(3), 37-37.

[14] Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. Statistical Science, 17(3), 235-255.

[15] Khan, S. S. (2014). A review of anomaly detection techniques in network intrusion detection system. International Journal of Computer Applications, 96(9), 13-18.

[16] Kejariwal, A., & Saltzman, J. (2015). Fast and memory efficient unsupervised anomaly detection in streaming environmental data. Environmental Modelling & Software, 72, 1-10.

[17] Xiong, B., Tao, B., & Li, G. (2019, August). Research status and trend of fault diagnosis based on deep belief network. In Journal of Physics: Conference Series (Vol. 1302, No. 2, p. 022082). IOP Publishing.

[18] Tukey, J. W. (1977). Exploratory data analysis (Vol. 2, pp. 131-160). Reading, MA: Addison-wesley.

[19] Goldstein, M., & Dengel, A. (2012). Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track, 1, 59-63.

[20] Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying

density-based local outliers. In Proceedings of the 2000 ACM SIGMOD international conference on Management of data (pp. 93-104).

[21] Liu, F. T., Ting, K. M., &amp; Zhou, Z.-H. (2008). Isolation Forest. In 2008 Eighth

IEEE International Conference on Data Mining (pp. 413-422). IEEE.

[22] Bierbrauer, D. A., Chang, A., Kritzer, W., & Bastian, N. D. (2021). Cybersecurity anomaly detection in adversarial environments. arXiv preprint arXiv:2105.06742.

[23] Jadidi, Z., Pal, S., Nayak, N., Selvakkumar, A., Chang, C. C., Beheshti, M., & Jolfaei, A. (2022, July). Security of machine learning-based anomaly detection in cyber physical systems. In 2022 International Conference on Computer Communications and Networks (ICCCN) (pp. 1-7). IEEE.

[24] Hdaib, M., Rajasegarar, S., & Pan, L. (2024). Quantum deep learning-based anomaly detection for enhanced network security. Quantum Machine Intelligence, 6(1), 26.

[25] Ma, Q., Sun, C., & Cui, B. (2021). A novel model for anomaly detection in network traffic based on support vector machine and clustering. Security and Communication Networks, 2021, 1-11.

[26] Zhao, M., Chen, J., & Li, Y. (2018, April). A review of anomaly detection techniques based on nearest neighbor. In 2018 International Conference on Computer Modeling, Simulation and Algorithm (CMSA 2018) (pp. 290-292). Atlantis Press.

[27] Saeedi Emadi, H., & Mazinani, S. M. (2018). A novel anomaly detection algorithm using DBSCAN and SVM in wireless sensor networks. Wireless Personal Communications, 98, 2025-2035.

[28] Highnam, K., Arulkumaran, K., Hanif, Z., & Jennings, N. R. (2021). BETH Dataset: Real Cybersecurity Data for Unsupervised Anomaly Detection Research. In CEUR Workshop Proc (Vol. 3095, pp. 1-12).

[29] Gebski, M., & Wong, R. K. (2007). An efficient histogram method for outlier detection.

In Advances in Databases: Concepts, Systems and Applications: 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007. Proceedings 12 (pp. 176-187). Springer Berlin Heidelberg.

[30] Nachman, B., & Shih, D. (2020). Anomaly detection with density estimation. Physical Review D, 101(7), 075042.

[31] Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In 2008 eighth ieee international conference on data mining (pp. 413-422). IEEE.

[32] Mahesh, B. (2020). Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], 9(1), 381-386.

[33] Xu, Z., Kakde, D., & Chaudhuri, A. (2019, December). Automatic hyperparameter tuning method for local outlier factor, with applications to anomaly detection. In 2019 IEEE International Conference on Big Data (Big Data) (pp. 4201-4207). IEEE.

[34] Fawcett, T. (2006). An introduction to ROC analysis. Pattern recognition letters, 27(8), 861-874.

[35] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information processing & management, 45(4), 427-437.

[36] python documentation Our Documentation | Python.org visited at 26/04/2024

[37] Spyder documentation Home — Spyder IDE (spyder-ide.org) visited at 26/04/2024

[38] pandas documentation pandas documentation — pandas 2.2.2 documentation (pydata.org) visited at 26/04/2024

[39] NumPy documentation NumPy – visited at 26/04/2024

[40] matplotlib documentation Matplotlib — Visualization with Python visited at 26/04/2024

[41] Seaborn documentation seaborn: statistical data visualization — seaborn 0.13.2 documentation (pydata.org) visited at 26/04/2024

[42] Sklearn documentation scikit-learn: machine learning in Python — scikit-learn 1.5.0rc1 documentation visited at 26/04/2024

[43] tkinter — Python interface to Tcl/Tk — Python 3.12.3 documentation visited at 26/04/2024

[44] PyOD documentation pyod 1.1.4 documentation visited at 26/04/2024

[45] customTikinter Official Documentation And Tutorial | CustomTkinter (tomschimansky.com) visited at 26/04/2024

[46] pandastable documentation Introduction — pandastable documentation visited at 26/04/2024