uc3m | Universidad Carlos III de Madrid

Dual Degree in Computer Science and Business
2021-2022

*Bachelor Thesis*

# "Design and Deployment of an Access Control Module for Data Lakes"

---


Marina Boyero Torrente

Tutors

José María de Fuentes García-Romero de Tejada

Jesús Muñoz Núñez


Colmenarejo, 2022



This work is licensed under Creative Commons **Attribution – Non Commercial – Non Derivatives**

# ACKNOWLEDGMENTS

After these years of learning, sacrifice, dedication, and enthusiasm, where I have grown professionally and personally, this important stage of my life comes to an end. Therefore, I would like to dedicate this chapter to all the people who have helped me get here.

First, I want to express gratitude to my parents, my main source of support and references, without them none of this would have been possible. Also, to my brothers, who I love and admire, and, in general, to my family, for encouraging me unconditionally.

Special appreciation to my granny Paqui, for her affection, her smile, her encouraging words before an exam or her calls afterwards to ask how it had gone, and for always being proud of her grandchildren.

I would also like to thank my project tutors, Chema, for his guidance and support during the realization of this work, and Jesús, for his patience and for making it possible for this project to see the light. Also, a big thanks to Iván, from who I have learnt a lot during this time.

Finally, to my friends, for being by my side. Although the names are not written, you know who you are.

# ABSTRACT

Nowadays big data has is considered an extremely valued asset for companies, which are discovering new avenues to use it for their business profit. However, an organization's ability to effectively extract valuable information from data is based on its knowledge management infrastructure. Thus, most organizations are transitioning from data warehouse (DW) storages to data lake (DL) infrastructures, from which further insights are derived.

The present work is carried out as part of a cybersecurity project in a financial institution that manages vast volumes and variety of data that is kept in a data lake. Although DL is presented as the answer to the current big data scenario, this infrastructure presents certain flaws on authentication and access control. Preceding work on DL access control points out that the main goal is to avoid fraudulent behaviors derived from user's access, such as secondary use[1], that could result in business data being exposed to third parties.

To overcome the risk, traditional mechanisms attempt to identify these behaviors based on rules, however, they cannot reveal all different kinds of fraud because they only look for known patterns of misuse.

The present work proposes a novel access control system for data lakes, assisted by Oracle's database audit trail and based on anomaly detection mechanisms, that automatically looks for events that do not conform the normal or expected behavior. Thus, the overall aim of this project is to develop and deploy an automated system for identifying abnormal accesses to the DL, which can be separated into four subgoals: explore the different technologies that could be applied in the domain of anomaly detection, design the solution, deploy it, and evaluate the results.

For the purpose, feature engineering is performed, and four different unsupervised ML models are built and evaluated. According to the quality of the results, the better model is finally productionalized with Docker.

To conclude, although anomaly detection has been a lasting yet active research area for several decades, there are still some unique problem complexities and challenges that leave the way open for the proposed solution to be further improved.

**Keywords**: Data Lake, log, event, audit, access control, security, machine learning, anomaly detection, unsupervised learning.

[1] secondary use or data misuse refers to the inappropriate use of data as defined when the data was originally collected.

# TERMINOLOGY/ ABBREVIATIONS

**Access control**: fundamental component of data security that dictates who's allowed to access and use company information and resources. Through authentication and authorization, access control policies make sure users are who they say they are and that they have appropriate access to company data.

**AE** – Autoencoder

**Alert**: message that is sent to a system or a person as a notification of a significant event that may require immediate attention.

**Anomaly detection**: identification of data points, events or observations that deviate from the normal behavior.

**Audit**: systematic, independent and documented process for obtaining evidence of the performance of operational procedures and evaluating it objectively.

**Data lake**: centralized repository for storing all structured and unstructured data at any scale.

**DL** – Data Lake

**DW** – Data Warehouse

**Event**: an identifiable action that happens on a device. Typical events include an action taken by a user (such as logging onto the device), an automated action (such as the results of a scheduled job), a detectable hardware or software event (such as a hard disk failure) or an external input (such as a network port scan).

**GMM** – Gaussian Mixture Model

**Log**: A record of a single event.

**Machine Learning**: branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn.

**ML** – Machine Learning

**Monitoring**: the process of pro-actively checking systems for information security incidents, normally by checking log messages and/or periodically verifying that the system is responding.

**SIEM** – Security Information and Event Management

**SOC** – Security Operations Center

**SOC analyst**: cybersecurity professional who works as part of a team to monitor and fight threats to an organization's IT infrastructure, and to assess security systems and measures for weaknesses and possible improvements.

**Unsupervised learning**: type of algorithm that learns patterns from unlabeled data.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

During the past decade, big data has experienced an exponential growth and, nowadays, it is considered an extremely valued asset for companies across the globe, which are discovering new avenues to use it for their business profit. From multinationals to small and medium enterprises (SMEs), data helps in support decision-making, monitoring and improving operational processes, or anticipating their target audience and customer preferences and needs.

Big data is defined by volume (amount of data captured or processed), variety (range of data sources and types) and velocity (speed or frequency with which data are recorded and analyzed). Additionally, it is a desirable attribute of big data obtaining relevant and significant insights from it.

An organization's ability to effectively extract valuable information from data is based on its knowledge management infrastructure's capability to transform data into strategic information, and it's one of the main challenges faced by enterprises in recent years. For this reason, most organizations are transitioning from data warehouse storages to data lake infrastructures, from which further insights are derived. Data lakes handle large (volume) and quickly arriving (velocity) bulks of data in its native format, whether it is structured or unstructured (variety), in contrast to data warehouses' highly structured data.

The present work is carried out as part of a cybersecurity project in a financial institution which comprises banking and insurance business, undertakes investments and collects information about more than 14 million clients. Hence, it all results in vast volumes and variety of data which is processed in real time and stored in the organization's data lake.

Although DL is presented as the answer to the current big data scenario, this infrastructure presents certain flaws related to authentication and access control. This is especially threatening for organizations because one of the principal menaces they face is the possibility of those who have access to data making an inappropriate use of it (also known as secondary use or data misuse). For instance, if the organization's IP address is used by an ex-employee or an outsider, it could result in business data being exposed to third parties.

To overcome the risk, traditional mechanisms attempt to identify fraud based on rules; however, they cannot reveal all different kinds of deception because they only look for known patterns of misuse. Thus, the present work proposes a novel access control system for data lakes, assisted by Oracle's database audit trail and based on anomaly detection mechanisms, that automatically looks for events that do not conform the normal or expected behavior.

## 1.1 Objectives

The overall aim of this project is to develop and deploy an automated system that can identify fraudulent accesses to the lake. To achieve the goal, it is broken down into a series of specific subobjectives:

- Explore the different technologies that could be applied in the domain of anomaly detection.
- Design a solution for automatically detecting and assessing anomalous accesses.
- Deploy the solution within the organization's infrastructure.
- Evaluate that the solution meets the established benchmark and produces the desired results.

## 1.2 Regulatory Framework

This section analyzes the recognized norms and standards applicable in the fields of security, data protection and Developing Operations (DevOps). It also reviews technical standards relevant to the applied programming language and the distribution of this project.

The final goal of this project is to help in the prevention of delicate data, such as client's information, being exfiltrated by malicious users. Thus, the present project is aligned with the General Data Protection Regulation (GDRP), which comprises a set of standards adopted by the EU to protect personal data and privacy of their residences [97]. Personal data means information (such as an identification number) about a natural person whose identity can be determined, directly or indirectly, from it [23]. For this reason, special care has been taken to maintain data confidentiality, and username encryption has been performed on logs, to protect identifiable information about an employee.

Additionally, because the designed solution relies on access auditing and monitoring, the Standard on Logging and Monitoring by the European Commission is reviewed [86]. According to this Standard, logging facilities must be protected against "deliberate and accidental threats". This is relevant for achieving the purpose of this project, since log files being edited or deleted by users could hide misuse (and this is intended to be detected in the first place) [86]. Thus, it has been guaranteed that events (logs) are stored as WORM (write one, read many), which means they cannot be modified (falsified) or deleted.

At the same time, the SLM Standard demands alerts raised by the monitoring system are analyzed and, where relevant, flagged as incidents in the Incident Management system to be followed up [86]. To facilitate this task, the developed solution relies on a Security Incident and Event Management (SIEM), which is designed to monitor all security-related events on an enterprise's network according to the best practices and guidelines established in the ISO/IEC 17799:2005.

On the other hand, although the EU Commission does not provide guidance on ML or Machine Learning Operations (MLOps) [33], 2675-2021 - IEEE Standard for DevOps [41] establishes that organization shall "apply policies and procedures for monitoring, audit, security, and quality control of integrated elements of the system". The Standard on Logging and Monitoring also indicates that "monitoring processes and systems must be reviewed regularly to ensure that they are performing adequately and not suffering from too many false positives or false negatives" [86]. Consequently, once the model is productionalized, its performance will be continuously monitored to ensure the best version of the model is available.

Furthermore, the Committee CTN 320 Cybersecurity and personal data protection [18] takes the lead in developing standards for cybersecurity, privacy, and data protection. Therefore, such organism should be aware of the data lake access control problem, and work on creating norms for its standardization.

Regarding the programming code, it relies on open-source libraries such as *scikit-learn*, *pandas* or *NumPy*, grouped under the BSD software license, which has zero cost and minimal legal issues [57]. *Python* and *matplotlib* are licensed under the PSF License Agreement, which does not claim ownership of any third-party code or content [52].

Finally, with the aim to promote research and knowledge transfer, the project has been openly published under Creative Commons license that allows third parties to copy and redistribute the material under the following terms:

1. recognize and properly cite the work and author.
2. is for non-commercial purposes.
3. do not produce derivative works thereof.

## 1.3 Socioeconomic Impact

To show the socioeconomic impact of the project, this section reviews some of the recent data breach incidents and their implications towards companies and other individuals.

In October 2017, Yahoo disclosed that "a breach performed by a group of hackers had compromised 3 billion accounts" [33] and resulted in user's information (names, email addresses, telephone numbers, encrypted or unencrypted security questions and answers, dates of birth, and hashed passwords) being exposed to identity theft. Furthermore, the breach was reported while Yahoo was in negotiations to sell itself to Verizon. As a result, Verizon closed the deal for a final price $350 million lower than it was originally agreed [33].

More recently, in March 2022, a misconfiguration in the AWS bucket[2] gave everyone free access to the Pegasus Airlines' database, which gathered millions of files about take offs, landing, and sensitive flight information [107]. Thus, the impact could have affected the safety of passengers and crew members of Pegasus. It is relevant to mention

[2] open distributed storage resource accessible in Amazon Web Services' (AWS).

that, even after the exposure was discovered by an external security company, the airline didn't have evidence of data compromise [107].

Although the previous attacks were committed by hackers outside the organization, most of the time, data leaks come from those who have authorized and legitimate access to company's assets [101]. In 2020, research estimated that the internal data breach's[3] average annual cost was USD $11.45 millions, with 63% of the incidents attributed to negligence [101]. However, this impact is not merely monetary, but also affects the reputation of the company, and its client/costumer/user's, who usually don't have the resources to recover.

The conclusion from this analysis is that cyberattacks through access abuse can severely harm a company, its employees, and its customers. Hence, the proposal of an effective access control solution for data lakes can have an incalculable positive impact on the company, and, especially, on people's safety.

## 1.4 Approach

To facilitate the reading of the document, this section presents a brief description of its content, composed of seven chapters and two annexes:

**Chapter 1 – Introduction and objectives**: offers an introduction to the current use of big data and data lakes within organizations, along with access control challenges this infrastructure presents. It details the objectives pursued, and applicable regulatory framework and socioeconomic impact of this project.

**Chapter 2 – State of the art**: informs about data lake's architecture and some of the current solutions to manage access control. Additionally, this chapter gives a panoramic of alternative mechanisms for combating fraud based on anomaly detection and reviews some studies where they have been used.

**Chapter 3 – Analysis**: presents a general perspective of the solution together with the study of the system's architecture, defining the components that will integrate it. The chapter also includes a study of the technologies to be used in the development and deployment of the system along with the specification of the use cases, software requirements and acceptance test plan defined to verify compliance with these requirements.

**Chapter 4 – Detailed design**: shows class diagrams for each of the components defined in the analysis chapter. Additionally, a set of sequence diagrams are shown to understand the different interactions that take place between the different components and classes of the system.

**Chapter 5 – Software deployment**: presents the most remarkable decisions of the implementation of the system along with the design of the acceptance test plan.

[3] data exposed by those who have authorized and legitimate access to it.

**Chapter 6 – Experimentation and results**: includes the experiments created to test the goodness of the designed models. Results include a comparison between them and a justification of which one to select, considering the anomalies detected and organization's considerations. Additionally, it shows the results of the acceptance test plan.

**Chapter 7 – Conclusions and future work**: presents the main difficulties and personal conclusions obtained from the development of the project, as well as some of the future lines to be explored to improve the results obtained, as well as alternative ways of doing it.

**Annex 1 – Project management**: details aspects related to the management of the project, including the planning of the work, the technical means used for its development, and the initial economic estimation of the project.

**Annex 2 – Templates**: shows the templates used for the definition of the use cases, software requirements and acceptance test plan.

# 2. STATE OF THE ART

The aim of this chapter is to inform about data lake's architecture and some of the current solutions to manage access control. Additionally, this chapter gives a panoramic of alternative mechanisms for combating fraud based on anomaly detection and reviews some studies where they have been used.

## 2.1 Data Lakes

This first section reviews data lake's advantages while comparing it to traditional storage infrastructures and evaluates some of the current solutions to manage access control.

Data lakes are centralized repositories for hosting massive raw, unprocessed enterprise data [26] [78]. These infrastructures are nurtured with raw unstructured or multi-structured data [26] from various sources (local or external to the organization) and have a great value for companies' Decision Support System (DSS) [78]. Although Data Warehouses (DW) have been commonly used in DSS [78], there are some reasons these solutions are not adapted to the current scenario.

(1) First, DWs only store structured data. However, in the business world, only 20% of the data are structured, and the other 80% conforms semi-structured and unstructured data [46]. As DLs are capable of storing data in unstructured and unorganized way, they are more suitable for the current big data scenario.

(2) Additionally, DWs follow a "schema-on-write" approach [46], since data is first modeled (according to a predefined schema) and then integrated into the DW. For this reason, DWs are only capable of answering predefined requirements, but are not able to attend out-of-box queries that need to be combined with unstructured data [46]. In other words, user queries are limited to the highly defined structured data.

Instead, DLs only define the structure of the data at the time it is used, applying a "schema-on-read" approach [26][46]. Thus, DLs are capable of handling non-predefined queries, which gives data scientists more flexibility to retrieve value from this unexplored raw data.

(3) Likewise, the extraction, transformation, and loading (ETL) of data to predefined schemas results in information loss [78], as the data that does not conform to the warehouse schema is disposed. DLs, on the other hand, store data in its original format, with no predefined data schemas, so no data is turned away [46]. Later, when queries are performed, data will be sent through ETL (extract, transform, load) pipelines, without prematurely stripping away vital information [45].

(4) On top of that, the cost of a DWs can grow exponentially because of the requirements of better performance, the growth of data volume and the complexity of the database [78]. Alternatively, DLs are based on an architecture with low-cost technologies [78], and its "schema-on-read" approach reduces the costs of preprocessing, transformation and data ingestion.

(5) Again, DWs are highly structured with highly governed data management [46]. Even though it is possible to change the DW design, it is very time consuming and requires huge efforts, as it is associated to several business processes [46]. DL, on the contrary, lacks the explicitly defined structure of DWs [46], therefore, is more flexible and agile.

(6) Finally, while DWs have been used by all kind of business professionals, DLs are more suitable for data scientists, data analysts, business intelligence (BI) professionals, etc., who apply statistical analysis, ML techniques [78] to drive real business value.

For a quick overview, Table 1 provides a summary of the cited differences between DWs and DLs.

| | **Data Warehouse** | **Data Lake** |
|---|---|---|
| Data | Structured, processed data | (Semi-)structured, unstructured data, raw/unprocessed data |
| Processing | Schema-on-write, could result on information loss | Schema-on-read, avoids information loss |
| Storage | Expensive | Low cost |
| Agility | Less agile, fixed configuration | High agility, flexible configuration |
| Users | Business professionals | Data scientists |

**Table 1**. Comparison of DWs and DL [46]

The above improvements make DLs worth exploring, but this solution also brings some challenges [73]. One of the biggest concerns with DLs is guaranteeing security (privacy and regulatory requirements) and access control [46] [21].

To recognize these challenges, it's important to understand the way data flows through the DL. Research [32] [72] [78] coincides that DLs present four principal stages, from data entry to its preparation for the final user: ingestion, storage, processing, and access. Additionally, Ravat & Zhao [78] propose a DL "functional architecture" (offered in Figure 1) with four differentiated zones: raw data, process, and access (which are divided, at the same time, in a treatment area (dotted rectangle) and a data storage space [78]), and a govern zone.

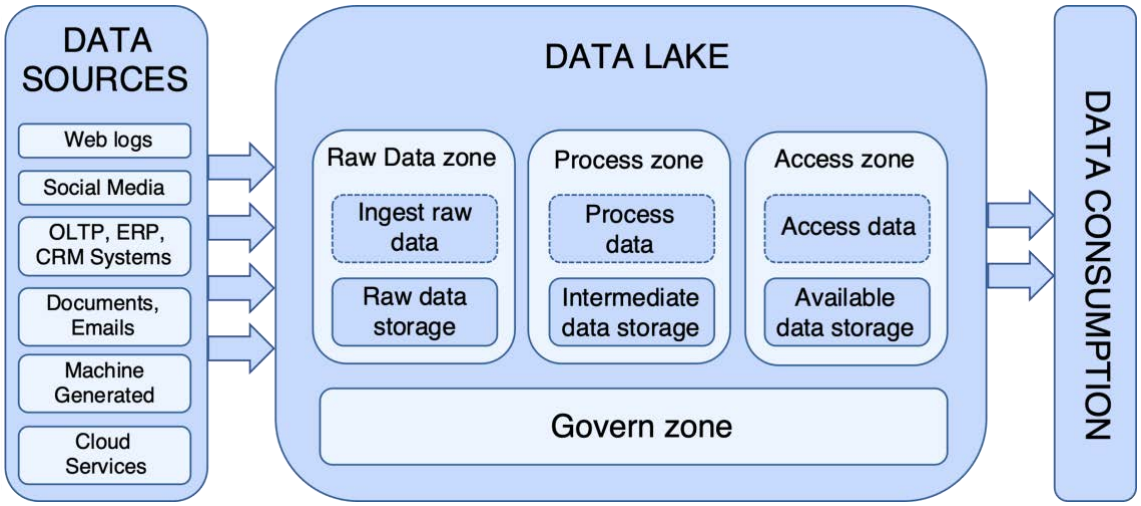



**Figure 1**. Data Lake architecture [78]

The first zone ingests raw data from different sources (such as web logs, social media, emails, online transaction processes (OTP), enterprise resource planning (ERP), customer relationship management (CRM) systems, etc.) and stores it. Following zone processes data, merging the new data with the existing data, while assuring its integrity, and providing a well-defined structure for consumption [72]. Lastly, data is accessed and consumed for different purposes.

The risk in which this project focuses occurs in this final stage, where data can be subject to malicious, unauthorized disclosure and/or use, that could severely harm an organization [72]. The role of DL's govern zone (see Figure 1) entails supervising data confidentiality, integrity, and availability, and ensuring the appropriate access privileges [72]. However, DLs poses a novel open access that conflicts with traditional governance approaches, for instance, from data warehousing [32]. According to [21] [32] and [46], there are still research gaps in DL's governance regarding access control. Thus, new models and systems must be defined and adapted to the DL's governance.

### 2.1.1 Preceding work on data lake access control

This subsection summarizes some of the contemporary proposed systems for managing access control in similar large-scale infrastructures as well as DLs.

On the one hand, since DLs' data architecture is closely similar to Apache Hadoop and its ecosystem of open source projects [26], it's relevant for this review to mention Gupta et. al's [35] attribute based access control for the Hadoop ecosystem (HeABAC).

It is based on the notion of grouping similar resources together and attaching a tag containing a set of operations that can be performed on the data. Tags are then assigned to roles, and roles are assigned to users, so those who have similar roles are allocated to groups. Groups are then defined in a hierarchy where users assigned to a higher group (parent) get all the roles of its child groups and, in turn, their resource's permissions.

This group hierarchy results beneficial for efficient role management for large-scale DLs, but the inheriting policy-based access doesn't consider access control requirements for individual users, and it is not very useful in limiting secondary use [88] (a user having permission to access a piece of data, doesn't mean that will use it lawfully). When data is used for a different purpose than it is actually intended for, it is considered a privacy violation and may lead to legal consequences [88].

To address this issue, Sultan & Jensen [88] introduce 'collection purpose', a piece of metadata attached to the data that records information such as "why the data was collected", "how it may be used", "conditions/requirements necessary for sharing", etc. This way, whenever the resource is accessed, the user must present a compatible 'access purpose'. In other words, the 'access purpose' is evaluated against the 'collection purpose' to decide whether the resource should be accessible (access control decision) and, if so, to what extent.

The main limitation of this proposal is that it's hard to know the purpose of unexplored, raw data, so its authors propose the resource is processed in order to be assigned an 'access purpose'. However, the nature and one of the main advantages of Data Lake is that data is only processed when a user requests its (schema-on-read).

This literature review points out that the main goal is to avoid fraudulent behaviors, such as secondary use, derived from user's access. Hence, the present work proposes a novel access control module for DLs that automatically looks for events (accesses) that do not conform the normal behavior, based on anomaly detection mechanisms, and assisted by Oracle's database audit trail.

## 2.2 Protection Mechanisms for Combating Fraud

This section recalls the significance of the problem that is tackled and introduces current mechanisms for combating fraud. It also identifies the better approach according to the requirements of the project

As it was previously introduced, anyone with the right level of access to a company's data could potentially expose confidential information, which would have a major impact on the organization [101]. Whether accidentally or deliberately, this type of attack is also referred as insider threat [101] [43], could result in millions of cost for the organization and irreparable damage for its clients.

As represented in Figure 2, to combat fraud, prevention and detection are the two commonly used mechanisms. Prevention [1] is the first layer of security against fraud. Its purpose is to avoid fraud from occurring in the first place by encrypting data or using firewalls for the connections between the internal privately owned network and external networks. Fraud detection [43], on the other hand, refers to the discovery of criminal activities occurring in organizations, including insider threat. More precisely, insider threat refers to the discovery of users within an organization undertaking malicious activities that could lead to information theft, sabotage, fraud, and data exfiltration [43].

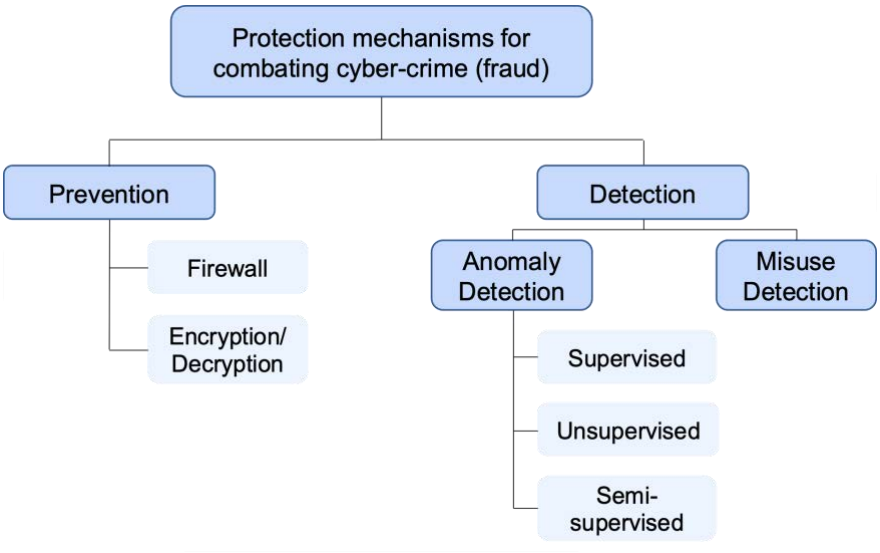



**Figure 2**. Mechanisms for combating fraud [1]

For detection, there are two methodologies. The first one is the misuse approach, based on rules, statistics, or heuristic methods that first define fraudulent behavior and then classify all other behaviors as normal [1]. Although it is considered fast and simple, its major limitation is that it is not possible to detect all different kinds of fraud because it only looks for known patterns of misuse [1]. For this reason, most techniques used to identify fraud and insider threats are based on anomaly detection [1], which look for events that do not conform to expected behavior without previously describing what is normal.

## 2.3 Anomaly detection

This section tackles anomaly detection by providing a definition of this concept and describes the two main methodologies that exist for identifying anomalies.

Anomaly detection, outlier mining, novelty detection or outlier modeling are different ways of referring to the process of detecting outliers in ML and data mining [99]. An outlier is an item, event or observation which does not obey to an expected pattern or other items in a dataset [99].

Although, there are straightforward methods to identify outliers consisting of histogram and statistical tests, such as boxplot, trimmed mean, extreme studentized deviate and the dixon-type test [99]; generally, more precise anomaly detection relies on the extraction of features from users' activities (login/logon, file access, email, messages, or web-browsing) and the design of anomaly detection models based on ML algorithms [43].

In this area, there are two main approaches, depending on how and what the models learn from the input data: supervised and unsupervised learning. Table 2 shows major differences between them.

| | Supervised | Unsupervised |
|---|---|---|
| Input Data | Labeled (e.g.: “normal”/ “anomalous”) | Unlabeled |
| Purpose | Predict outcomes for new data | Discover the inherent structure and patterns in data |
| Techniques | • Classification<br>• Regression | • Clustering/Segmentation<br>• Dimension reduction |
| Common applications | ○ Spam detection<br>○ Pattern detection<br>○ Natural Language Processing<br>○ Sentiment analysis<br>○ Automatic image classification<br>○ Weather forecasting<br>○ Pricing predictions | ○ Object segmentation (users, costumers, movies, products, etc.)<br>○ Similarity detection<br>○ Automatic labeling<br>○ Anomaly detection<br>○ Recommendation engines |

**Table 2**. Major differences between Supervised and Unsupervised learning [20]

There is a third method that lies in between supervised and unsupervised learning and has been proposed to overcome the main problems of both approaches [79], which is semi-supervised learning (SSL). Unlike supervised learning, SSL is capable of learning with small amount of training data. For instance, in this context of anomaly detection, SSL would learn from a dataset with observations belonging to the normal class, which is useful in cases where it may be difficult or expensive to model an anomaly [38]. Additionally, because it is based on labeled data, it can classify the unknown (or) test data more precisely than unsupervised learning [79].

Supervised and unsupervised learning encompass different algorithms that can be classified as presented in Figure 3.

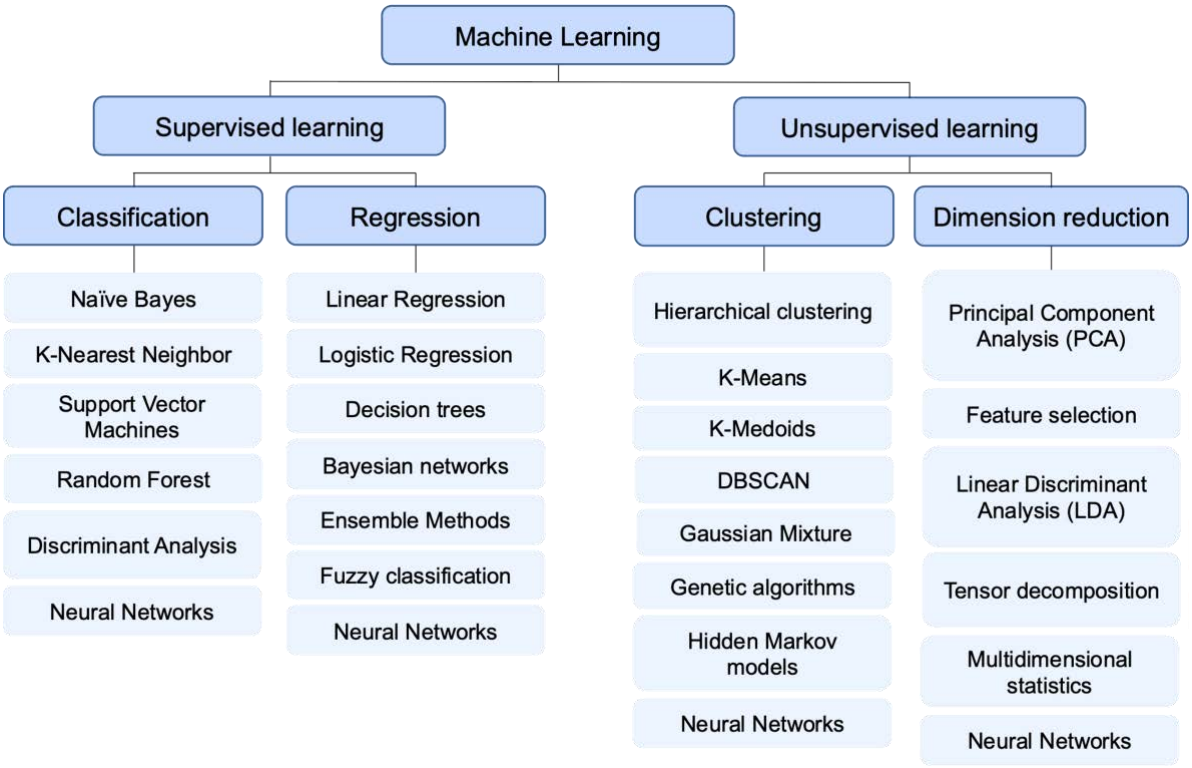



**Figure 3**. Taxonomy of ML algorithms [6] [89] [91]

### 2.3.1 Preceding work on anomaly detection

This section revisions other studies that have applied anomaly detection for combating fraud and gives a panoramic of algorithms that could be explorable for the current task.

Over the last couple of decades, there has been an increase in the amount of research seeking to apply ML algorithms to detect anomalous behaviors in the cybersecurity field [24].

For instance, anomaly detection techniques have been applied to detect financial fraud related to credit card fraud, insurance fraud, money laundering, and others. According to Hilal et. Al [38], frequently used models for financial fraud are decision trees (DT), naïve Bayes (NB) classifiers, support vector machines (SVM) logistic regression (LR), K-means clustering or k-nearest neighbors (kNN), which can be used individually or ensembled with others. Moreover, deep learning has emerged and demonstrates significantly better performance, including neural network (NN) architectures such as convolutional neural networks (CNN), long short-term memory networks (LSTM) and generative adversarial networks (GANs).

Although supervised methods are shown to have improved performance, unsupervised methods are mostly adopted for anomaly detection because the massive volume of activities within an organization implies most of the collected data is unlabeled [43], and there is a problem of class imbalance [99], since the amount of normal data is much higher than abnormal data.

Hilal et. Al [38] also review up-to-date unsupervised algorithms such as Isolation Forest (IF), Self-Organizing Maps (SOM) and autoencoders (AE). In the context of security and anomaly detection, the Self-Organizing Map (SOM) and autoencoders are finding new applications never approached. Recently, Barletta et al. [9] evaluated the performance of an anomaly-based intrusion detection system using an unsupervised Kohonen SOM neural network for the identification of attack messages sent on the CAN bus. Likewise, Wang et al. [99] remark the use of autoencoders as an unsupervised deep model that detects anomalies through reconstruction of errors.

In addition, Kabir et al. [44] offer deep learning and traditional unsupervised learning algorithms for intrusion detection. They investigate the performance of two unsupervised algorithms – K-means and Self Organizing Maps (SOM) – and two deep neural architectures – deep autoencoding Gaussian mixture model (DAGMM), and adversarially learned anomaly detection (ALAD) – on identifying "unseen" and "minority" attacks.

Alternatively, graph-based learning methods have been the focus of some studies. Recent work discusses the use of graph convolutional networks (GCN) [43] for insider threat detection and fraud detection. This model is said to be an extended version of convolutional neural networks (CNN) that overcomes its limitations.

This review on anomaly detection techniques shows that there are numerous ways to integrate AI into cybersecurity and that it is possible to evaluate and compare the performance of various ML algorithms. It will also lead to a better understanding of the topic and provide background on the availability of technologies.

# 3. ANALYSIS

This chapter presents a general perspective of the solution together with the study of the system's architecture, defining the components that will integrate it. There is also included a study of the technologies to be used in the development and deployment of the system, along with the specification of the use cases, software requirements and acceptance test plan defined to verify compliance with these requirements.

## 3.1 General Perspective

This section addresses the system that will be developed to cover the first defined objective of the project: detect fraudulent or anomalous access to the data lake.

As pointed in the literature revision, anomaly detection broadly consists in training a machine learning model in order to find outliers. In this context, the dataset is made of logs that capture user's access activity. Logs or events (these terms will be used indistinctly) represent a source for finding traces and detecting malicious activities [2].

For this purpose, security information and event management systems (SIEMs) [56] play a major role. SIEMs are responsible for collecting logs from different areas of an enterprise and presenting security operations center (SOC) analysts the most critical security events to act on [56]. In case the event requires deeper analysis, the SOC creates a ticket (to keep track of the threat being handled) and forwards the details to the next analyst [61].

Once the ML model is trained, Figure 4 shows the pipeline/overall steps in which the anomaly detection system will work. First, the event (log with information about the user access to the Data Lake) is sent by the SIEM to the system. Then, feature engineering is performed so the model can read data. Finally, the model automatically computes a score that indicates whether the access is anomalous or not.

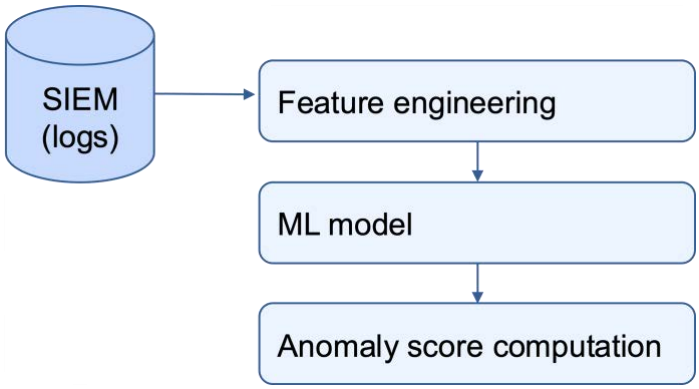



**Figure 4**. General process for anomaly detection

## 3.2 Technological Research

This section provides an overview of different technologies that could be applied for developing the described system. In the following subsections, a range of imposed technologies, as well as applicable unsupervised algorithms and programming languages will be studied and finally selected according to a certain criterion.

### 3.2.1 Imposed technologies

This first subsection reviews the only technological imposition, which is given by the organization requirements for productionalizing the solution. So, for integrating the machine learning model into the existing business infrastructure, Docker will be selected.

Docker is a tool designed to make it easier to create, deploy, and run applications [82] because it ensures the same environment [81]. The idea is that if an application or script is able to run on a particular machine, it can easily run anywhere else as well. To replicate the behavior of the application irrespective of the host on which it's running, Docker relies on containers [81].

Containers are independent and isolated environments that allow packaging all the required files needed by an application (such as libraries, binaries, and other dependencies) within a single package [82]. Docker containers are nothing but the running instance of an image, and, at the same time, images are created by building a Dockerfile. Figure 5 highlights this process.

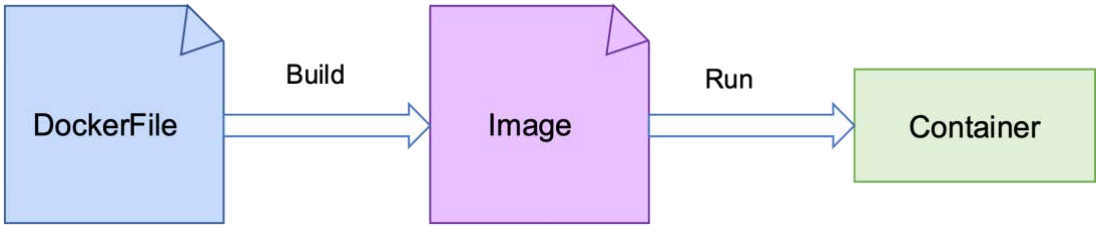



**Figure 5**. Process to run a container on Docker

So, the starting point of the Docker container lifecycle is the Dockerfile. A Dockerfile is a simple text file with no extension that contains the requirements to build the Docker image [81]. Docker images, on the other hand, act like a source of all the required files and directories to run the specific application [81], that is, the container.

### 3.2.2 Applicable algorithms for anomaly detection

This subsection further explores state-of-the-art unsupervised algorithms that could be applied for the task of anomaly detection.

As pointed before, unsupervised learning is more adequate for this problem of anomaly detection because: 1) data is unlabeled and 2) the amount of normal data is much higher than abnormal data. In this subsection, a variety of anomaly detection algorithms will be reviewed to justify the final selection.

#### 3.2.2.1 Clustering

Anomalies are typically found with respect to clustering structure in data [102]. By clustering, similar objects within a dataset are grouped together while dissimilar objects are kept separated in different groups [48]. Because there are over a hundred clustering algorithms [102], this overview will list the prominent ones, classified in hierarchical, centroid-based, density-based, distribution-based and neural network clustering [105].

##### 3.2.2.1.1 Hierarchical clustering

The core idea behind hierarchical clustering algorithms is that an observation is more related to nearby observations than to those farther away, so clusters are formed based on the distance between observations [105].

Hierarchical clustering utilizes dendrograms to show the hierarchy of clusters and the height at which they merge. For instance, reading the dendrogram presented in Figure 6 from the bottom up, observations 1 and 2 are most similar to each other than they are to any other, so they could be clustered into one group. Figure 6 also shows that observation 3 is completely separate from the others, which could be interpreted as being substantially different.

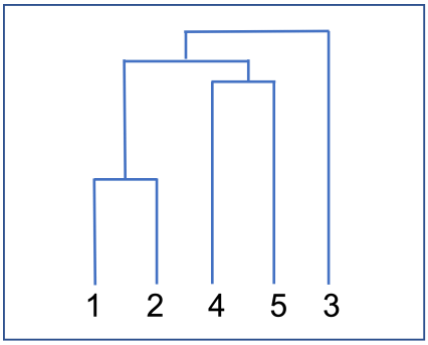



**Figure 6**. Dendrogram to exemplify hierarchical clustering

However, this type of clustering is not the best option for identifying outliers because they may not always appear as isolated clusters but rather cause the merging of different clusters [105], which would difficult anomaly identification.

##### 3.2.2.1.2 Centroid-based clustering

Another type of clustering is the centroid-based [105]. The most popular centroid-based clustering algorithm is K-means, which identifies k (known a priori) number of centroids (cluster's centers) and allocates each data point to its nearest cluster based on the squared Euclidean distance between the observation and the cluster's centroid [25]. In general, there is no method for determining exact value of k, however the Elbow method approach can be used, which selects the value of k that causes sudden drop in the sum of squared distances [62]. This value can be validated with the silhouette method [62]. Once k is determined, the algorithm iteratively updates centroids by calculating the new means of the observations in the cluster until a stopping criterion, i.e., centroids no longer change [25].

K-means has no notion of outliers [62], so all observations are assigned to a cluster, even if they do not belong in any. In the domain of anomaly detection, this means anomalous points will be assigned to the same cluster as "normal" data points, but they will be found far from the centroids of the cluster they belong to [38], as exemplified in Figure 7, or appear separately in form of small clusters [38].

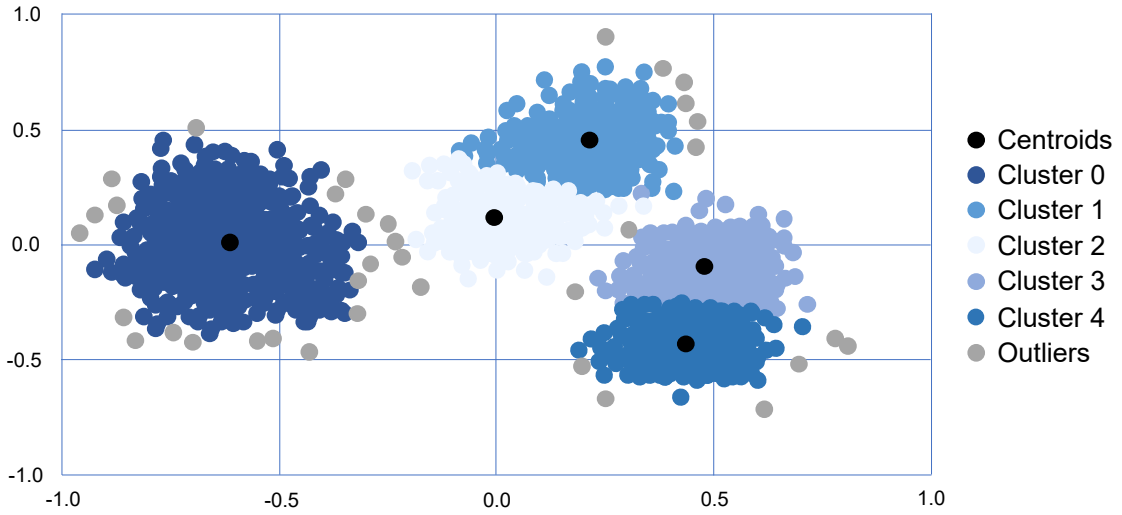



**Figure 7**. Outlier detection in K-means

Similarly, k-medoids breaks the dataset up into k groups, but, in contrast to K-means, it chooses datapoints as centers (medoids). For each medoid (m) and each data point (p) associated to m, medoids are updated by swapping m and p and computing the total cost of the configuration (that is, the average dissimilarity of p to all the data points associated to m), and then selecting as medoid the p with the lowest cost of the configuration. [25]

Thus, the main difference between K-means and k-medoids is that the first one attempts to minimize the total squared error, while the second minimizes the sum of dissimilarities between points assigned to a cluster and the center of that cluster (medoid) [25].

Although K-means should be considered for its simplicity and availability (since most of the popular machine learning packages contain an implementation of this algorithm), one of its drawbacks is that it assumes that clusters are convex (of spherical shape) [62], so when clusters are non-circular, trying to fit circular clusters may result in a mixing of cluster assignments where the resulting circles overlap [62] (see Figure 8). This is where density-based clustering comes in.

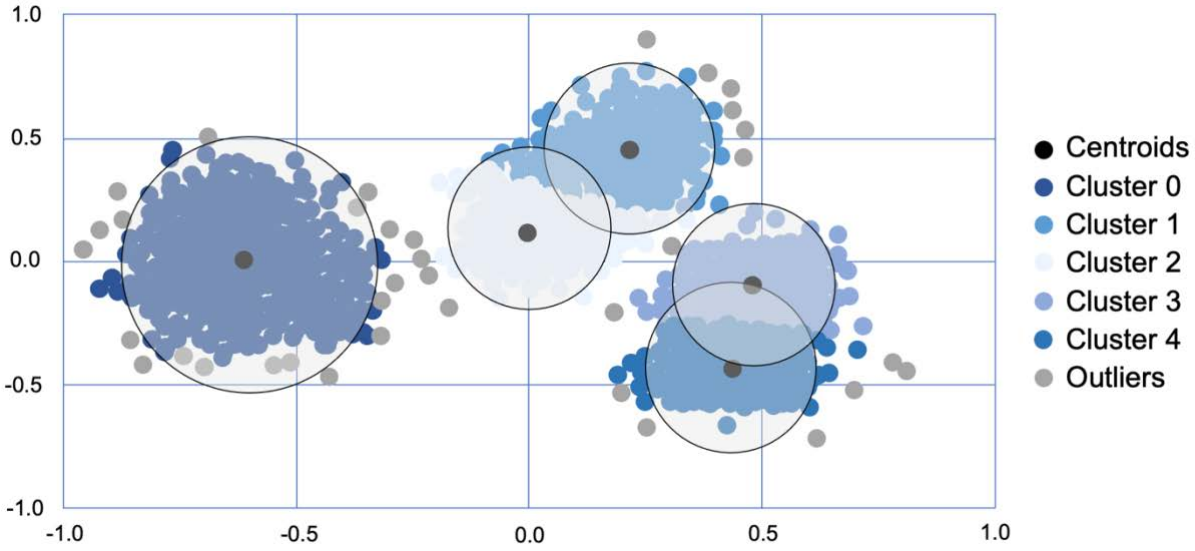



**Figure 8**. K-means cluster overlapping

#### 3.2.2.1.3 Density-based clustering

One of the most popular density-based clustering methods is DBSCAN (Density-based Spatial Clustering of Applications with Noise). DBSCAN can find arbitrarily-shaped and non-linearly separable clusters [106], like the ones presented in Figure 9, which could not be adequately clustered with K-means.

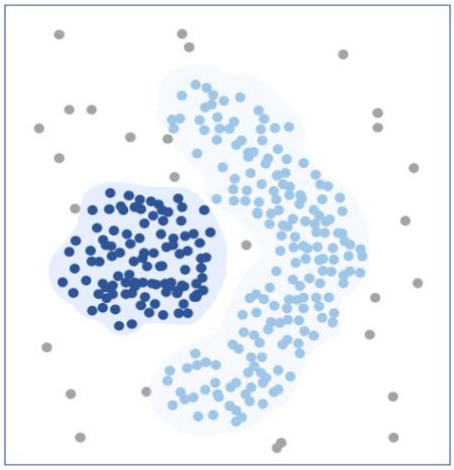


**Figure 9**. Arbitrarily-shaped, non-linearly separable clusters

To do this, DBSCAN locates regions of high density separated by regions of low-density [48]. The points in low-density regions are called "noise", while high-level regions are formed of "core" or "border" points. A point p is classified as core, border or noise [48] according to the two hyperparameters: Eps (ϵ) and MinPts. Eps (ϵ) describes de radius of a circle with center p, while MinPts indicates the minimum number of points that should be contained in the circle in order to identify the point p as core. If the number of points within the circle (with center p and radius ϵ) is less than MinPts but some of these points are core, p is considered a border point.

To understand what noise is and how clusters are formed, reachability and connectivity concepts must be defined. First, a point q is directly density-reachable from p if q is within the ε-radius of p and p is a core point [31] (see Figure 10). Alternatively, "a point is called density-reachable from another point if they are connected through a series of core points" [40] (see Figure 11).

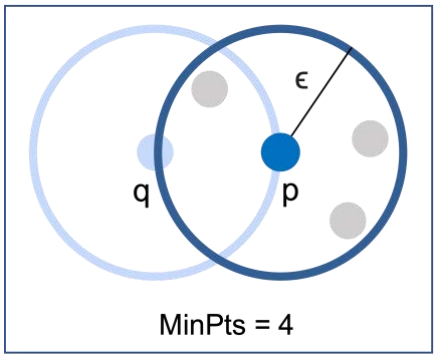



**Figure 10**. Direct density-reachability

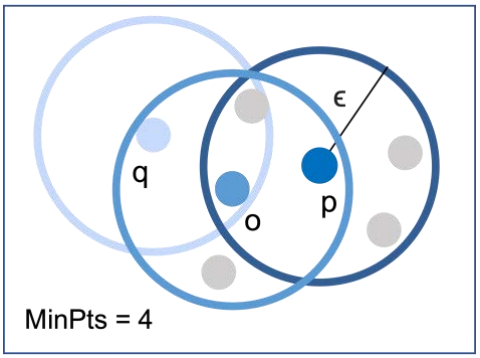



**Figure 11**. Density-reachability

In Figure 10, only q is directly density-reachable from p, but p is not directly density-reachable from q, since q is not a core point. Similarly, in Figure 11, o and p are core points, but q is not, so q is density-reachable from p but p is not density-reachable from q.

Finally, two points p and q are density-connected if both p and q are directly or just density-reachable from a point o [106]. Figure 12 shows that p and q are directly density-reachable from o (since o is core), so p and q are mutually density-connected.

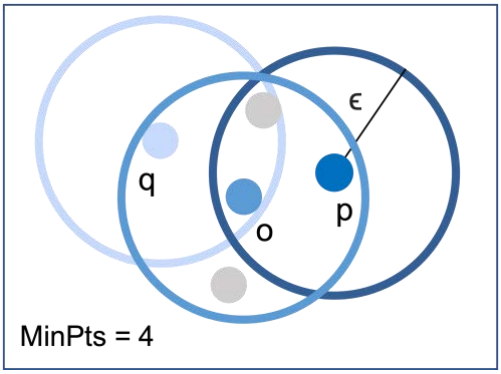



**Figure 12**. Density-connectivity

Thus, a cluster is formed by points that are mutually density-connected [106] and points that are density-reachable from some point of the cluster [106]. All points not reachable from any other point are considered noise. Figure 13 illustrates the graphical difference between core, border, and noise points.

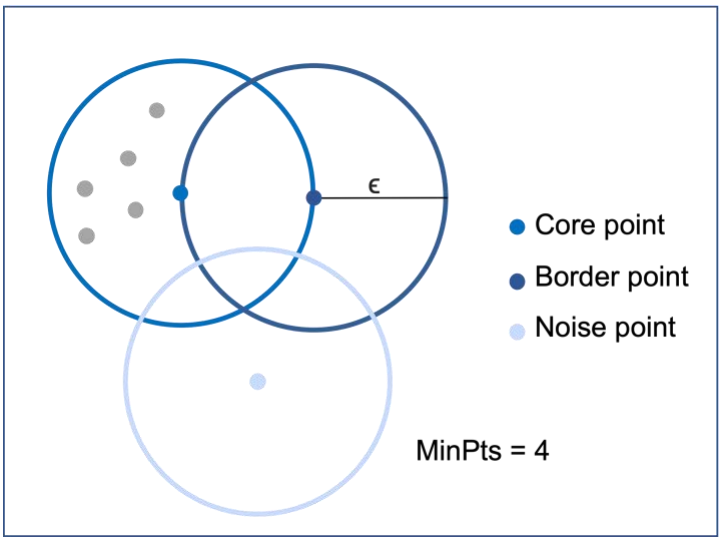



**Figure 13**. Core, Border and Noise points in DBSCAN

The main drawback of DBSCAN is that, although it does not require knowing the number of clusters a priori, the algorithm is very sensitive to hyperparameters (with a small change in Eps, the formation of clusters largely variates), which can be difficult to find if the domain expertise fails to understand the data very well [84]. Additionally, DBSCAN checks the ε-neighborhood of each point in the data, which makes the algorithm time complexity rise to $O(n^2)$ [83].

#### 3.2.2.1.4 Distribution-based clustering

Distribution-based clustering algorithms [105] offer new ways of allocating observations, where they can be in multiple clusters simultaneously (with different probabilities), in contrast to K-means, where data point were assigned to either one cluster or another [62]. Additionally, it is contemplated for clusters to overlap and adopt oblong shapes, whereas K-means tried to fit clusters into circles, which sometimes resulted in a poor fit and in circles to overlap when real groups were non-circular [62]. This is exemplified in Figure 14, where distribution-based clustering provides a better fit to the data than K-means.

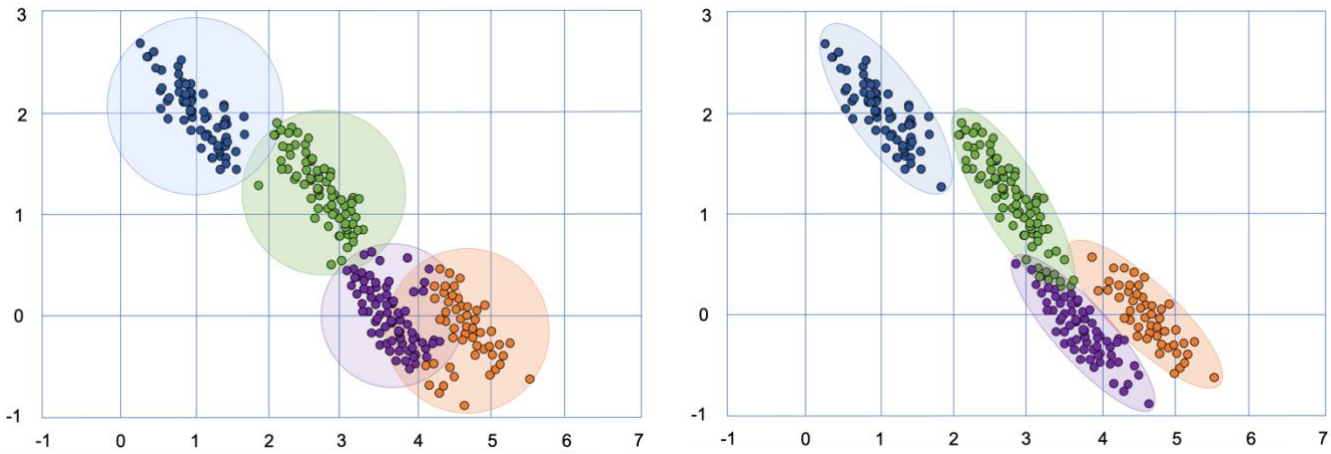


**Figure 14**. K-means (left) vs. Distribution-based clustering (right)

One of such algorithms is Gaussian Mixture Models (GMM). This algorithm assumes data generated from a mixture (i.e., superposition) of k gaussian distributions[4] (bell shapes) [95]. So, as presented in Figure 15, the algorithm works with three parameters: mean ($\mu$), variance ($\sigma$) and weight ($\pi$), associated to each gaussian [96].

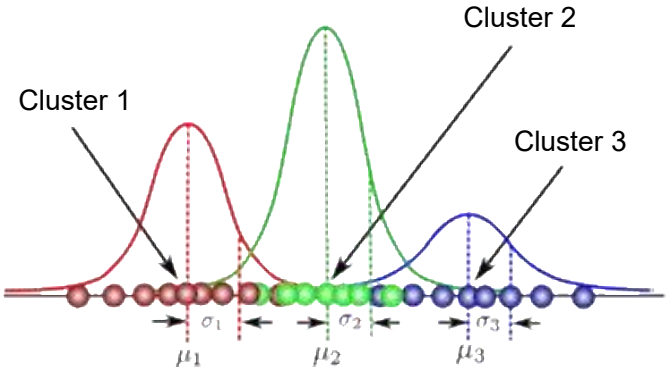



**Figure 15**. GMM's parameters associated to each gaussian [30]

The number of components (k) in a Gaussian Mixture can be efficiently selected with the BIC criterion [29], however, the main difficulty in GMM when data is unlabeled is that it is hard to know which points came from which latent component [29]. Fortunately, there is an iterative algorithm called expectation-maximization (EM) [29] which estimates the value of $\mu$, $\sigma$ and $\pi$ by (i) calculating the expected probability of each observation belonging to a cluster [96], followed by (ii) a "maximization of the expectations with respect to the model parameters". The second step consists in computing, for each point, the probability of being generated by each component of the model and then updating the values of $\mu$, $\sigma$ and $\pi$ [96] to maximize the likelihood of the data given those assignments [29]. Repeating this process is guaranteed to always converge to a local optimum [29].

Once the parameters have been learned, the probability density of each observation belonging to each component, and the distribution as a whole, can be calculated [4]. Finally, anomalies are identified as observations with very low probability density [4].

#### 3.2.2.1.5 Neural network clustering

The Self-Organizing Map, commonly known as Kohonen [55], is a type of Artificial Neural Network (ANN) that allows the visualization of high-dimensional data on a low-dimensional (usually 2D) grid. However, this algorithm differs from other ANN in that it can be trained by unsupervised learning, so it does not require the label values of the input vectors [9]. In fact, SOM by Kohonen has a stronger connection with the K-means algorithm [68].

A presented in Figure 16, Kohonen computes similarities among data in the input layer and represents them in an output layer of interconnected neurons. In order to find similarities, the output layer is formed of neurons and n-dimensional codebook vectors (weights) are assigned to each neuron. Additionally, the SOM network uses competitive

[4] Gaussian distributions are also referred as gaussians, components or clusters.

learning [9], so, given an input data point, the winner neuron, called Best Matching Unit (BMU), will be the closest one. After that, the codebook vectors [9] of that neuron are updated on a weighted average in order to move closer to the input vector (data point). Additionally, the neighboring neurons adjust their codebook vectors in order to better match with the input vector, thereby ensuring that neighboring neurons have similar codebook vectors. Thus, similar data points will be mapped in neighboring neurons.

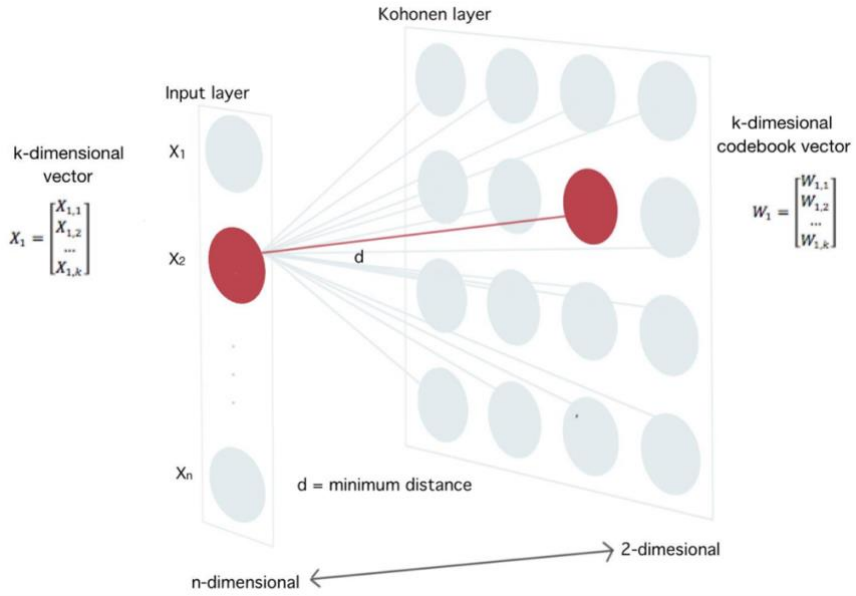



**Figure 16**. Kohonen Self-Organizing Map (SOM) neural network [9]

In conclusion, at the end of the learning process, the weights (codebook vectors) are grouped in clusters depending on their distance in the input space. Similar to K-means, the maximum distance (between an input vector and its BMU) of the normal data could set a threshold for declaring a new case to be anomalous.

Nevertheless, some of the drawbacks of this algorithm are that it results time-consuming when dealing with big datasets [66] and requires adjusting the number of parameters, the values for the training parameters and the size and topology of the map [66].

#### 3.2.2.3 Autoencoders

Autoencoders [8] (AEs) are mainly designed to encode the input data into a compressed and meaningful representation, and then decode it back such that the reconstructed input is similar as possible to the original one. To detect anomalies, the reconstruction error is used [13]. Since autoencoders can be trained to minimize this error, in doing so, they learn the relationships among the features of the input set. So, a trained AE should reproduce a new input that resembles the data used for the training. If this is not the case, the autoencoder cannot correctly reconstruct the input and the error will be greater [13].

In this case, although data is unlabeled, the train dataset is imbalanced, since the amount of normal data is much higher than abnormal data; so the AE will learn from data that most of the times (more than 99% of times) follows a normal pattern.

Figure 17 represents the basic architecture of an AE.

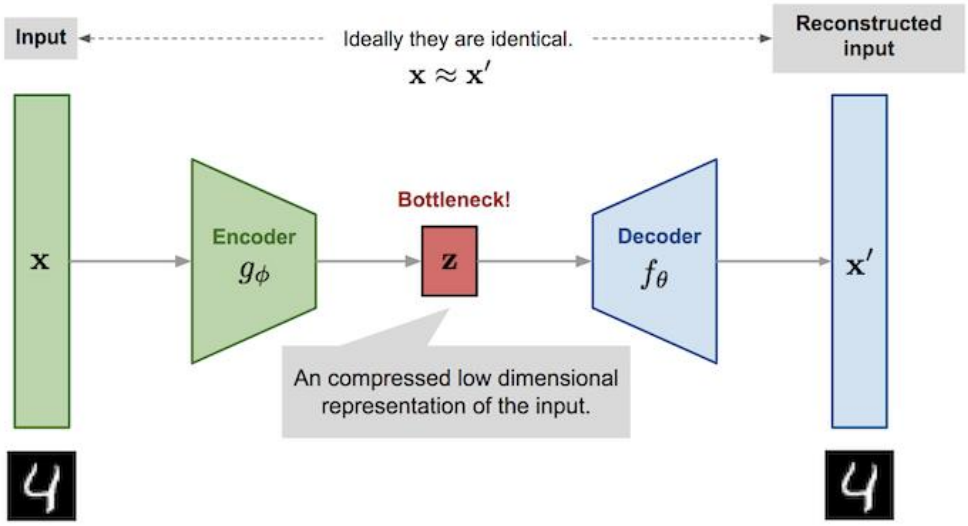



**Figure 17**. Autoencoders architecture [7]

### 3.2.2.4 Isolation Forest

According to Liu et. al [55], two major drawbacks of clustering-based methods are: 1) they are designed to report normal instances, but not optimized to detect anomalies, so they may cause too many false alarms (having normal instances identified as anomalies) or too few anomalies being detected; 2) in many cases, existing methods are constrained to low dimensional data and small size datasets because of their high computational complexity. Consequently, these authors propose Isolation Forest [55] for explicitly isolating anomalies instead of profiling normal points.

Isolation Forest (or iForest) is based on the notion that anomalies are more susceptible to isolation than normal points because they are 'few and different' [55]: they are the minority, consisting of fewer instances, and, their feature's values are very different from those of normal instances [55]. One of the advantages of this algorithm is its capacity to scale up to handle extremely large data size and high-dimensional problems [55] since it does not rely on distance or density measures, thus, eliminates major computational cost of distance calculation, present in all distance-based methods and density-based methods. Instead, iForest relies on the construction of isolation trees (iTrees), binary trees where each node has exactly zero or two child nodes [33].

Isolation Forest involves training and testing [55], and takes two parameters as input: the sub-sampling size ($\psi$) and the number of trees ($t$) [55]. When training, first, the algorithm starts by selecting a random feature and computing a random threshold within the range of minimum and maximum values of the selected feature. After that, branching is done considering the feature's value of the data point: if it's less than the selected threshold, it goes to the left branch, otherwise goes to the right, as represented in Figure 18. Figure 19 shows how this process is recursively performed until all instances are isolated or a specific tree height ($\psi$) is reached [55].

In the testing stage, test instances are passed through isolation trees and are assigned a score of -1 (if anomalous), or a 1 (if classified as normal). The underlying idea is that anomalies are typically found on short average path lengths on the iTrees [55], thus, data points are assigned a score according to their path lengths.

Moreover, although this algorithm works with hyperparameters, its authors give an empirical estimation of their values. They demonstrated that a sub-sampling size ($\psi$) of 256 generally provides enough detail to perform anomaly detection across a wide range of data and increasing this value would only increase processing time and memory size, since it does not gain in detection performance [55]. Finally, the number of trees or ensemble size (t) usually converges well before 100, so this value should be used as default.

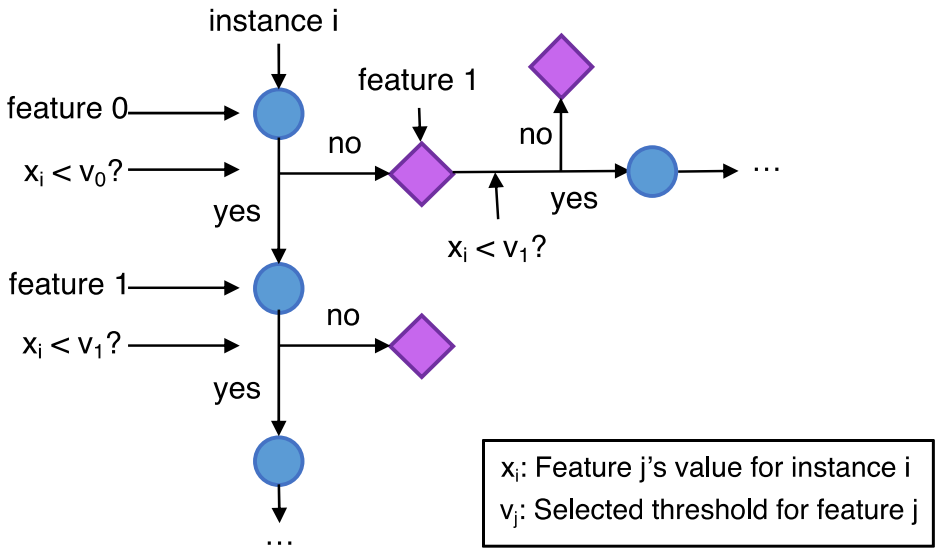



**Figure 18**. Isolation Forest branching procedure during the training stage

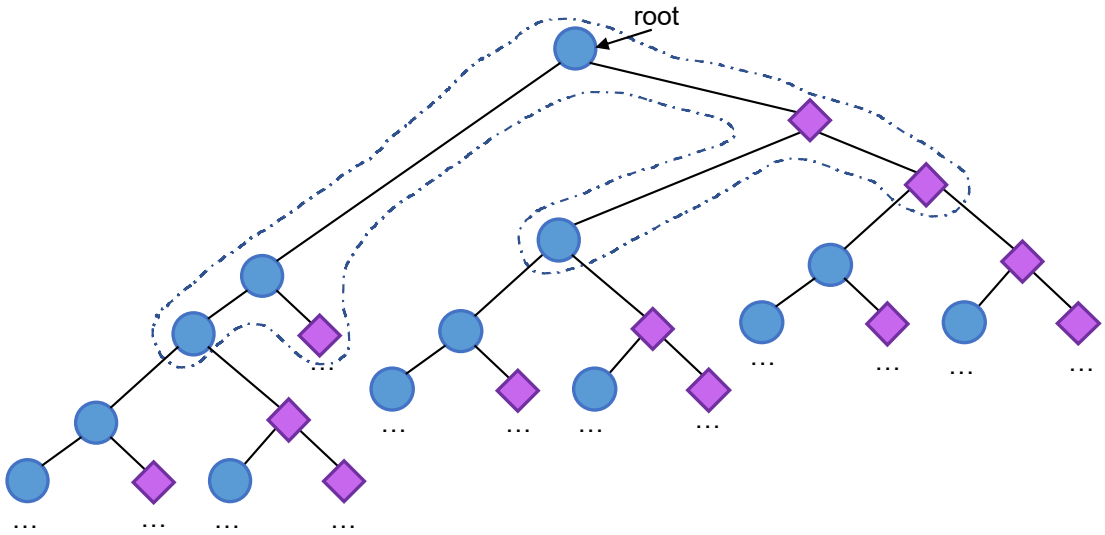



**Figure 19**. Representation of construction of isolation trees during the training stage

#### 3.2.2.5 Principal Component Analysis

Principal Component Analysis (PCA) is a standard technique popularized in modern data analysis that can be used for dimensionality reduction, data compression, feature extraction, and data visualization [49]. In this section, PCA is going to be analyzed for its property of reducing dimensionality of large data sets.

Scientifically, observations are described by several dependent variables that, in general, are intercorrelated and include noise. Hence, this tool [49] is especially relevant in problems that could suffer the called curse of dimensionality [100], used to describe the extraordinarily rapid growth in the difficulty of problems as the number of variables (dimension) increases.

PCA is used to extract the important information from such observations and to reduce the noise [49]. To achieve this, PCA computes a set of new orthogonal variables (called principal components) which are obtained as linear combinations of the original variables [49]. PCA can be also formulated as the linear projection that minimizes the average projection cost (defined as the mean squared distance between the data points and their projections) [49]. Finally, the number of principal components is chosen in a way that the first few capture a large part of the total variance [100].

### 3.2.3 Applicable programming languages for ML

This subsection revisions some of the programming languages that could be applied for the design and deployment of the model/s.

Over the past 20 years, a variety of machine learning algorithms have been developed and implemented in different programming languages [36]. R, MATLAB, C/C++, Julia, GO, or Python are commonly used [37] [28].

First, R provides support for machine learning in the form of packages such as glmnet, h2o, ranger, xgboost, and keras [11] to effectively model and gain insight from data. This language makes it possible to work with data structures such as vectors, factors, lists, arrays, matrices and dataframes [50]. Additionally, since it is very common to store datasets in text files, R allows importing and exporting from and to programs such as Microsoft Excel, providing a quick and easy form of working with spreadsheet data [50]. However, R stores data in system memory (RAM), which is a constraint when working with big amounts of data.

Then, MATLAB provides high-quality algorithms for data analysis and visualization [69], but its main drawbacks are the cost of its software and its running time, which is comparably much slower than C/C++ [5]. In contrast, C/C++ is highly efficient [28] and offers TensorFlow and other learning frameworks that are implemented in this language [42]. However, it requires programming algorithms from scratch [42] and presents difficulties in learning and usage [28].

Among the cited languages, Julia and Golang are considered for their enhanced performance.

On the one hand, Golang (or GO) is syntactically similar to C and provides memory safety, garbage collection and structural typing, and machine learning libraries written directly in this language. Moreover, it provides GoLearn, a library that allows construction of ML algorithms, model fitting and data splitting in train and test datasets [63].

On the other hand, Julia is an open-source language that was originally designed for high-performance computing [28]. This language has developed third-party libraries for machine learning [28] but they are not as mature as Python's [14]. For instance, there is no Julia PCA (Principal Component Analysis) package, so Python is the most frequently used language for developing PCA algorithms [28].

Lastly, Python is a high-level language with simple syntax and an extensive collection of scientific libraries and packages, many of whom are specific for developing machine learning, as listed in Table 3.

| Package/Library | Description |
|---|---|
| Pandas [70] | • Provides data structures (such as dataframes) useful for performing data analysis in Python.<br>• Most powerful and flexible analysis/manipulation tool available in any language. |
| NumPy [65] | • Package for array computing.<br>• Provides a powerful N-dimensional array object.<br>• Can also be used as an efficient multi-dimensional container of generic data. |
| SciPy [80] | • Works with NumPy arrays.<br>• Provides efficient routines for numerical integration and optimization. |
| Statsmodels [87] | • Complement to SciPy<br>• Used for statistical computations (descriptive statistics and estimation and inference for statistical models). |
| Keras [17] | • Library for constructing neural networks.<br>• Includes functionalities such as neural layers, activation and cost functions, objectives, batch normalization, dropout, and pooling. |
| TensorFlow [90] | • Library for high performance numerical computation.<br>• With strong support for machine learning and deep learning.<br>• Allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. |

| Package/Library | Description |
|---|---|
| Scikit-learn (sklearn) [17] | • Includes easy integration with different ML programming libraries like NumPy and Pandas.<br>• Supports various algorithms of Classification, Regression, Clustering, Dimensionality Reduction, Model Selection and Preprocessing. |
| Matplotlib [59] | • Visualization utility<br>• Useful for creating static, animated, and interactive graphs. |
| Pickle [75] | • Implements binary protocols for serializing and de-serializing a Python object structure.<br>• Useful for saving a model as a serialized object. |

**Table 3**. Useful Python's libraries and packages for Machine Learning [36]

### 3.3 Selection of Non-Imposed Technologies

After the previous systematic review, this subsection details how the different technologies will be selected according to some reasoning that has led to that decision.

First, Python will be selected because of the collection of scientific libraries and packages specifically designed for developing ML, and for being a high level and friendly programming language (higher level of abstraction from machine languages, easy to learn and understand, find error and debugs, etc.). Additionally, Python code can be easily run on Jupyter Notebooks, which will be selected for this purpose because it offers a user-friendly interface that is well suited for most data analytic work and visualization [93].

Secondly, from the unsupervised algorithms, five of them are considered for the process of anomaly detection. First, K-means, because it is very efficient and easy to implement compared to other clustering algorithms [54], and it only requires one parameter (k), which can be estimated with techniques such as Elbow Method and Silhouette Coeficient. However, as reviewed, the main drawback of K-means is that it does not behave very well when the clusters have varying sizes, different densities, or non-spherical shapes.

For this reason, Gaussian Mixture Models will be evaluated as well. This algorithm maximizes only the likelihood of an observation being part of a group and, will not force the clusters to have specific structures. Also, GMM is highly effective when a good estimate of the number of components in the mixture is available [77].

Additionally, Isolation Forest will be considered because it offers an alternative to the previous clustering algorithms and is optimized for anomaly detection. Moreover, due to its fast execution and low memory requirement [55].

The autoencoder will also be tested against the data to find anomalies. Assumed that more than 99% of observations are normal, the AE will learn to represent these data. So, when an outlier arises, the neural network will not be able to reconstruct it well, thus helping to detect anomalies when they occur.

Finally, running PCA prior to K-means or GMM algorithms can alleviate the "curse of dimensionality" problem and speed up the computations [62].

Each of these algorithms can be implemented using Python's library sklearn which makes it possible the design, train and test them. After that, results (detected anomalies) will be evaluated with the organization's security analysts and data lake experts aid, and one model will be chosen to be deployed.

### 3.4 Preliminary System Architecture

This section presents the design of the preliminary system architecture, considering all selected algorithms.

As introduced earlier, the system to be deployed presents three differentiated parts: the preprocessing of data, the ML model, and the prediction of a score that indicates whether the access is anomalous. However, initially, different solutions have been explored that could be applied for the anomaly detection problem. Thus, Figure 20 shows the preliminary system, which shows the way components would interact if different technologies were selected. Each internal component is responsible for one clear aim within the larger component: the fraud manager.

Inside the fraud manager, two main components are presented: preprocessing and prediction. The first one consists of four others, which represent different stages of data preprocessing. At the beginning, data flows into the feature engineering component, which transforms primary selected fields into new features and passes them to the normalization and standardization components. Depending on the algorithm/model, different preprocessing is required. For instance, two of the explored solutions (k-means and GMM) require dimension reduction for better performance, which is performed by the PCA component.

The prediction component encompasses four subcomponents that correspond to the unsupervised algorithms that have been considered to identify anomalous data: autoencoder, isolation forest, k-means and gaussian mixture model. After their evaluation, one of them will be selected and deployed, resulting in the simplified final architecture that is presented in the next section.

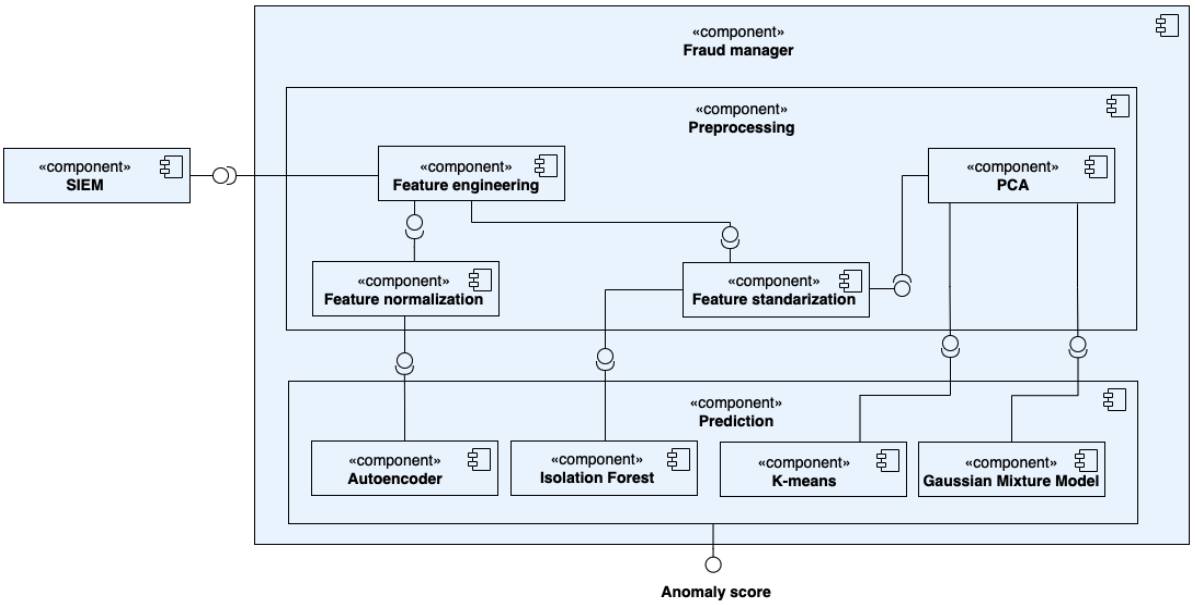



**Figure 20**. Preliminary component diagram

## 3.5 High-Level Architecture

This section presents the actual system under development, which relies on a single model, the autoencoder, whose selection will be justified in later chapters. Figure 21 shows the final component diagram.

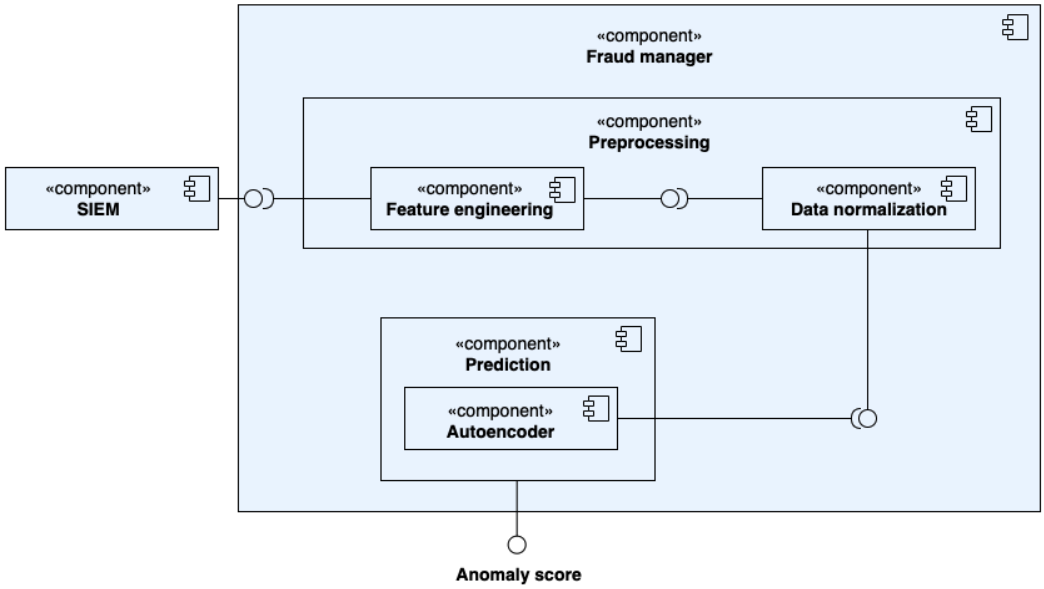



**Figure 21**. Final component diagram

On the left-hand side, the fraud manager waits for the SIEM (external component) to send the data. This information then flows into the preprocessing component, which, this time, is comprised of two components: feature engineering and normalization, since the autoencoder does not require any other kind of preprocessing. Finally, a score is calculated by the prediction component, which only encompasses the autoencoder subcomponent.

## 3.6 Use Cases

This section shows the application use case diagram along with the textual definition of each of the use cases present in the diagram. The study of the described use cases will facilitate the extraction of requirements in later phases of the project.

### 3.6.1 Use cases diagram

Figure 22 shows the diagram with the identified use cases: monitoring the production model and checking the threat score that the fraud manager returns.

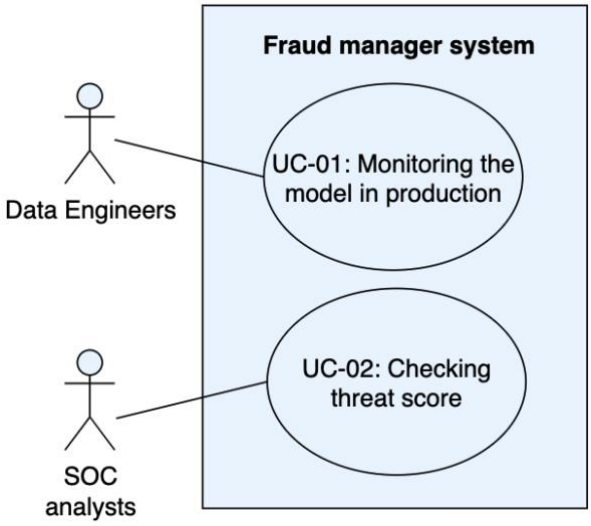



**Figure 22.** Use cases diagram

### 3.6.2 Textual definition of the use cases

This subsection presents detailed information about the identified use cases in the previous diagram.

Table 4 shows the first identified use case, which consists in detecting problems with the model and the system serving the model in production before it starts to generate negative business value.

| **Identifier: UC-01** | |
|---|---|
| **Use case name** | Monitoring the model in production |
| **Author** | Marina Boyero Torrente |
| **Use case overview** | Verifying that the system is responding and performs according to the expected benchmark. |
| **Actors** | Data Engineers |
| **Precondition** | The application is running. |
| **Post-condition** | The application returns an enriched log with the original fields, generated features and anomaly score for each access to the data lake. |

| Identifier: UC-01 | |
|---|---|
| **Basic path** | 1. Check for every log received a new (enriched) log is sent.<br>2. Check the score matches the risk of the access. |
| **Alternative path** | • The application is not receiving/sending data from/to the SIEM.<br>1a. The Docker container has stopped.<br>2a. Run Docker container.<br>1b. The code has raised an error.<br>2b. Look for the error and fix it.<br>• The score does not match the risk of the access.<br>2a. Some variable has become unavailable<br>3a. The model needs to be retrained and redeployed without that feature.<br>2b. Data used to train the model in the research environment comes different from the data streamed into the system.<br>3b. The model has to be trained with adequate data format.<br>2c. Data used to train the model does not represent the data that the system is receiving.<br>3c. The model needs to be retrained with up-to-date data.<br>2d. The selected features are not good enough.<br>3d. The model has to be retrained with new features. |

**Table 4**. Textual definition of use case UC-01

Table 5 shows the second identified use case, in which the SOC analyst checks the score delivered by the system to react accordingly.

| Identifier: UC-02 | |
|---|---|
| **Use case name** | Checking threat score |
| **Author** | Marina Boyero Torrente |
| **Use case overview** | Check anomaly score as predicted by the model. |
| **Actors** | SOC analysts |
| **Precondition** | The application sends a new piece of data to the SIEM. |
| **Post-condition** | The piece of data contains a score associated to the access. |
| **Basic path** | 1. The analyst checks the score of an access.<br>2. If needed, the SOC specialist checks other variables of the log.<br>3. The analyst takes action according to the score. |
| **Alternative path** | • The analyst cannot find some variable.<br>1a. Check if the SIEM's table is the one that should have that information. |

**Table 5**. Textual definition of use case UC-02

## 3.7 Software Requirements

Software requirements that are necessary for achieving the project's goals are identified in this section. Table 6 gathers different types of requirements as indicated by the IEEE 830 standard [10][60][92].

| Software Requirements | | | |
|---|---|---|---|
| **Type**: **Functional** | | | |
| **Subtype**: System Interfaces | | | |
| **Id** | **Name** | **Description** | **Priority** |
| FR-01 | SIEM connection | The system connects to the SIEM. | High |
| FR-02 | Event transformation | The system transforms events (extracts desired features and normalizes data) before passing it to the model. | High |
| FR-03 | Score prediction | The model predicts a score of -1 if an access is anomalous, otherwise the assigned score is 1. | High |
| FR-04 | Event delivery | The system sends the event back to the SIEM, with information about the predicted score for that access. | High |
| **Subtype:** Software Interfaces | | | |
| **Id** | **Name** | **Description** | **Priority** |
| FR-05 | Error messages | If an exception occurs during the execution of the system, it will be handled to display an error message. | Medium |
| **Type: Non- Functional** | | | |
| **Subtype:** External Interfaces | | | |
| **Id** | **Name** | **Description** | **Priority** |
| NFR-01 | System-SIEM communication | The system communicates with the SIEM through its Client API. | Medium |
| **Subtype:** Performance Requirements | | | |
| **Id** | **Name** | **Description** | **Priority** |
| NFR-02 | Event reception | The system receives events from the SIEM in near real time. | Medium |
| NFR-03 | Quick response time | The system must read the event and compute a score in near real time, with little to no delays. | Medium |
| NFR-04 | Capacity | The system must be able to process all accesses that are retrieved from the SIEM, without missing events. | High |

| **Subtype:** Design constraints | | | |
|---|---|---|---|
| **Id** | **Name** | **Description** | **Priority** |
| NFR-05 | Event format | A log that is sent/received to/from the SIEM must be formatted in JSON. | High |
| NFR-06 | Unprocessed log schema | Logs received from the SIEM (not yet processed) must present the following lists of attributes, displayed in this order:<br>• EVENT_TIMESTAMP<br>• OS_USERNAME<br>• DBUSERNAME<br>• DBPROXY_USERNAME<br>• CLIENT_PROGRAM_NAME<br>• USERHOST<br>• RETURN_CODE | High |
| NFR-07 | EVENT_TIMESTAMP format | Date in UNIX format of the user whose access was audited. | High |
| NFR-08 | OS_USERNAME format | Operating system login username of the user whose access was audited.<br>Datatype: VARCHAR2(30). | High |
| NFR-09 | DBUSERNAME format | Data base login username of the user whose access was audited.<br>Datatype: VARCHAR2(30). | High |
| NFR-10 | DBPROXY_USERNAME format | Proxying username, in the case of proxy authentication. The value is empty if user proxying is not in effect.<br>Datatype: VARCHAR2(30). | High |
| NFR-11 | CLIENT_PROGRAM_NAME format | Name of the application or program from which the user accessed.<br>Datatype: VARCHAR2(48). | High |
| NFR-12 | USERHOST format | Name of the host machine from which the session was spawned.<br>Datatype: VARCHAR2(128). | High |
| NFR-13 | RETURN_CODE format | Oracle error code generated by the action. Zero if the action succeeded.<br>Datatype: NUMBER. | High |

| **Subtype:** Design constraints | | | |
|---|---|---|---|
| **Id** | **Name** | **Description** | **Priority** |
| NFR-14 | Processed log schema | Processed logs that are sent back to the SIEM must contain information about the original fields (previously listed), generated features (resulting from feature engineering) and the computed score that indicates whether it is anomalous. | High |
| **Subtype:** Software system attributes | | | |
| **Id** | **Name** | **Description** | **Priority** |
| NFR-15 | Maintainability[5] | The model must be monitored and retrained periodically with new data in order to keep it up to date. | High |
| NFR-16 | Portability | To be able to replicate the behavior of the application irrespective of the host and/or operating systems on which it's running. | High |

**Table 6**. Software requirements

[5] this requirement is especially relevant because fraudulent behavior evolves quickly over time and fraudsters are constantly developing their strategies to avoid being detected [98]. This implies that machine learning models need to be updated and take into consideration these changes to be valid or adequate [98]

### 3.8 Acceptance Test Plan Design

Once the requirements have been specified, in this section, the acceptance test plan is designed. The system must meet all designed tests to be considered adequate, and, if it does not pass one, an action will be taken to recover the error and pass the test.

The acceptance test plan presented in Table 7 consists of a set of checks that need to be passed before considering the system is correctly deployed. Each test indicates the corresponding software requirements being verified, the input to the system, the expected result to that input, and the (recovery) action to take if the test does not pass. Finally, they are presented in the same order they should be carried out and they should also be tested after the deployment stage.

| Acceptance test plan | | | | |
|---|---|---|---|---|
| **Id** | **Test element** | **Input** | **Expected output** | **Recovery action** |
| AT-01 | FR-01 | Monitored accesses entering the SIEM. | System receives those accesses. | If logs are not being sent from the SIEM, check if SIEM-system connection is lost (AT-03). |
| AT-02 | FR-01 | Logs flowing into the system. | Logs flowing back to the SIEM. | If logs are not being sent back to the SIEM, check whether SIEM-system connection is lost (AT-03). |
| AT-03 | FR-01, FR-05 | Connection between system and SIEM fails. | Error message indicating 'time out error connection'. | If this error is not displayed, check exceptions in the code that do not raise an alert when this occurs. |
| AT-04 | FR-02 | The system receives a log from the SIEM. | The log is processed before being passed to the model. | If it's not processed (feature extraction + normalization), properly call these functions and check they work as they are supposed to. |
| AT-05 | FR-03 | The processed log is passed to the model. | Model gives a score of 1 or -1. | If a different score is delivered, the part of the code that entails prediction must be revised. |
| AT-06 | FR-04 | System sends the enriched log. | SIEM received the log, expressed by the original variables, but also containing information about the new features and score. | If logs are not being sent back to the SIEM, check whether SIEM-system connection is lost (AT-03). |

**Table 7**. Acceptance test plan

# 4. DETAILED DESIGN

This chapter shows class diagrams for each of the components defined in the previous section. Additionally, a set of sequence diagrams are shown to understand the different interactions that take place between the different components and classes of the system.

## 4.1 Software Design

This section provides a description of the components identified in the high-level architecture in Figure 21. Similarly, Figure 23 presents the final system architecture and the different components identified with a number, corresponding to the subsection in which they are detailed. Although it is essential, because the SIEM component has not been implemented in this project, it will not be explained in detail.

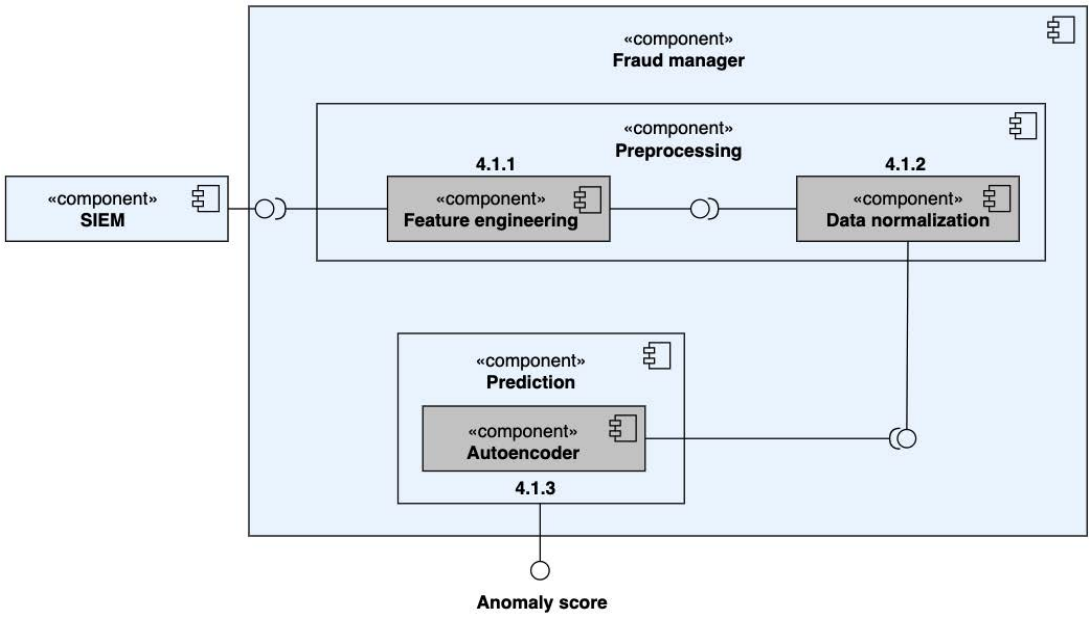



**Figure 23**. Identified components in final component diagram

In this case, the main functionality of the fraud manager component would be returning a score every time someone accesses the organization's data lake, as presented in the following class diagram.

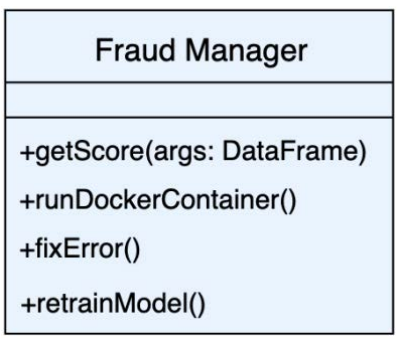



**Figure 24**. Fraud Manager class diagram

Next sections show class diagrams for each subcomponent. They are all very simple, as each of them performs only one task that allows getting the final score (delivered by the fraud manager).

### 4.1.1 Feature engineering component

This component is responsible for preprocessing a new log that comes from the SIEM. To do so, it must receive the log described by the fields: EVENT_TIMESTAMP, OS_USERNAME, DBUSERNAME, DBPROXY_USERNAME, CLIENT_PROGRAM_NAME, USERHOST and RETURN_CODE. As a result, it generates the appropriate features for the model. Its class diagram is presented in

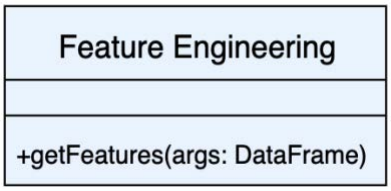



**Figure 25**. Feature Engineering class diagram

### 4.1.2 Data normalization component

Before passing the features to the model, data must be normalized to the range [0,1], which is performed by this component.

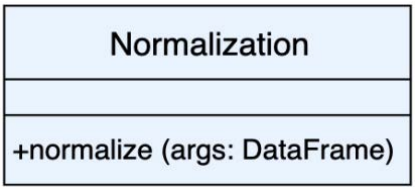



**Figure 26**. Data normalization class diagram

### 4.1.3 Autoencoder component

The autoencoder component receives normalized features and predicts reconstruction error of the access log in order to determine if it's anomalous, in which case it assigns a -1 to it.

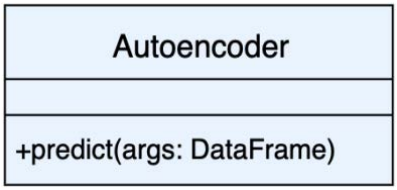



**Figure 27**. Autoencoder class diagram

### 4.2 Sequence Diagram

In this section, interactions between the user and the (fraud manager) system are described. The SIEM takes an important role as well, as it acts as a linkage between the user and the system. For representing these interactions, use cases defined in section 3.6 are displayed through sequence diagrams and, both, basic and alternative paths are represented.

#### 4.2.1 Monitoring the model in production (UC-01)

This subsection presents the sequence of actions for UC-01: Monitoring the model in production, which is represented in Figure 28.

The monitoring process starts when the data engineer (actor) checks that for one or more access logs (contained in the SIEM), a new (enriched) log is also there or being sent by the fraud manager. This could be done for either checking the system is receiving and sending back data, or to check the delivered score.

Alternatively, it could happen that the application is not working properly because the Docker container has stopped, an error has been raised or the model is outdated and needs to be retrained.

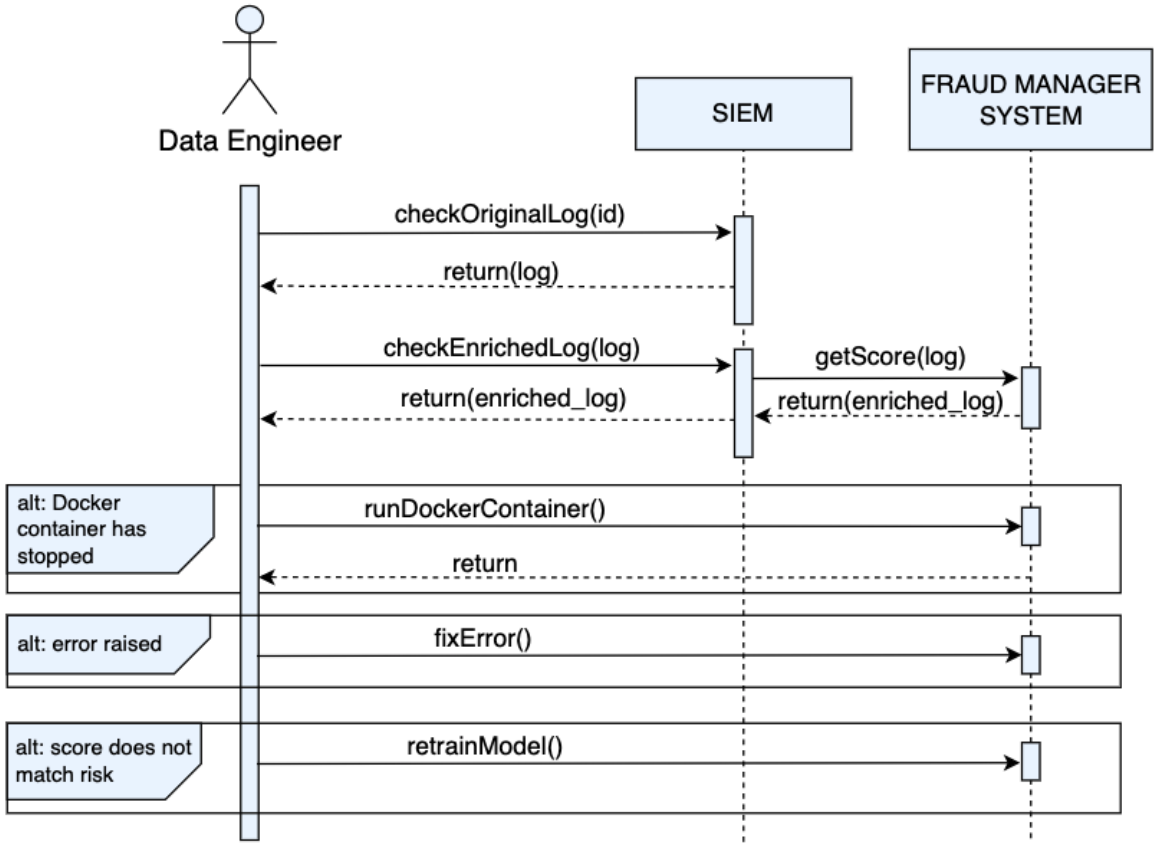



**Figure 28**. Sequence diagram: monitoring the model in production (UC-01)

### 4.2.2 Checking threat score (UC-02)

This subsection presents the sequence of actions for UC-02: checking threat score.

As exhibited in Figure 29, the SOC analyst starts this process to obtain the score associated to an event. It could happen that, for assessing its value, they want to check other variables of the log for understanding the context of the score. If the variable is not found, it is probable that the analyst is checking in the wrong table, so they should change to the one in which logs are being processed.

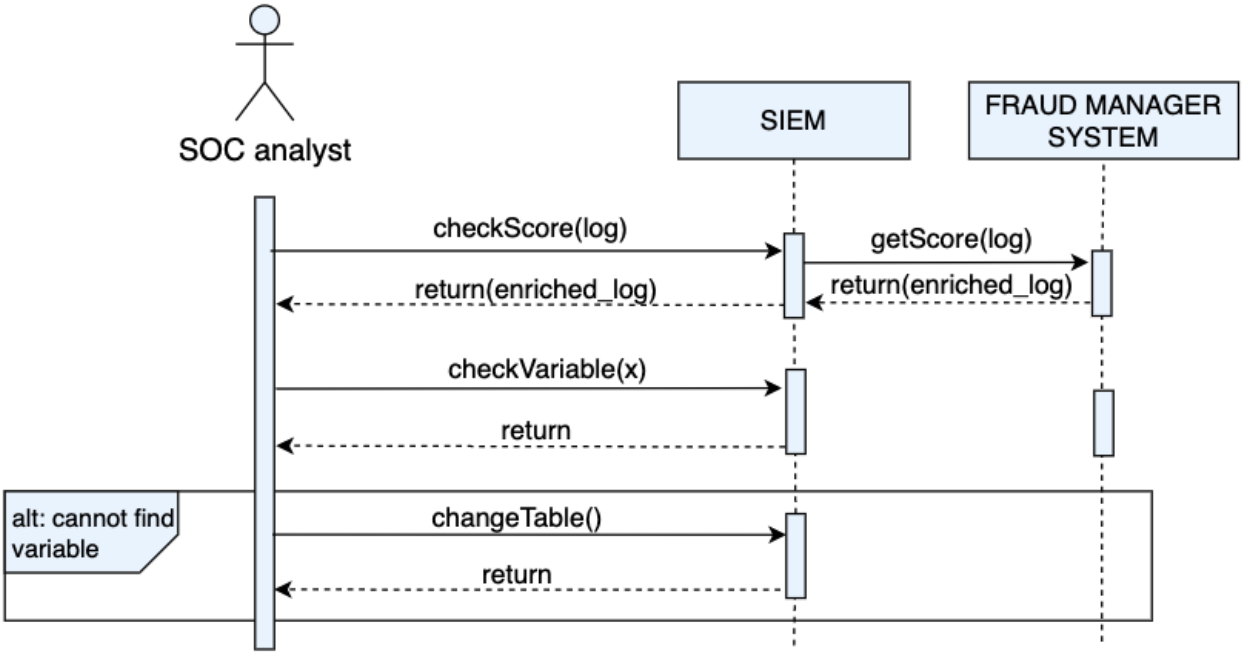



**Figure 29**. Sequence diagram: checking threat score (UC-02)

# 5. SOFTWARE DEPLOYMENT

In this chapter the most remarkable decisions of the implementation are presented. Additionally, the acceptance test plan is designed.

## 5.1 Deployment Decisions

This section details the decisions that have been made regarding the initial design and final deployment of the machine learning model to production.

### 5.1.1 Solution Approach

Before properly diving into model deployment, it was necessary to explore the accessible resources for the stated problem: detecting anomalous accesses to the organization's data lake.

Exploring available data derived into selecting an unsupervised learning approach. The main reason was that there was not prior knowledge of what was considered abnormal, and, even if it had been informed, the amount of data labeled as "normal" would still be greater than anomalies, thus resulting in an imbalanced dataset.

Once the approach was decided, a range of unsupervised machine learning solutions were studied and, finally, it was decided that four of them would be designed and analyzed before deciding which one to productionalize. They were a deep learning network, an ensemble model of isolation trees, and a pair of clustering solutions.

### 5.1.2 Feature Engineering

During the execution of the project, most of the time has been dedicated to the design of the features for feeding the model, known as feature engineering. The main purpose of features is to identify and provide the model with useful information. In this scenario where the goal was to find observations deviating from the majority, features must capture information about trends and patterns in data.

It was decided that it would be overcome with exploratory data analysis (EDA), which makes it possible to understand patterns within the data, detect outliers or anomalous events and find interesting relations among the variables. Precisely, statistical univariate and bivariate visualization techniques (such as boxplots or contingency tables) provided a summary of anomaly indicators. According to these statistics, features have been created.

### 5.1.3 Model Selection

Among the designed machine learning solutions, one had to be selected for its deployment, and this selection had to be done according to some performance indicator. For this purpose, it is usually necessary to have a labeled dataset, which, as specified before, was not available at the beginning of the project. To overcome this issue, it was finally decided to build a labeled dataset for evaluating the model based on anomalies that were detected during the EDA and normal instances provided by expert knowledge from the organization.

Following the imbalanced nature of the problem, the testing dataset contained 99.85% of normal instances and only 0.15% of anomalies.

### 5.1.4 Model Explainability

In many safety-critical domains there may be some major risks if anomaly detection models are directly used as black-box models [71]. For example, rare data instances reported as anomalies may lead to possible algorithmic bias against users that are under-presented in the data. Therefore, to mitigate this type of risk and provide explanation of why a specific access is identified as anomalous, it has been decided to provide such explanation through a reporting function that delivers straightforward clues to the SOC analysts.

This function receives two arguments: data (explained by some variables) that has been passed to the algorithm/model, and the labels that this algorithm/model has assigned. Based on this information, the function outputs a set of rules that describe an instance based on the values its variables take. For example, in k-means, all instances that share some low and upper values in their variables would be grouped together and this information would be outputted by the explainability function.

As this functionality is based on the implementation of a decision tree, it has to be considered that the variables that are passed to the function are numerical. Not other type would be valid. For this reason, it is suggested that, when trying to obtain insights about the classification of an instance, data is passed to this explainability function described by the generated features, which are numerical and have been designed to look into particular hints that point out abnormality.

### 5.1.5 Model Productionization

One of the main requirements for the solution to be deployed is quick response time, so there are two major decisions that had to be made for productionalizing the model. The first one was related to SIEM communication, which is necessary for both, retrieving and delivering data. For these operations to perform simultaneously at a considerable speed, a thread and mutex solution has been proposed. Thus, one thread would acquire the logs coming from the SIEM and save them to a global queue variable, while the second thread would deque one by one, transform them, predict their anomaly score and send them back to the SIEM.

### 5.2 Acceptance Test Plan Results

This section shows the results of the acceptance tests defined in section 3.8. As indicated in Table 8, all defined acceptance tests have passed, so the deployed model is considered adequate. However, during the deployment stage, these tests must be checked periodically to ensure its validity.

| Acceptance test plan[6] | | |
|---|---|---|
| **Id** | **Tested element** | **Output** |
| AT-01 | FR-01 | pass |
| AT-02 | FR-01 | pass |
| AT-03 | FR-01, FR-05 | pass |
| AT-04 | FR-02 | pass |
| AT-05 | FR-03 | pass |
| AT-06 | FR-04 | pass |

**Table 8**. Acceptance test plan results

[6] executed during the design stage and after the first deployment of the model, however, it has to be monitored periodically and these tests must be checked during the whole live of the system

# 6. EXPERIMENTATION AND RESULTS

This chapter includes the experiments created to test the goodness of the designed models. Results include a comparison between them and a justification of which one to select, considering the anomalies detected and organization's considerations. Additionally, it shows the results of the acceptance test plan.

## 6.1 Experimentation

This section presents the experiments that have been designed for evaluating and benchmarking the goodness of the anomaly detection models.

In this setting, there is no previous information (or labelling) about what normal or abnormal accesses are, and, as opposed to traditional detection mechanisms that attempt to identify fraud based on rules, there are no predetermined normalcy values. Thus, experiments are generated consistent with the definition of anomalies as "instances that significantly deviate from the majority of data instances" [67]. In other words, experiments displayed in Table 9 are log entries whose characteristics diverge from the characteristics and distribution of the majority of the training dataset.

| **Abnormal accesses** **(obtained from exploratory data analysis)** | **Training data characteristics** |
|---|---|
| Accesses from applications whose length is longer than 25. | It's been analyzed in the training dataset that program names are normally between 1 and 25 characters long. |
| Accesses from applications that contain more than 2 punctuation marks. | In the training dataset, program names often don't present more than 2 punctuation marks. |
| Accesses performed with OS usernames whose length is longer than 12 or shorter than 6. | OS usernames in the training dataset tend to be 7-10 characters long. |
| Accesses performed with DB usernames whose length is longer than 29 or shorter than 3. | DB usernames in the training dataset tend to be 6-12 characters long. |
| Accesses whose DB and OS usernames are nominal ("3o") but don't match. | 80% of accesses performed by nominal users have the same value in OS_USERNAME and DB_USERNAME. |
| Accesses whose proxy and OS usernames are nominal ("3o") but don't match. | 87% of accesses performed by nominal users have the same value in OS_USERNAME and DBPROXY_USERNAME. |
| Accesses through a class B IP whose length is shorter than 12. | Most of class B IPs have a length between 12 and 14. |
| Accesses through a class C IP whose length is different than 13. | Normally, class C hostnames have a length of 13. |

| Abnormal accesses (obtained from exploratory data analysis) | Training data characteristics |
|---|---|
| Accesses performed on Thursday and Saturday at 3 am. | Most accesses are performed between 8-12 pm during the week, and the frequency is lower during the weekends. Traffic is especially low on Thursday and Saturday at 3 am. |

**Table 9**. Examples of abnormal accesses according to training data

One could think that accesses that don't present the above characteristics should be considered "normal", however, there could exist more complex relations between data variables that have not been detected during data analysis and could be descriptive of abnormal behavior. Instead, for testing the models, normal cases have been provided by organization's experts. Table 10 shows some traces of these normal accesses, although a bigger dataset has been created with more traces.

| Normal accesses | | | | | | |
|---|---|---|---|---|---|---|
| EVENT_ TIMESTAMP | OS_ USERNAME | DB USERNAME | DBPROXY_ USERNAME | CLIENT_ PROGRAM _NAME | USER HOST | RETURN _CODE |
| 1630592460 | 3oe8eger | IKDEARQ1 | 3oe8eger | python | 10.23.20.86 | 0 |
| 1630592460 | 3oepexgf | 3oepexgf | | sas | 10.18.0.114 | 0 |
| 1630593840 | SENSECO RPSRV | PBBISFC1 | | QvConnec t64.EXE | 10.22.98.74 | 0 |
| 1630912980 | Pdtping1 | IDTPING1 | | JDBC Thin Client | 10.18.5.27 | 0 |
| 1630913280 | 3oepgops | ECI_FUS | AECIFUS1 | SQL Developer | 10.148.11.69 | 0 |
| 1630913760 | neolane | OOBIGIN1 | | nlserver | 10.119.5.106 | 0 |
| 1630914360 | 3oe8oxep | ELE_CLD | | sqldevelop er64W.exe | 10.148.11.20 2 | 0 |
| 1630914720 | root | ITGAMDT 5 | | DDTEK ODBC Oracle | 10.241.13.17 4 | 0 |
| 1632226080 | 3oeprpts | ELE_TLE | AELETLE1 | Toad.exe | 10.148.11.13 6 | 0 |

**Table 10**. Examples of normal accesses according to expert knowledge

The final test dataset for assessing the goodness of the models is created with 99.85% normal accesses and 0.15% abnormal ones.

### 6.2 Results Obtained

In this section results are analyzed according to different performance metrics, and, finally, one of the designed models is selected. However, it must be taken into consideration that the organization's experts have evaluated other aspects to consider this model appropriate, that cannot be revealed due to agreed confidentiality.

To evaluate how well each model detects anomalies, they are compared according to different performance metrics that are presented in Table 11.

| **K-means performance metrics** | | | | | |
|---|---|---|---|---|---|
| **Accuracy** | **Recall** | **Specificity** | **Precision** | **F1-Score** | **AUC** |
| 0.693 | 0.358 | 0.640 | 0.002 | 0.003 | 0.449 |
| **GMM performance metrics** | | | | | |
| **Accuracy** | **Recall** | **Specificity** | **Precision** | **F1-Score** | **AUC** |
| 0.996 | 0.150 | 0.997 | 0.078 | 0.103 | 0.573 |
| **Isolation forest metrics** | | | | | |
| **Accuracy** | **Recall** | **Specificity** | **Precision** | **F1-Score** | **AUC** |
| 0.994 | 0.098 | 0.996 | 0.034 | 0.050 | 0.547 |
| **Autoencoder performance metrics** | | | | | |
| **Accuracy** | **Recall** | **Specificity** | **Precision** | **F1-Score** | **AUC** |
| 0.996 | 0.171 | 0.997 | 0.080 | 0.109 | 0.584 |

**Table 11.** Benchmarking of models according to a series of performing metrics

Looking into the accuracy metric, it indicates the fraction of predictions the model got right, so one could think the Gaussian Mixture Model or autoencoder are the best anomaly detectors, closely followed by the isolation forest model. However, the main issue with the accuracy metric is that is not reliable for models trained on imbalanced datasets, which is the case. So, in this scenario, this metric will not be considered unless it equals 1, because predicting all instances as “normal” would already give an accuracy of 0.999.

Recall (or sensitivity), on the other hand, could be a good indicator in this scenario, as it reveals the proportion of anomalies that are correctly predicted by the model. This metric is important for the purpose of the project whose aim is to detect abnormality. According to this metric, K-means model seems to outperform the others.

Similarly, specificity shows the proportion of normal instances correctly classified. Although this metric is important, it is not as relevant as the previous one, because a lower value would only indicate some normal instances are being classified as anomalous (false positives), which is preferable over anomalous instances being classified as normal, therefore allowing those potentially fraudulent accesses to the DL.

Additionally, precision indicates the proportion of real anomalies detected out of all the instances that have been predicted as anomalous, and results very useful when classes are imbalanced. Considering this metric, none of the models have good precision, but the autoencoder or GMM take the highest values in this metric.

Since both, precision and recall, should be optimized in this scenario, looking into the F1-score (a metric that gives equal weight to precision and recall) will give a hint of the better model. According to it, again the autoencoder and GMM present better results, although they are far from optimal.

Finally, but as important, in this scenario of imbalanced training dataset, AUC (area under ROC curve) is presented as a robust metric. According to [39], the following rule of thumb can be followed to assess the goodness of the model:

| | |
|---|---|
| 0.5 | No discrimination |
| 0.5-0.7 | Poor discrimination |
| 0.7-0.8 | Acceptable discrimination |
| 0.8-0.9 | Excellent discrimination |
| >0.9 | Outstanding discrimination |

According to the AUC and other performance metrics, none of the models seem optimal discriminators. However, it must be highlighted that they have been tested and benchmarked on a dataset that relies on labels that are probably partial or incomplete (they do not span the entire set of anomaly class), inexact or inaccurate (some given labels can be incorrect). Furthermore, anomaly detection presents some underlying complexities and unsolved challenges that add uncertainty to this task, as will be reviewed in the following chapter. Keeping this in mind, among the presented models, the autoencoder offers a slightly better performance for the task of anomaly detection and that is why it will be deployed. Nevertheless, in the next section, some recommendations are provided to extend and improve this solution.

# 7. CONCLUSIONS AND FUTURE WORK

This section presents the main difficulties and personal conclusions obtained from the development of the project, as well as some of the future lines to be explored to improve the results obtained, as well as alternative ways of doing it.

In general, the project has been considered successful, as it proposes a first approach for automatically detecting anomalous accesses to data lakes, which had not been contemplated before. Additionally, it results beneficial for SOC analysts and organization as a whole, who previously relied on a rigid system of rules for controlling data lake access.

Nevertheless, the performance of the model is far from being perfect, prove of it are the performance indicators that have been obtained. For that reason, rather than being solely utilized, it should be considered complementary to the existing misuse approach. In fact, before this solution is good enough to be blindly trusted, certain improvements and some challenges must be addressed.

## 7.1. Project difficulties

The most relevant difficulties encountered during the execution of this project seem to be related to the inherent and unsolved complexities anomaly detection presents [71]. The first one is that, although a variety of anomaly detection methods have been introduced over the years, the current state-of-the-art unsupervised methods like the one presented in this work, still incur high false positive rates on real-world datasets [71]. How to reduce false positives and enhance detection recall rates is “one of the most important and yet difficult challenges”, considering the significant cost of missing real anomalies [71].

Secondly, anomalies are associated with the unknown and are only discovered when they occur [71]. This problem is bigger in high-dimensional spaces [71], like the one presented in this setting where anomalies become unnoticeable. Thus, identifying intricate variable interactions through feature selection is vital and, still, remains a major challenge [71].

Lastly, other difficulties have been related to the initial ignorance of some of the technologies that are needed for the project, such as Docker, which I never worked with before. Additionally, some of the proposed machine learning algorithms were new for me because, at that point, I still hadn't taken some related modules at university. As a result, a lot of knowledge has been acquired but, in turn, sometimes it has been tough to move forward.

## 7.2 Personal conclusions

During the execution of this project, some personal circumstances have turned it into a real challenge.

In the first place, the work that has been described has been part of a bigger project in a national organization located in Barcelona, so I had to move there, and, during this time, I've been living on my own. On top of that, I've had to keep up with lessons at university, which took place in Madrid, and frequently travel between these two locations due to exams or personal circumstances.

Nevertheless, the realization of this project has given me valuable experiences, and it's important to me because I have sharpened skills such as time management, persistence, and capability to face adversities, qualities that have been vital for the adequate execution of this project.

The task presents an important research aspect as well, which has been very interesting and rewarding, being able to see the process between the detection of a problem or need and the actual development of the proposed solution.

Finally, this project has given me some notions of cybersecurity and why it's important for any business to control who accesses delicate data; it has allowed me to work surrounded by great professionals from who I have learned a lot of what I now know; and it is given me a new perspective on how two different research areas such as cyber security and machine learning come together to solve a unique task.

## 7.3 Future Work

This section proposes some extensions and alternatives that could be further investigated. Most of them have been identified throughout the design and deployment of the model and are mainly focused on improving the detection of fraudulent accesses.

The first one would be related to the training dataset. Due to technological limitations, this model is built upon one month of data, however, ideally, the model must be trained on a longer timeline, based on accesses performed to the data lake during the last couple of years, for instance.

Secondly, since the model's ability to identify abnormal behavior depends on features' capability to represent nonlinear and complex relations between data variables, selecting different features or creating a subset of new one that enriches the current features should be addressed.

Another design that was studied for the present work was to assemble a labeled dataset, aided by expert's knowledge, to tackle the problem as a fully supervised anomaly detection task. Due to the difficulty and cost of collecting large-scale labeled anomaly data, this solution was considered impractical, however it could be explored in greater depth.

Moreover, taking advantage of the different models that have been implemented (isolation forest, k-means, GMM and autoencoder), future research could combine them to form a composite predictor. This idea of building a prediction model by combining the strengths of a set weak learners is known on research world as ensemble learning and comes as an additional proposal for improving the current one.

Finally, there is an alternative approach that may be parallelly explored which signals suspicious activity based on user's behavior. User Behavior Analytics (or UBA) builds user's profile based on their past activity and raises alerts when deviations from the usual patterns are detected. Therefore, this could result key for detecting when someone accesses the data lake with a different IP address or unusual timing, for instance.

After this revision, it can be concluded there is further scope for improvement in this area of research.

# REFERENCES


1. Abdallah, A., Maarof, M. A., & Zainal, A. (2016). Fraud detection system: A survey. Journal of Network and Computer Applications, 68, 90–113.
2. Ali, A. (2021). Audit Logs Management and Security-A Survey. *Kuwait Journal of Science*, *48*(3).
3. Amadeu, A. L., Vinturin, F., Morais, G. A. Z., Hubner, M., Pereira, E. M., & Santos, M. (2021, July). Machine Learning based Pricing Methodology for the Logistic Domain: a Preliminary Approach. In *Anais do XLVIII Seminário Integrado de Software e Hardware* (pp. 166-171). SBC.
4. Amat J. (December, 2020) Detección de anomalías con Gaussian Mixture Model (GMM) y Python. https://www.cienciadedatos.net/documentos/py23-deteccion-anomalias-gmm-python.html
5. Arnold, J. (2019, April 22). *What are the Pros and Cons of Using MATLAB Programming Software?* BookMyEssay - Academic Support Center Australia Help with Assignment Writing Services, Essay Writing, Homework Help, Nursing Assignment Help, Management Assignment Help, Case Studies, Dissertation Writing Help. https://www.bookmyessay.com/what-are-the-pros-and-cons-of-using-matlab-programming-software/
6. *Artificial Intelligent Algorithms – Towards Data Science*. (2021). [Image]. Retrieved 30 October 2021, from https://cdn-images-1.medium.com/max/1600/1*70yha0i9IwzxGxi-WHiw-w.png
7. *Autoencoder architecture*. (2020). [Picture]. https://sci2lab.github.io/ml_tutorial/autoencoder/
8. Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. *arXiv preprint arXiv:2003.05991*.
9. Barletta, V. S., Caivano, D., Nannavecchia, A., & Scalera, M. (2020). Intrusion detection for in-vehicle communication networks: An unsupervised kohonen som approach. *Future Internet*, *12*(7), 119.
10. Blanco, C., & Ruiz, F. (2011). *Ingeniería del Software. Requisitos.* [Diapositiva de PowerPoint]. Univ. Cantabria – Fac. de Ciencias. https://ocw.unican.es/pluginfile.php/1403/course/section/1792/is1_t04_requisitos.pdf
11. Boehmke, B., & Greenwell, B. (2019). *Hands-on machine learning with R*. Chapman and Hall/CRC.
12. Bonaccorso, G. (2017). *Machine learning algorithms*. Packt Publishing Ltd.
13. Borghesi, A., Bartolini, A., Lombardi, M., Milano, M., & Benini, L. (2019, July). Anomaly detection using autoencoders in high performance computing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 9428-9433).
14. Boudreau, E. (2021, December 14). *Should You Jump Python's Ship And Move To Julia? - Towards Data Science*. Medium. https://towardsdatascience.com/should-you-jump-pythons-ship-and-move-to-julia-ccd32e7d25d9

15. Chakraborty, S. (2020, April 20). *Using GoLang with Docker*. GoLang Docs. https://golangdocs.com/golang-docker

16. *Component Diagram Tutorial*. (2021). Lucidchart. https://www.lucidchart.com/pages/uml-component-diagram

17. Costa, C. D. (2021, December 13). *Best Python Libraries for Machine Learning and Deep Learning*. Medium. https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deep-learning-b0bd40c7e8c

18. *CTN 320 Ciberseguridad y protección de datos personales*. (2022, 22 junio). UNE 2022. https://revista.une.org/14/ctn-320-ciberseguridad-y-proteccion-de-datos-personales.html#:%7E:text=CTN%20320%20Ciberseguridad%20y%20protecci%C3%B3n%20de%20datos%20personales,Internet%20de%20las%20Cosas%20%28IoT%29%20o%20Big%20Data

19. DeepAI. (2020, June 25). *Feature Extraction*. https://deepai.org/machine-learning-glossary-and-terms/feature-extraction

20. Delua, J. D. (2021, March 12). *Supervised vs. Unsupervised Learning: What's the Difference?* IBM. Retrieved 26 October 2021, from https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

21. Derakhshannia, M., Gervet, C., Hajj-Hassan, H., Laurent, A., & Martin, A. (2020). Data lake governance: Towards a systemic and natural ecosystem analogy. *Future internet*, *12*(8), 126.

22. Dong, X., Yu, Z., Cao, W., Shi, Y., & Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science*, *14*(2), 241-258.

23. *DOUE-L-2001-80052 Reglamento (CE) nº 45/2001 del Parlamento Europeo y de Consejo, de 18 de diciembre de 2000, relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales por las instituciones y los organismos comunitarios y a la libre circulación de estos datos.* (2001). Agencia Estatal Boletín Oficial del Estado.

24. Elmrabit, N., Zhou, F., Li, F., & Zhou, H. (2020, June). Evaluation of machine learning algorithms for anomaly detection. In *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)* (pp. 1-8). IEEE.

25. E.M. Mirkes. (2011). *K-means and K-medoids applet.* University of Leicester. http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html

26. Fang, H. (2015, June). Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)* (pp. 820-824). IEEE.

27. Fong, J. (2018, September 1). *Are Containers Replacing Virtual Machines?* Docker Blog. https://www.docker.com/blog/containers-replacing-virtual-machines/

28. Gao, K., Mei, G., Piccialli, F., Cuomo, S., Tu, J., & Huo, Z. (2020). Julia language in machine learning: Algorithms, applications, and open issues. *Computer Science Review*, *37*, 100254.

29. *Gaussian mixture models*. (2022). Scikit-Learn. https://scikit-learn.org/stable/modules/mixture.html

30. *Gaussian Mixture Models Explained*. (2019). [Picture]. https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

31. GeeksforGeeks. (2019, May 20). *ML | DBSCAN reachability and connectivity*. https://www.geeksforgeeks.org/ml-dbscan-reachability-and-connectivity/

32. Giebler, C., Gröger, C., Hoos, E., Schwarz, H., & Mitschang, B. (2019, August). Leveraging the data lake: current state and challenges. In *International Conference on Big Data Analytics and Knowledge Discovery* (pp. 179-188). Springer, Cham.

33. Goel, V. (2017, February 21). *Verizon Will Pay $350 Million Less for Yahoo*. *The New York Times*. https://www.nytimes.com/2017/02/21/technology/verizon-will-pay-350-million-less-for-yahoo.html

34. Granlund, T., Stirbu, V., & Mikkonen, T. (2021). Towards regulatory-compliant MLOps: oravizio's journey from a machine learning experiment to a deployed certified medical product. *SN computer Science*, *2*(5), 1-14.

35. Gupta, M., Patwa, F., & Sandhu, R. (2018, March). An attribute-based access control model for secure big data processing in hadoop ecosystem. In *Proceedings of the Third ACM Workshop on Attribute-Based Access Control* (pp. 13-24).

36. Hao, J., & Ho, T. K. (2019). Machine learning made easy: a review of scikit-learn package in python programming language. *Journal of Educational and Behavioral Statistics*, *44*(3), 348-361.

37. Harrington, P. (2012). *Machine learning in action*. Simon and Schuster.

38. Hilal, W., Gadsden, S. A., & Yawney, J. (2021). A Review of Anomaly Detection Techniques and Applications in Financial Fraud. *Expert Systems with Applications*, 116429.

39. Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (Vol. 398). John Wiley & Sons.

40. Hurra, T. (2021, December 14). *DBSCAN — Make density-based clusters by hand - Towards Data Science*. Medium. https://towardsdatascience.com/dbscan-make-density-based-clusters-by-hand-2689dc335120#:%7E:text=Density%20Reachable%3A%20A%20point%20is%20called%20density%20reachable,are%20called%20density%20reachable%20from%20each%20other.%203

41. IEEE Standards Committee. (2021). IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021.

42. *I want to use C++ to learn Machine Learning instead of Python or R, is it fine?* (2021, March 13). Quora. https://www.quora.com/I-want-to-use-C++-to-learn-Machine-Learning-instead-of-Python-or-R-is-it-fine

43. Jiang, J., Chen, J., Gu, T., Choo, K. K. R., Liu, C., Yu, M., & Mohapatra, P. (2019, November). Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM)* (pp. 109-114). IEEE.

44. Kabir, M. A., & Luo, X. (2020, August). Unsupervised learning for network flow based anomaly detection in the era of deep learning. In *2020 IEEE Sixth*

*International Conference on Big Data Computing Service and Applications (BigDataService)* (pp. 165-168). IEEE.

45. Keating, G. (2021, February 16). *Data Lakes: What They Are and Why Companies Use Them*. Segment Blog. https://segment.com/blog/data-lakes/#why-do-companies-use-data-lakes
46. Khine, P. P., & Wang, Z. S. (2018). *Data lake: a new ideology in big data era. ITM Web of Conferences, 17, 03025.* doi:10.1051/itmconf/20181703025
47. Kohonen, T., & Honkela, T. (2007). Kohonen network. *Scholarpedia*, *2*(1), 1568.
48. Kriegel, H.-P., Kröger, P., Sander, J., & Zimek, A. (2011). *Density-based clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1(3), 231–240.* doi:10.1002/widm.30
49. Kurita, T. (2019). Principal component analysis (PCA). *Computer Vision: A Reference Guide*, 1-4.
50. Lantz, B. (2019). *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd.
51. Lassiter, A., & Darbari, M. (2020). Assessing alternative methods for unsupervised segmentation of urban vegetation in very high-resolution multispectral aerial imagery. *Plos one*, *15*(5), e0230856.
52. *Legal Statements*. (2022). Python.Org. https://www.python.org/about/legal/
53. Lewinson, E. (2021, August 26). *Outlier Detection with Isolation Forest - Towards Data Science*. Medium. https://towardsdatascience.com/outlier-detection-with-isolation-forest-3d190448d45e#:%7E:text=Isolation%20Forest%20is%20an%20outlier%20detection%20technique%20that,be%20scaled%20up%20to%20handle%20large%2C%20high-dimensional%20datasets.
54. Li, L. (2021, December 10). *K-means Clustering with scikit-learn - Towards Data Science*. Medium. https://towardsdatascience.com/K-means-clustering-with-scikit-learn-6b47a369a83c#:%7E:text=As%20we%20will%20see%2C%20the%20K-means%20algorithm%20is,algorithm%20belongs%20to%20the%20category%20of%20prototype-based%20clustering
55. Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413-422). IEEE.
56. Magomedov, S., Pavelyev, S., Ivanova, I., Dobrotvorsky, A., Khrestina, M., & Yusubaliev, T. (2018). Anomaly detection with machine learning and graph databases in fraud management. *International Journal of Advanced Computer Science and Applications*, *9*(11), 33-38.
57. Montague, B. (2008). Why you should use a BSD style license for your Open Source Project. *FreeBSD. org*.
58. Margulies, J. (2015). A developer's guide to audit logging. *IEEE Security & Privacy*, *13*(3), 84-86.
59. *matplotlib*. (2021, December 11). PyPI. https://pypi.org/project/matplotlib/
60. Mcgowan, C. (2012, November 5). *Non-Functional Requirements*. SlideServe. https://www.slideserve.com/chick/non-functional-requirements

61. Mohanur Jagadeesan, P. R. (2020). *A Framework Design to Improve and Evaluate the Performance of Security Operation Center (SOC)* (Doctoral dissertation, Dublin, National College of Ireland).
62. Murat, M. (2019). *Implementing K-means Clustering from Scratch - in Python*. Mustafa Murat ARAT. https://mmuratarat.github.io/2019-07-23/kmeans_from_scratch
63. Nathaniel, J. (2021, December 19). *Golang for Machine Learning? - Towards Data Science*. Medium. https://towardsdatascience.com/golang-for-machine-learning-bd4bb84594ee#:%7E:text=GoLearn%20is%20a%20machine%20learning%20library%20in%20GoLang,fitting%2C%20and%20even%20data%20splitting%20for%20training%2Fevaluation%20processes
64. Novikau, A. (2020, December 29). *Understanding Concurrency and Parallelism in Golang*. Spiral Scout. https://spiralscout.com/blog/understanding-concurrency-and-parallelism-in-golang
65. *numpy*. (2021, December 31). PyPI. https://pypi.org/project/numpy/
66. Omar, S., & Ngadi, A. H. jebur, h.(2013). *Machine Learning Techniques for Anomaly Detection: An Overview. International Journal of Computer Applications*, *79*(2), 33-41.
67. Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.
68. Park, D. C. (2000). Centroid neural network for unsupervised competitive learning. *IEEE Transactions on Neural Networks*, *11*(2), 520-528.
69. Paluszek, M., & Thomas, S. (2016). *MATLAB machine learning*. Apress.
70. *pandas*. (2021, December 12). PyPI. https://pypi.org/project/pandas/
71. Pang, G., Shen, C., Cao, L., & Hengel, A. V. D. (2021). Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, *54*(2), 1-38.
72. Pasupuleti, P., & Purra, B. S. (2015). *Data lake development with big data*. Packt Publishing Ltd.
73. Patel, P., Greg, W., Diaz, A. (2017). *Data lake governance best practices.* https://dzone.com/articles/data-lake-governance-best-practices
74. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, *12*, 2825-2830.
75. *pickle — Python object serialization*. (2022). Python 3.10.4 documentation. https://docs.python.org/3/library/pickle.html
76. Pourhabibi, T., Ong, K. L., Kam, B. H., & Boo, Y. L. (2020). Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, *133*, 113303.
77. Qiu, H., Eklund, N., Hu, X., Yan, W., & Iyer, N. (2008, June). Anomaly detection using data clustering and neural networks. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* (pp. 3627-3633). IEEE.

78. Ravat, F., & Zhao, Y. (2019, August). Data lakes: Trends and perspectives. In *International Conference on Database and Expert Systems Applications* (pp. 304-313). Springer, Cham.

79. Reddy, Y. C. A. P., Viswanath, P., & Reddy, B. E. (2018). Semi-supervised learning: A brief review. *Int. J. Eng. Technol*, *7*(1.8), 81.

80. *scipy*. (2021, November 24). PyPI. https://pypi.org/project/scipy/

81. Sensarma, D., & Sarma, S. S. (2015). A survey on different graph based anomaly detection techniques. *Indian Journal of Science and Technology*, *8*(31), 1-7.

82. Shakya, D. (2020, April 18). *Beginner's guide to use docker (Build, Run, Push and Pull)*. Medium. https://medium.com/@deepakshakya/beginners-guide-to-use-docker-build-run-push-and-pull-4a132c094d75

83. Shreiber, A. (2021, December 14). *A Practical Guide to DBSCAN Method - Towards Data Science*. Medium. https://towardsdatascience.com/a-practical-guide-to-dbscan-method-d4ec5ab2bc99

84. Singh, S. (2021, December 29). *All you need to know about the DBSCAN Algorithm - Analytics Vidhya*. Medium. https://medium.com/analytics-vidhya/all-you-need-to-know-about-the-dbscan-algorithm-f1a35ed8e712

85. Singh, P. (2021). *Deploy Machine Learning Models to Production*. Apress.

86. SOUKA, A. B. M. I. STANDARD ON LOGGING AND MONITORING.

87. *statsmodels*. (2021, November 13). PyPI. https://pypi.org/project/statsmodels/

88. Sultan, S., & Jensen, C. D. (2021). Secondary Use Prevention in Large-Scale Data Lakes. In *Intelligent Computing* (pp. 967-985). Springer, Cham.

89. *Supervised learning algorithm example*. (2021). [Image]. Retrieved 30 October 2021, from https://csdl-images.computer.org/mags/so/2016/05/figures/mso20160501101.gif.

90. *tensorflow*. (2021, November 4). PyPI. https://pypi.org/project/tensorflow/

91. *The 10 Algorithms every Machine Learning Engineer should know*. (2021). [Image]. Retrieved 30 October 2021, from https://media.geeksforgeeks.org/wp-content/cdn-uploads/20190522174744/MachineLearning.png

92. The Institute of Electrical and Electronics Engineers, Inc. (1998, June 25). IEEE Recommended Practice for Software Requirements Specifications. *IEEE-SA Standards Board.* http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

93. *TrustRadius*. (2021, December 16). Trustradius. https://www.trustradius.com/products/jupyter-notebook/reviews?qs=pros-and-cons

94. Ünlü, R., & Xanthopoulos, P. (2019). A weighted framework for unsupervised ensemble learning based on internal quality measures. *Annals of Operations Research*, *276*(1), 229-247.

95. *Unsupervised Learning: Gaussian Mixture Model (1D GMM)*. (2018, 2 diciembre). [Vídeo]. YouTube. https://www.youtube.com/watch?v=fVsmnZqrBUs

96. V. (2019b, September 6). *Gaussian Mixture Models Clustering - Explained*. Kaggle. https://www.kaggle.com/vipulgandhi/gaussian-mixture-models-clustering-explained

97. Wolford, B. (2022, May 26). *What is GDPR, the EU's new data protection law?* GDPR.Eu. https://gdpr.eu/what-is-gdpr/

98. Wang, L., Zhang, Z., Zhang, X., Zhou, X., Wang, P., & Zheng, Y. (2021). A Deep-forest based approach for detecting fraudulent online transaction. In *Advances in Computers* (Vol. 120, pp. 1-38). Elsevier.

99. Wang, H., Bah, M. J., & Hammad, M. (2019). Progress in outlier detection techniques: A survey. *Ieee Access*, *7*, 107964-108000.

100. Wang, X., & Sloan, I. H. (2007). Brownian bridge and principal component analysis: towards removing the curse of dimensionality. *IMA Journal of Numerical Analysis*, *27*(4), 631-654.

101. *What are insider threats? | IBM*. (2022). IBM Security. https://www.ibm.com/topics/insider-threats

102. *What is Clustering?* (2022). NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/data-science/clustering/

103. *What is Component Diagram?* (2021). Visual Paradigm. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

104. *Why is Julia faster than Go?* (2016). Quora. https://www.quora.com/Why-is-Julia-faster-than-Go

105. Wikipedia contributors. (2022, January 13). Cluster analysis. In *Wikipedia, The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=Cluster_analysis&oldid=1065378878

106. Wikipedia contributors. (2021, September 7). DBSCAN. In *Wikipedia, The Free Encyclopedia*. Retrieved from https://en.wikipedia.org/w/index.php?title=DBSCAN&oldid=1043021240

107. Zimmermann, D. (2022, June 2). Confidential Data of Pegasus Airlines Was Exposed to the Public. *Adware Guru*. https://adware.guru/pegasus-airlines-data/#:%7E:text=According%20to%20researchers%2C%20cybercriminals%20can%20spoof%20sensitive%20flight,Subsidiary%20airlines%20using%20PegasusEFB%20may%20also%20be%20affected

# ANNEX I: PROJECT MANAGEMENT

This annex details aspects related to the management of the project, including the planning of the work, the technical means used for its development and the socio-economic analysis of it.

## 1. Project plan

Here it is detailed the initial plan and the actual schedule of the project, as well as a study of the deviations observed between the two.

### 1.1 Initial project plan

For the initial plan, it has been considered the different stages in which the project has to be broken down and the estimated effort for each of them. Additionally, it should be contemplated that this project must be carried along with another project (that requires the same amount of time), the completion of three modules from university, and a full-time work that requires a total of 39 hours per week. However, from October 4th 2021 till 3th of April, the full-time work will be related to the deployment of the model, so it has been estimated that, out of the 125 working days, the 80% will be efficiently dedicated to the deployment stage, which results in 100 days.

Hence, the planning of the project goes from October 4th 2021 to 15th of May 2022. Table 12 shows the expected start and end date for each task, while Figure 30 presents the Gantt chart with the expected schedule.

| | days | Start date | End date |
|---|---|---|---|
| **Overall project** | **176** | **Mon. 4/10/2021** | **Thu. 15/05/2022** |
| **Initial planning** | **3** | **Mon. 4/10/2021** | **Wed. 6/10/2021** |
| **State of the art** | **6** | **Sat. 9/10/2021** | **Sun. 24/10/2021** |
| Data Lakes | 2 | Sat. 9/10/2021 | Sun. 10/10/2021 |
| Protection mechanisms for combating fraud | 2 | Sat. 16/10/2021 | Sun. 17/10/2021 |
| Anomaly detection | 2 | Sat. 23/10/2021 | Sun. 24/10/2021 |
| **Analysis** | **12** | **Sat. 30/10/2021** | **Sat. 11/12/2021** |
| General perspective | 1 | Sat. 30/10/2021 | Sat. 30/10/2021 |
| Technological research | 3 | Sun. 31/10/2021 | Sun. 7/11/2021 |
| High-level design | 2 | Sat. 13/11/2021 | Sun. 14/11/2021 |
| Use Cases | 1 | Sat. 20/11/2021 | Sat. 20/11/2021 |
| Software Requirements | 2 | Sun. 21/11/2021 | Sun. 28/11/2021 |
| Acceptance test plan design | 3 | Sat. 4/12/2021 | Sat. 11/12/2021 |
| **Detailed design** | **18** | **Sat. 18/12/2021** | **Sun. 13/02/2022** |
| Software design | 10 | Sat. 18/12/2021 | Sun. 16/01/2022 |
| Sequence diagram | 8 | Sat. 22/01/2022 | Sun. 13/02/2022 |
| **Software deployment** | **100** | **Mon. 4/10/2021** | **Sun. 3/04/2022** |
| **Experimentation and results** | **15** | **Sat. 9/04/2022** | **Sat. 23/04/2022** |
| **Preparation of the report**[7] | **22** | **Sun. 24/04/2022** | **Thu. 15/05/2022** |

**Table 12**. Detailed initial planning

[7] documentation of the project (including introduction and objectives, and annexes).

**Design and Deployment of an Access Control Module for Data Lakes**

Start Date: Monday, October 4 of 2021
End Date: Sunday, May 15 of 2022

| Stage | Progress | Start | End |
|---|---|---|---|
| **Overall project** | 100% | **4/10/21** | **15/5/22** |
| **Initial planning** | 100% | **4/10/21** | **6/10/21** |
| **State of the art** | 100% | **9/10/21** | **24/10/2021** |
| Data Lakes | 100% | 9/10/2021 | 10/10/2021 |
| Protection mechanisms for combating fraud | 100% | 16/10/2021 | 17/10/2021 |
| Anomaly detection | 100% | 23/10/2021 | 24/10/2021 |
| **Analysis** | 100% | **30/10/21** | **11/12/2021** |
| General perspective | 100% | 30/10/2021 | 30/10/2021 |
| Technological research | 100% | 31/10/21 | 7/11/21 |
| High-level design | 100% | 13/11/21 | 14/11/21 |
| Use Cases | 100% | 20/11/21 | 20/11/21 |
| Software Requirements | 100% | 21/11/21 | 28/11/21 |
| Acceptance test plan design | 100% | 4/12/21 | 11/12/21 |
| **Detailed design** | 100% | **18/12/21** | **13/2/22** |
| Software design | 100% | 18/12/21 | 16/1/22 |
| Sequence diagram | 100% | 22/1/22 | 13/2/22 |
| **Software deployment** | 100% | **4/10/21** | **3/4/22** |
| **Experimentation and results** | 100% | **9/4/22** | **23/4/22** |
| **Preparation of the report** | 100% | **24/4/22** | **15/5/22** |

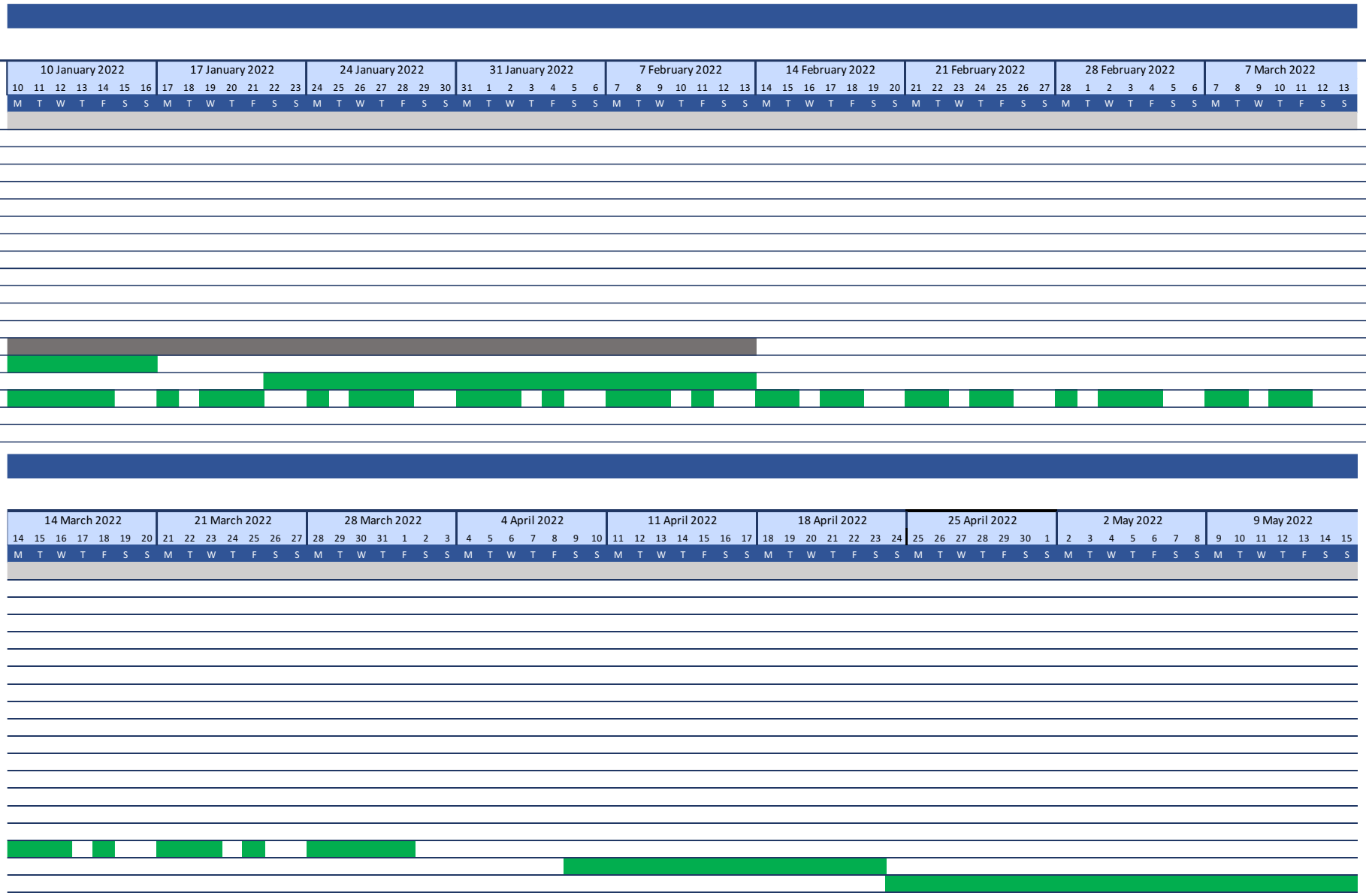



**Figure 30**. Gannt chart of initial planning

It can be observed in this initial planning that more than half of the time allocated to the development of the solution, essential part of any project.

### 1.2 Actual project plan

The purpose of this section is to illustrate the actual development of the project and compare it with the initial planning to study the deviations that occurred throughout the project.

Table 13 shows the days spent in each of the tasks. It is relevant that, although 176 days were estimated for the overall project, it finally has taken 39 more days. In fact, Table 14 illustrates the difference between the initial and actual schedules of the project.

| | **days** | **Start date** | **End date** |
|---|---|---|---|
| **Overall project** | **215** | **4/10/21** | **20/6/22** |
| **Initial planning** | **3** | **4/10/21** | **6/10/21** |
| **State of the art** | **8** | **9/10/21** | **26/10/2021** |
| Data Lakes | 3 | 9/10/2021 | 11/10/2021 |
| Protection mechanisms for combating fraud | 1 | 17/10/2021 | 17/10/2021 |
| Anomaly detection | 4 | 23/10/2021 | 26/10/2021 |
| **Analysis** | **22** | **6/11/21** | **30/05/2021** |
| General perspective | 2 | 6/11/2021 | 7/11/2021 |
| Technological research | 11 | 13/11/21 | 23/11/21 |
| High-level design | 3 | 28/5/21 | 30/5/21 |
| Use Cases | 1 | 4/12/21 | 4/12/21 |
| Software Requirements | 2 | 11/12/21 | 19/12/21 |
| Acceptance test plan design | 3 | 26/12/21 | 28/12/21 |
| **Detailed design** | **5** | **22/4/21** | **1/5/22** |
| Software design | 3 | 22/4/21 | 24/4/22 |
| Sequence diagram | 2 | 30/5/22 | 1/5/22 |
| **Software deployment** | **109** | **18/10/21** | **12/6/22** |
| **Experimentation and results** | **10** | **4/6/22** | **13/6/22** |
| **Preparation of the report** | **58** | **9/10/22** | **20/6/22** |

**Table 13**. Actual schedule of the project

| | **Estimated days** | **Actual days** | **Days of difference** | **% of variation** |
|---|---|---|---|---|
| **Overall project** | **176** | **215** | 39 | 22.16% |
| **Initial planning** | **3** | **3** | 0 | 0.00% |
| **State of the art** | **6** | **8** | 2 | 33.33% |
| Data Lakes | 2 | 3 | 1 | 50.00% |
| Protection mechanisms for combating fraud | 2 | 1 | -1 | -50.00% |
| Anomaly detection | 2 | 4 | 2 | 100.00% |
| **Analysis** | **12** | **22** | 10 | 83.33% |
| General perspective | 1 | 2 | 1 | 100.00% |
| Technological research | 3 | 11 | 8 | 266.67% |
| High-level design | 2 | 3 | 1 | 50.00% |
| Use Cases | 1 | 1 | 0 | 0.00% |

| | Estimated days | Actual days | Days of difference | % of variation |
|---|---|---|---|---|
| Software Requirements | 2 | 2 | 0 | 0.00% |
| Acceptance test plan design | 3 | 3 | 0 | 0.00% |
| **Detailed design** | **18** | **5** | -13 | -72.22% |
| Software design | 10 | 3 | -7 | -70.00% |
| Sequence diagram | 8 | 2 | -6 | -75.00% |
| **Software deployment** | **100** | **109** | 9 | 9.00% |
| **Experimentation and results** | **15** | **10** | -5 | -33.33% |
| **Preparation of the report** | **22** | **58** | 36 | 163.64% |

**Table 14**. Analysis of planning deviations

According to this information, the biggest deviation had been experienced on the preparation of the report, which has taken 36 extra days due to the several revisions and details that had to be covered. Other tasks related to the detailed design have finally taken less time than estimated.

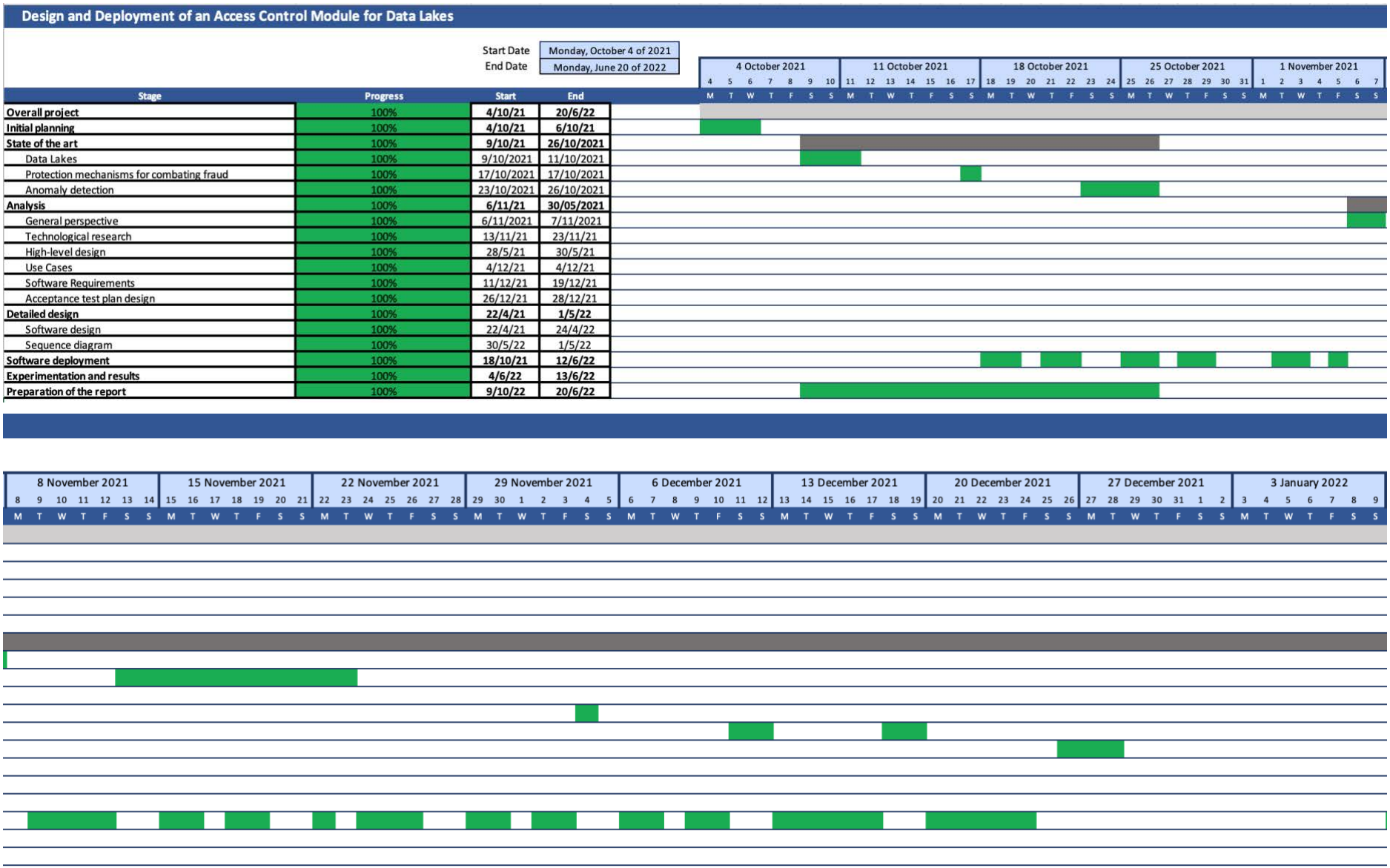
Overall, the project has required 22.16% more of time than it was first expected, however, the initial plan was made so that this could happen, and the project would still be finished on the desired date.

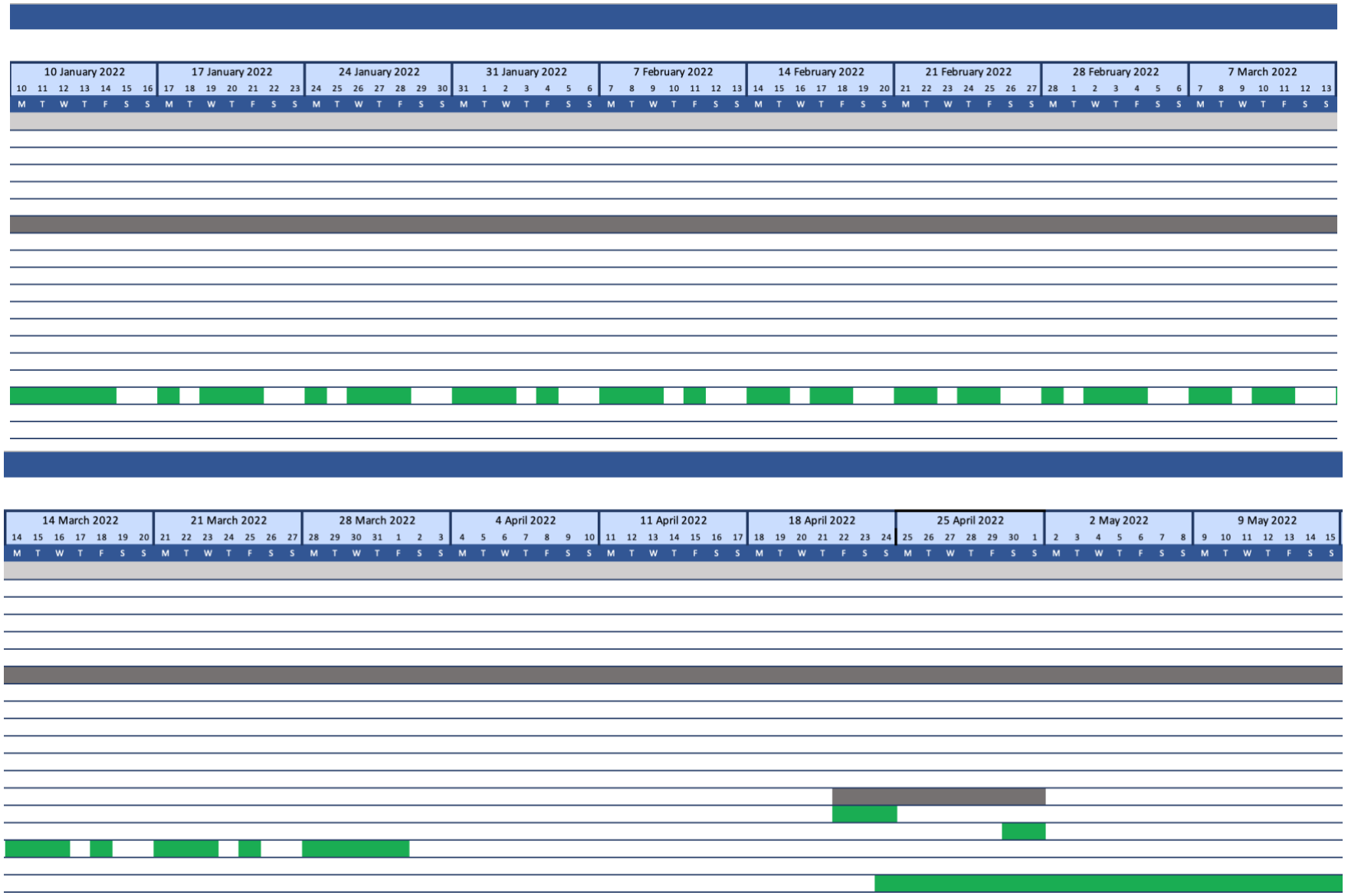
Finally, displays the Gantt chart with the final planning, which shows that the timeline has been spread, going the end date one moth ahead, from May to June.

**Design and Deployment of an Access Control Module for Data Lakes**

Start Date: Monday, October 4 of 2021
End Date: Monday, June 20 of 2022

| Stage | Progress | Start | End |
|---|---|---|---|
| **Overall project** | 100% | **4/10/21** | **20/6/22** |
| **Initial planning** | 100% | **4/10/21** | **6/10/21** |
| **State of the art** | 100% | **9/10/21** | **26/10/2021** |
| Data Lakes | 100% | 9/10/2021 | 11/10/2021 |
| Protection mechanisms for combating fraud | 100% | 17/10/2021 | 17/10/2021 |
| Anomaly detection | 100% | 23/10/2021 | 26/10/2021 |
| **Analysis** | 100% | **6/11/21** | **30/05/2021** |
| General perspective | 100% | 6/11/2021 | 7/11/2021 |
| Technological research | 100% | 13/11/21 | 23/11/21 |
| High-level design | 100% | 28/5/21 | 30/5/21 |
| Use Cases | 100% | 4/12/21 | 4/12/21 |
| Software Requirements | 100% | 11/12/21 | 19/12/21 |
| Acceptance test plan design | 100% | 26/12/21 | 28/12/21 |
| **Detailed design** | 100% | **22/4/21** | **1/5/22** |
| Software design | 100% | 22/4/21 | 24/4/22 |
| Sequence diagram | 100% | 30/5/22 | 1/5/22 |
| **Software deployment** | 100% | **18/10/21** | **12/6/22** |
| **Experimentation and results** | 100% | **4/6/22** | **13/6/22** |
| **Preparation of the report** | 100% | **9/10/22** | **20/6/22** |




10 January 2022
17 January 2022
24 January 2022
31 January 2022
7 February 2022
14 February 2022
21 February 2022
28 February 2022
7 March 2022
14 March 2022
21 March 2022
28 March 2022
4 April 2022
11 April 2022
18 April 2022
25 April 2022
2 May 2022
9 May 2022

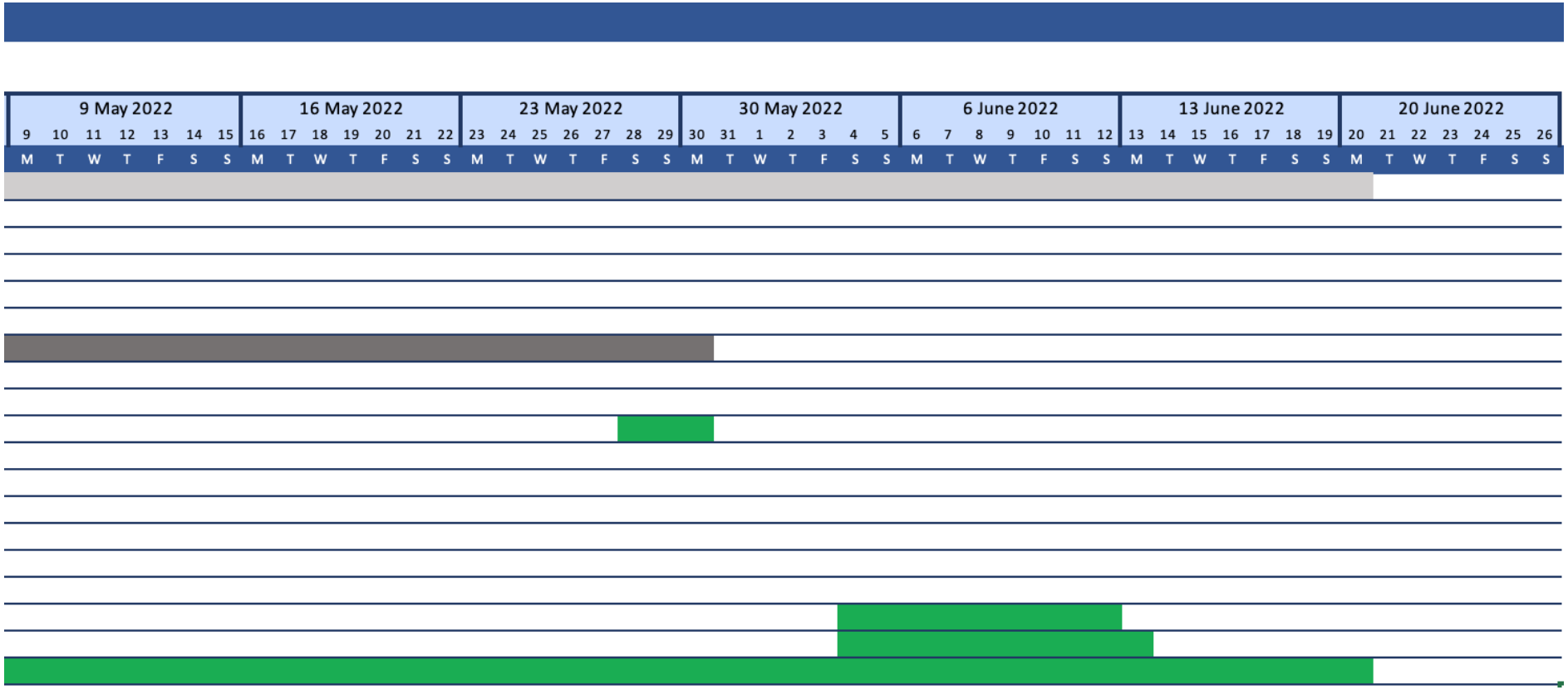



**Figure 31**. Gantt chart of actual planning

## 2. Technical means used for the project

This section details the tools that have been used for the development of the project.

First, Table 15 lists the technologies that have been required for the project, along with a brief description.

| Technology | Description |
|---|---|
| Jupyter Notebook | Web application for creating and executing de code programmed in Python. |
| Docker | Application development solution that enables businesses to build applications or share container images. |
| Putty | Software terminal emulator for launching the Docker container. |
| draw.io | Modeling software used for the creation of all diagrams in the analysis and design sections. |
| Windows 10 Enterprise | Operating system on which the implementation of the system was carried out. |
| macOS Big Sur 11.6 | Operating system on which the tasks of analysis, design and documentation of the project have been developed. |
| Microsoft Office | Office suite used for the documentation of the project and other tasks such as the Gantt chart or some of the figures. |
| Google Chrome | Web browser used throughout the development of the project. |

**Table 15.** Technological tools used in the project

Additionally, Table 16 details the necessary equipment.

| Equipment | Description |
|---|---|
| HP EliteBook 845 G8 Notebook PC | Used for the design and deployment of the solution. |
| MacBook Pro 15-inch<br>2,2 GHz, Intel “Core i7” processor | Used for all other tasks of this project. |
| iPhone 11 | Smartphone used as agenda and for checking the email in absence of the previous laptop. |

**Table 16.** Equipment used for the project

## 3. Economic analysis of the project

This section focuses on the economic analysis of the project. It is essential, when carrying out a project like this, to estimate the budget that would entail its development. For this reason, this section analyzes the expected costs of labor, equipment, and software necessary for the development of the project, in order to obtain an estimated budget to be presented to the company, who considers its viability. The estimates costs to be incurred are:

### Costs of labor

The vast majority of this project will be carried out by two people, one who acts as supervisor, and the other who is in charge of the design and deployment of the solution. The salary for the second has been established beforehand by the company at 23.000,00€ gross per year (which includes the net salary paid to the employee and related payroll taxes and benefits), being therefore the estimated cost for the company of 1.917,00€ per month, approximately. Supervisor's salary has been estimated, considering his experience, at 60,000€ gross per year, representing a cost for the company of 5.000,00€ per month.

On the other hand, the total duration of the project, shown in the previous planning, was initially estimated to 6 months, so the total cost of labor is summarized in Table 17.

| Position | Wage | Time | Cost |
|---|---|---|---|
| Supervisor | 1.917,00€/month | 6 months | 11.502,00€ |
| Designer and developer | 5.000,00€/month | 6 months | 30.000,00€ |
| | | **Total** | **41.502,00€** |

**Table 17**. Estimated cost of labor

### Costs of equipment

The only equipment required for the realization of the project is the HP EliteBook 845 G8 Notebook PC, as stated in the previous section. Considering its price at the beginning of the project was 1.350,00€ and 2 laptops (for each of the employees) would be required, the total cost of equipment results in 2.700,00€.

However, taking into account the cost of amortization, that has been estimated as 4 years (48 months), a final cost of use results in 337,50€, as detailed in Table 18.

| Product | Quantity | Price | Use | Useful life | Cost |
|---|---|---|---|---|---|
| HP EliteBook 845 G8 Notebook PC | 2 | 1.350,00€ | 6 months | 48 months | **337,50€** |

**Table 18**. Estimated cost of equipment

### Costs of software

Some licensed programs were necessary for the development of this project, listed below:

1. Microsoft 365 Business Premium: 18,60€ per user and per month[8].
2. Docker Business: 20,00€ per user and per month[9].

[8] price retrieved from: https://www.microsoft.com/en-us/microsoft-365/business/compare-all-microsoft-365-business-products

[9] price retrieved from: https://www.silicon.es/docker-introduce-un-nuevo-nivel-de-suscripcion-para-empresas-medianas-y-grandes-2444198#:~:text=Docker%20introduce%20un%20nuevo%20nivel%20de%20suscripci%C3%B3n%20para,un%20precio%20mensual%20de%2021%20d%C3%B3lares%20por%20usuario

Other software solutions intended to be used, such as Jupyter Notebook or Putty can be used for free.

Thus, the overall cost of software, considering the already necessities of the project can be estimated as follows:

| Product | Price per user and month | Quantity | Months | Cost |
|---|---|---|---|---|
| Microsoft 365 Business Premium | 18,60€ | 2 | 6 | 223,20€ |
| Docker Business | 20,00€ | 2 | 6 | 240,00€ |
| | | | **Total** | **463,20€** |

**Table 19**. Estimated cost of software

### Total cost

Once all the costs of the project have been found separately, a total estimate of the project's budget is obtained, presented in Table 20.

| Element | Cost |
|---|---|
| Labor | 41.502,00€ |
| Equipment | 337,50€ |
| Software | 463,20€ |
| **Total** | **42.302,70€** |

**Table 20**. Total estimate of the project's budget

Thus, the total estimated budget for the project is 42.302,70€, being 98% of it associated to employee's salary.

# ANNEX II: TEMPLATES

This annex shows the templates used for the definition of the use cases, software requirements and acceptance test plan.

### 1. Use-cases template

| **Identifier: UC-#** | |
|---|---|
| **Use case name** | |
| **Author** | |
| **Use case overview** | |
| **Actors** | |
| **Precondition** | |
| **Post-condition** | |
| **Basic path** | |
| **Alternative path** | |

**Table 21**. Template for use cases definition

### 2. Software requirements template

| **Software requirements** | | | |
|---|---|---|---|
| **Type** | | | |
| **Subtype** | | | |
| **Id** | **Name** | **Description** | **Priority** |
| | | | |

**Table 22.** Template for software requirements

### 3. Acceptance test plan template

| **Acceptance test plan** | | | | |
|---|---|---|---|---|
| **Id** | **Test element** | **Input** | **Expected output** | **Recovery action** |
| | | | | |

**Table 23.** Template for acceptance test plan