People's Democratic Republic of Algeria
الجمهورية الجزائرية الديمقراطية الشعبية
Ministry of Higher Education and Scientific Research
وزارة التعليم العالي و البحث العلمي

Saad Dahlab University of Blida
Science Faculty
Computer Science Department

End-of-study Report

in view of obtaining a Master's diploma in Computer science

Option : Informatics systems and networks

behavioral analysis of Active Directory logs

Presented by:
Mr. AISSANI Youcef

Supervisor :
Mr. NEFFAH Mohamed

Defense Jury:

Mrs.	:	Cherfa Imene
Mrs.	:	Tebbi Hanane

Promotion : 2022/2023

acknowledgements

First and foremost, I thank God for allowing me to reach this point. I would like to thank my supervisor NEFFAH MOHAMED for all the work and guidance he gave me along the way. I would like to thank all my family for supporting me along the way of my scientific journey, their help during my studying and research time was invaluable.i would also like to thank friends who made my journey more enjoyable.

-Aissani Youcef

Abstract

This thesis focuses on the behavioral analysis of log data for anomaly detection and clustering in the field of cybersecurity. The objective is to obtain insights into patterns, anomalies, and potential threats present in sonatrach's logs. Various algorithms, including K-means, DBSCAN, GMM, and Isolation Forest, were evaluated and compared in terms of their performance in detecting anomalies and clustering the data. The results showed that while K-means performed poorly, DBSCAN, GMM, and Isolation Forest exhibited different levels of sensitivity and performance. The findings provide valuable insights for improving anomaly detection and threat analysis in cybersecurity.

Keywords : anomaly detection, behavioral analysis, clustering, preprocessing, machine learning.

Résumé

Cette thèse se concentre sur l'analyse comportementale des données de logs pour la détection d'anomalies et le regroupement dans le domaine de la cybersécurité. L'objectif est d'obtenir des informations sur les motifs, les anomalies et les menaces potentielles présentes dans les logs de Sonatrach. Différents algorithmes, tels que K-means, DBSCAN, GMM et Isolation Forest, ont été évalués et comparés en termes de performances pour détecter les anomalies et regrouper les données. Les résultats ont montré que, tandis que K-means a obtenu de mauvais résultats, DBSCAN, GMM et Isolation Forest ont démontré différents niveaux de sensibilité et de performance. Ces résultats offrent des informations précieuses pour améliorer la détection d'anomalies et l'analyse des menaces en cybersécurité.

Mots clés : détection d'anomalies, analyse comportementale, regroupement, prétraitement, apprentissage automatique.

Contents

List of Figures

list of abbreviations and acronyms

SVM	*Support Vector machine.*
IP	*Internet Protocol.*
TF-IDF	*term frequency-inverse document frequency.*
LDA	*Linear Discriminant Analysis.*
PCA	*Principal Component Analysis.*
t-SNE	*t-Distributed Stochastic Neighbor.*
URI	*Uniform Resource Identifier .*
TruncatedSVD	*truncated singular value decomposition.*
DBSCAN	*Density-based spatial clustering of applications with noise.*
GMM	*Gaussian Mixture Model.*
HTTP	*Hypertext Transfer Protocol.*
SSE	*Sum of Squared Errors.*
WCSS	*Within-Cluster Sum of Squares.*
CPU	*Central Process Unit.*
RAM	*Random Access Memory .*
GPU	*Graphics Processing Unit.*
OS	*Operating System.*

Presentation of the host organization

SONATRACH is the proud Algerian national company dedicated to the exploration, exploitation, pipeline transportation, processing, and marketing of hydrocarbons and their derivatives. With its mission, it is committed to optimizing valuable national hydrocarbon resources, thus generating wealth for the economic and social development of the country. As a major player in the oil and gas industry, SONATRACH holds a leading position in Africa and the Mediterranean region. Its activities are deployed in four essential areas: Upstream, Downstream, Pipeline Transportation, and Marketing. Through its involvement in various projects in Africa, Latin America, and Europe, the company collaborates with diverse partners, thereby strengthening its international influence.

For over 50 years, SONATRACH has fully assumed its role as an engine of the national economy. By effectively leveraging the country's hydrocarbon resources, it actively contributes to the creation of wealth that promotes the economic and social development of the country. With an unwavering commitment, SONATRACH continues to excel in its activities of exploration, exploitation, transportation, and marketing, ensuring a promising future for Algeria.

Motivation for our project

Last year, the Director of Digitalization and Information System (DSI) made an important discovery regarding unusual connections in the Active Directory logs. These abnormal connections raised concerns and necessitated an investigation and anomaly detection within the system. Our research is taking place within the DSI's Innovation Center, located at Sonatrach headquarters in Hydra. In this collaborative environment, we have had access to valuable log data under the supervision of engineer Mr. Neffah. This opportunity allows us to deepen the analysis and detection of anomalies with the aim of improving the overall security and integrity of the system.

General Introduction

In recent years, cybersecurity has become a critical concern due to the increasing number of cyber threats and attacks. Anomaly detection plays a vital role in identifying suspicious activities and potential security breaches. Log data analysis offers valuable information for understanding system behavior and detecting anomalies. This thesis focuses on the analysis of log data and aims to explore different algorithms for anomaly detection and clustering.

Our thesis begins with the Fundamental Principles, providing an overview of machine learning and anomaly detection in logs. The methodology chapter describes our conceptual system, preprocessing techniques as well as the challenges faced during the collection and preprocessing.

The Implementation chapter describes the algorithms used in detail, and presents the tools used and the findings of the evaluation. It highlights the performance and limitations of each algorithm in detecting anomalies and clustering the log data. The distributions of suspicious connections based on the clustering results are visualized.

The thesis also includes the development of an application that allows the network engineer to apply the selected algorithms to log files and identify users with abnormal behavior. The application provides a user-friendly interface and enables further analysis of abnormal logs.

Our contribution to cybersecurity here is by comparing different machine learning algorithms for anomaly detection and clustering in unlabeled log data analysis. The results highlight the strengths and weaknesses of each algorithm and provide valuable insights for improving anomaly detection techniques. The developed application offers a practical tool for analyzing log data and identifying abnormal behavior, enhancing the overall cybersecurity efforts.

Chapter 1

Fundamental Principles

1.1 Introduction

Machine learning, an exciting and rapidly advancing field, leverages mathematical algorithms to empower machines with the ability to learn and automate tasks. It has revolutionized numerous domains by providing intelligent solutions and automating complex processes.

Log anomaly detection has been a longstanding challenge, particularly due to the sheer volume and complexity of logs generated by systems. Manual detection by technicians proved to be impractical and time-consuming, necessitating the development of automated techniques. Artificial intelligence (AI) has emerged as a powerful ally in addressing this challenge, offering efficient and effective solutions for log analysis and anomaly detection.

We provide in this chapter a brief overview of the different methods used in machine learning and how log anomaly detection is structured. This will serve as a foundation for understanding the subsequent sections, where we delve into log anomaly detection. By leveraging the capabilities of machine learning algorithms, we can unlock hidden patterns, correlations, and anomalies within log data, enabling us to make informed decisions and take proactive measures to mitigate risks.

1.2 Overview of Machine Learning and Deep Learning:

1.3 Types of Learning

1.3.1 Supervised Learning

Supervised learning is one of the main approaches in machine learning. In this type of learning, we start with an existing dataset that includes both inputs and their corresponding outputs. The goal is to build a model capable of mapping new inputs to their corresponding outputs based on the provided training examples.

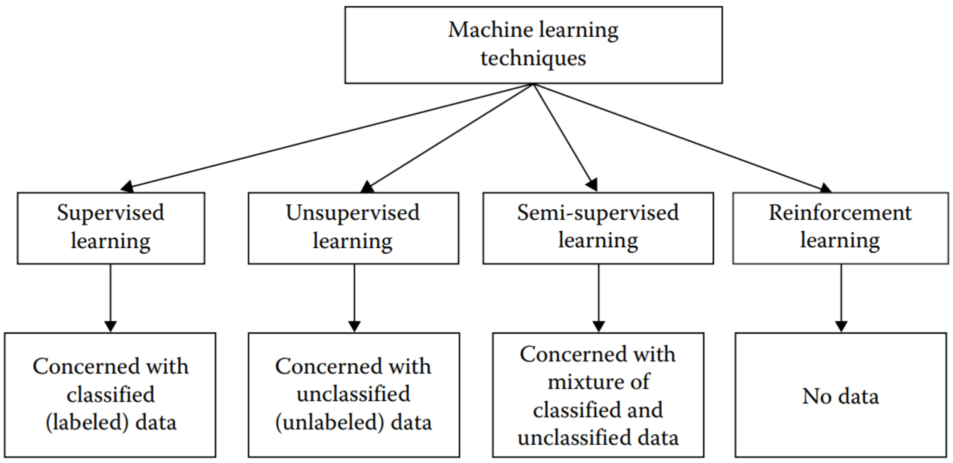

Machine learning techniques
Supervised learning
Unsupervised learning
Semi-supervised learning
Reinforcement learning
Concerned with classified (labeled) data
Concerned with unclassified (unlabeled) data
Concerned with mixture of classified and unclassified data
No data

Figure 1.1: Types of learning in artificial intelligence and the required data[1]

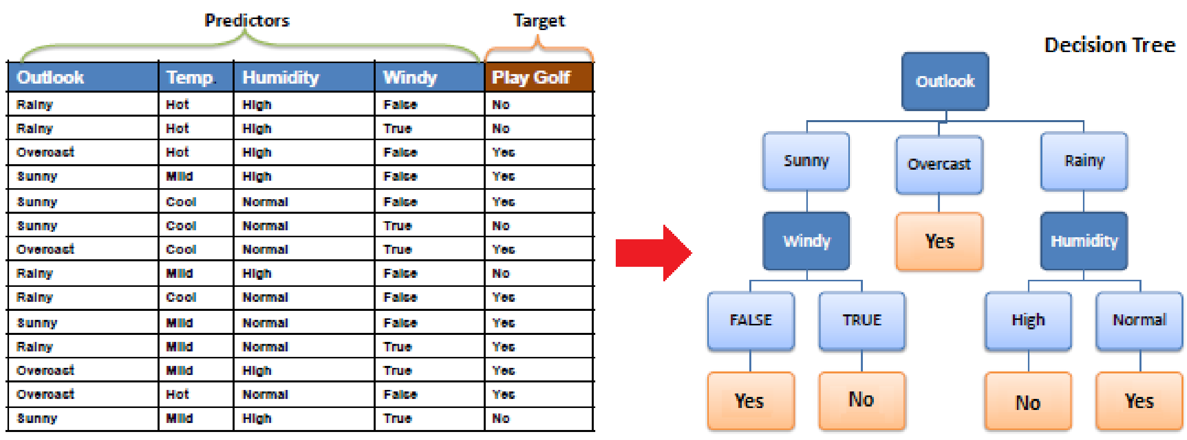
The training dataset consists of input-output pairs, also known as training examples. Each training example consists of an input and the corresponding output that we want to predict. The learning process involves adjusting the model's parameters to minimize the difference between the predicted outputs by the model and the actual outputs provided in the training dataset.

Once our model has been trained on the training dataset, we can use it to predict the corresponding outputs for new inputs that have not been seen during the training. This allows us to generalize the model beyond the specific examples it was trained on[2].

Some examples of supervised learning algorithms include:

Decision Trees

Decision trees are widely used supervised learning algorithms in the field of machine learning. They use a tree-like structure to represent decisions based on the features of a given input. Each node in the tree corresponds to a condition or feature to check, and each branch represents a possible response or another condition to evaluate. The leaves of the tree represent the predicted classes or values.



Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Figure 1.2: Example of a decision tree[3]

Naive Bayes

The Naive Bayes classification model is a widely used algorithm in the field of machine learning. It is based on Bayes' theorem, which is a statistical method for calculating the conditional probability of an event using prior information.

In the case of Naive Bayes, it is assumed that each feature used for classification is independent of the others. This naive assumption simplifies the calculation of conditional probabilities and allows for an efficient implementation of the algorithm. In other words, it considers each feature to independently contribute to the final classification, without considering potential interactions between them.

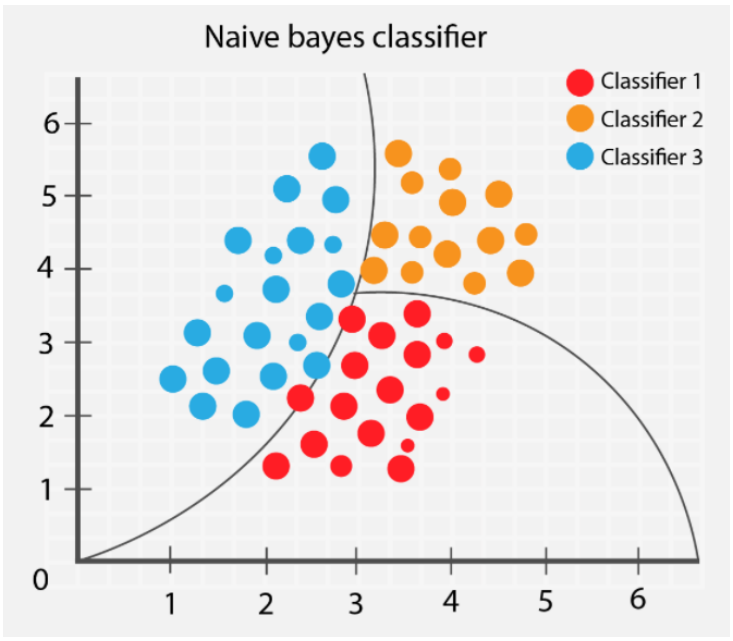


Figure 1.3: An example of Naive Bayes's classification[4]

Support Vector Machines (SVM)

Support Vector Machines (SVM) are powerful algorithms used in machine learning for classification and regression tasks. Their strength lies in their ability to effectively separate different data classes using hyperplanes in a high-dimensional space.

The SVM approach is based on the concept of maximum margin. The goal is to find the hyperplane that optimally separates the classes, by maximizing the distance between the closest observations of each class and the hyperplane. This maximum margin helps minimize the classification error and ensures good generalization of the model.

SVMs are particularly useful when the classes are linearly separable, meaning that there exists a hyperplane that can perfectly distinguish them. However, through the use of kernel functions, SVMs can also be extended to handle nonlinear classification problems by transforming the data into higher-dimensional spaces where linear separation becomes possible[5].

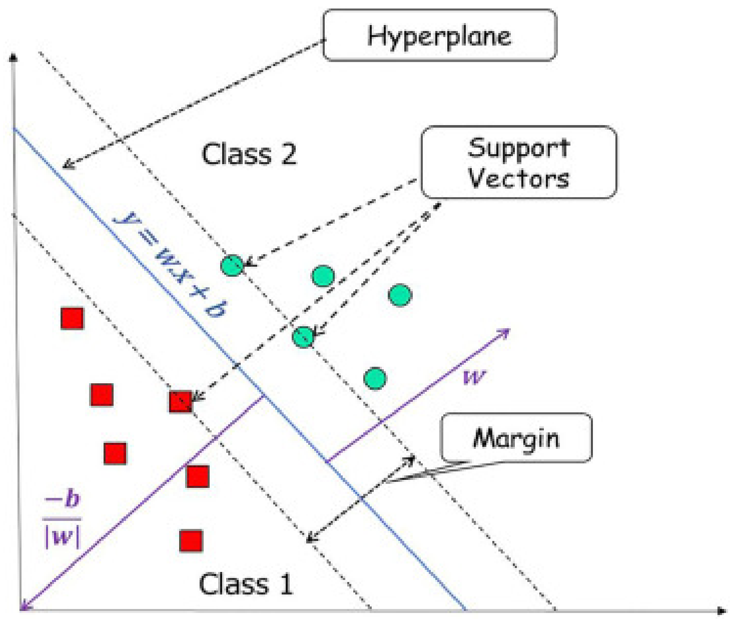


Figure 1.4: Support vector machine classification[6]

1.3.2 Unsupervised Learning

Unsupervised learning is an approach in machine learning where we explore and analyze data without having predefined labels or target values. Unlike supervised learning, where we have labeled training data, in unsupervised learning, we let the algorithm discover hidden structures or patterns in the data.

The main goal of unsupervised learning is to find natural groups or clusters in the data based on similarity of features or other criteria. This helps identify intrinsic relationships and subtle patterns that can be used to group the data into coherent sets.

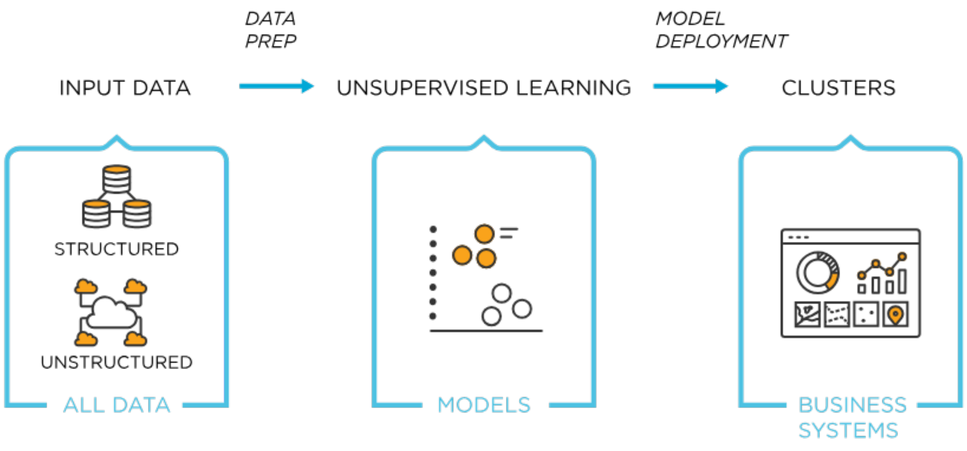


Figure 1.5: Unsupervised learning mechanism [7]

K-means

K-means is an unsupervised algorithm that uses Euclidean distance to cluster data. In this algorithm, the data is considered as points, and it is initialized with k points and iterated multiple times to obtain the final clusters.

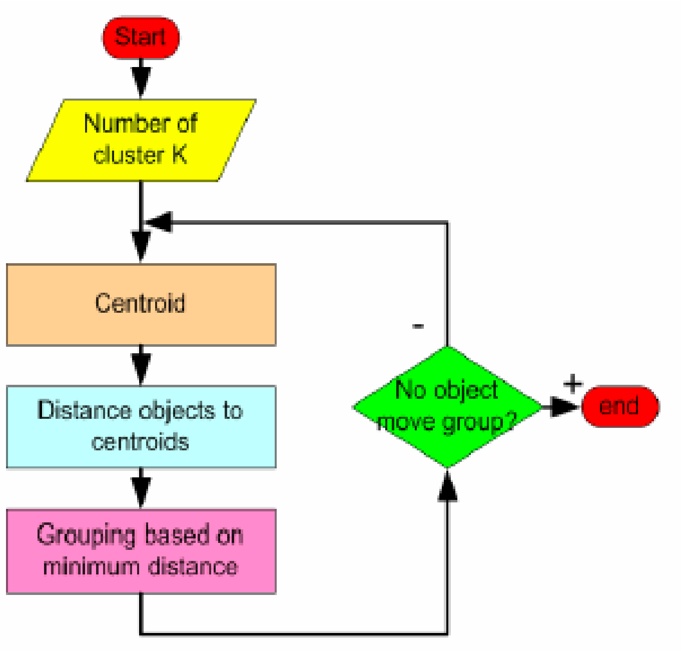

Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

Figure 1.6: K-means flow diagram[8]

1.3.3 Reinforcement Learning

Reinforcement learning is a branch of machine learning that relies on a reward system. Unlike supervised learning where the responses are provided, in reinforcement learning, the model learns through trial and error by interacting with an environment.

At the beginning of training, the model has no prior knowledge about the actions to take in order to maximize the reward. It starts with a starting point and explores different possible actions. With each action, the environment provides a reward or punishment based on the quality of the performed action.

The model uses these rewards to adjust its strategies and learn from previous choices. It seeks to maximize the overall reward over a period of interaction with the environment. Reinforcement learning is often used to solve sequential decision-making problems where the actions of an agent affect future states and obtained rewards.

1.3.4 Semi-Supervised Learning

Semi-supervised learning is a hybrid approach that combines elements of supervised and unsupervised learning. This type of learning is particularly useful when we have partially labeled data, meaning that only a small portion of the data is labeled while a large part remains unlabeled.

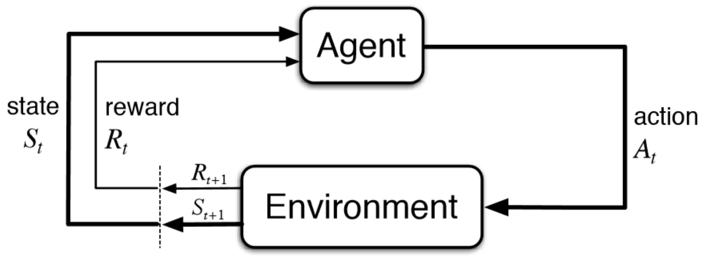


Figure 1.7: Diagram of reinforcement learning[9]

The goal of semi-supervised learning is to use the labeled data to guide learning on the unlabeled data. By leveraging the available information in the labeled data, the model seeks to generalize and extend its knowledge to the unlabeled data.

1.3.5 Ensemble Learning

Ensemble learning refers to the technique of combining multiple models to obtain better predictions. It includes techniques such as bagging, stacking, and boosting. Other applications of ensemble learning include assigning confidence to the model's decision, optimal feature selection, data fusion, incremental learning, non-stationary learning, and error correction[5].

1.4 Overview of Anomaly detection with logs

1.4.1 Structure of an Anomaly Detection System

Log collection

Log collection plays a fundamental role in anomaly detection. Software systems regularly perform logging to record the system's state, activities, errors, and other important events. Logs are chronological records that provide valuable information about the system's functioning and behavior[10]. A log typically consists of two parts: a constant part and a variable part. The constant part is predefined in the system's source code and includes information such as the log's severity level, event type, and the relevant module or function. The variable part contains event-specific information, including the associated message and the timestamp indicating when the log was written. In distributed systems, logs are often collected from multiple sources, such as servers, virtual machines, applications, and services. These logs are then centralized in a common repository for further analysis. Log collection can be done using dedicated tools or specific software agents configured to collect and transfer logs to the central repository.

Figure1.8 presents the log collection architecture [11].

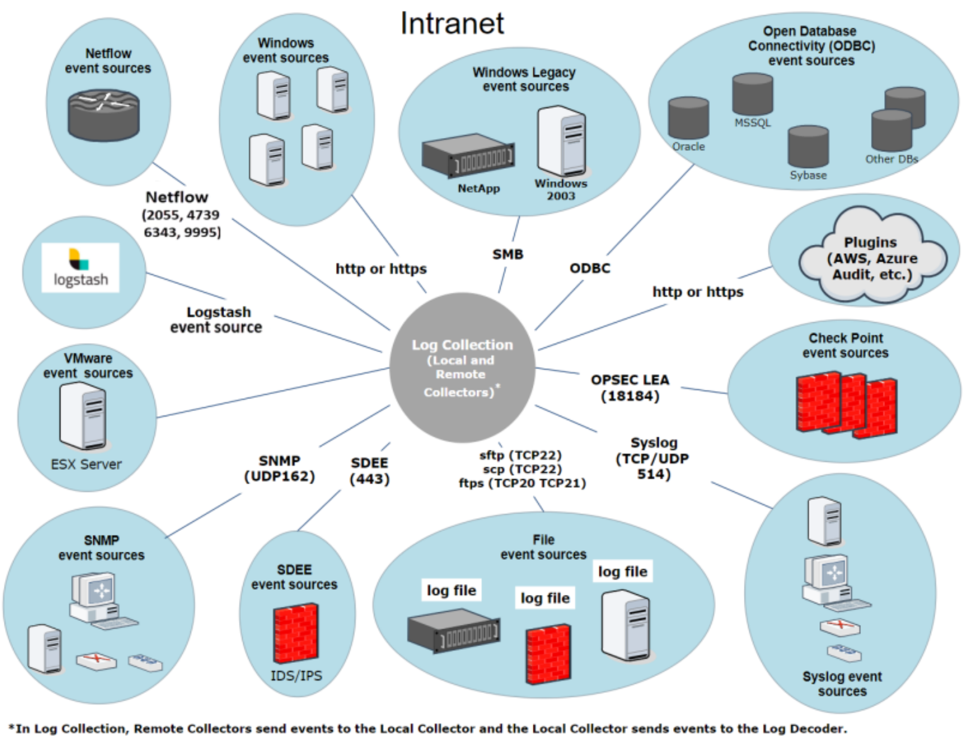
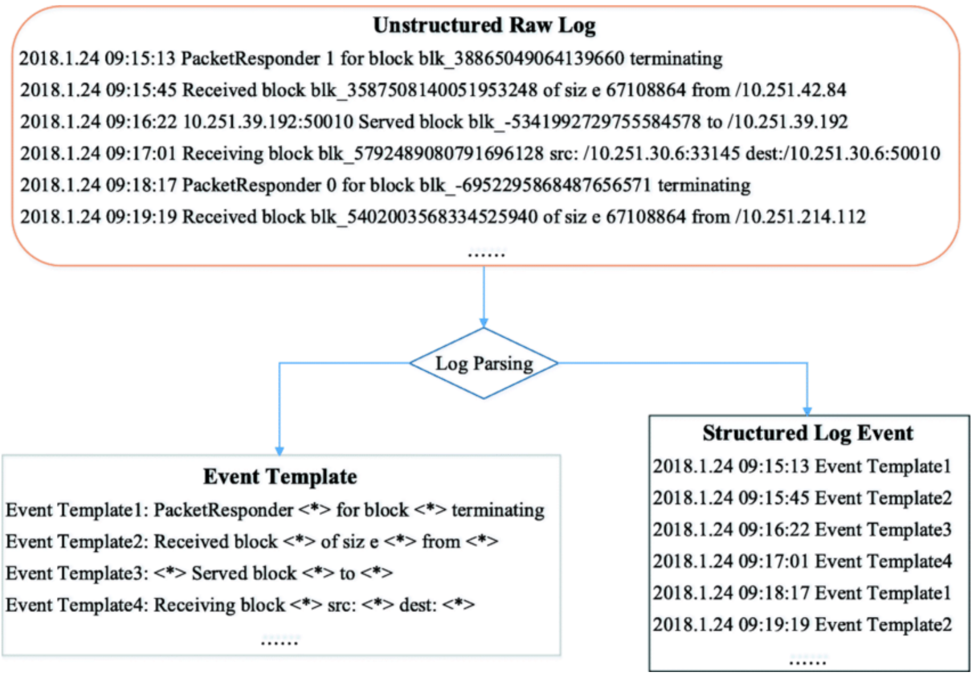

Intranet
Netflow event sources
Windows event sources
Windows Legacy event sources
NetApp
Windows 2003
Open Database Connectivity (ODBC) event sources
Oracle
MSSQL
Sybase
Other DBs
Netflow (2055, 4739 6343, 9995)
http or https
SMB
ODBC
logstash
Logstash event source
Plugins (AWS, Azure Audit, etc.)
http or https
Log Collection (Local and Remote Collectors)*
VMware event sources
ESX Server
Check Point event sources
OPSEC LEA (18184)
Syslog (TCP/UDP 514)
SNMP (UDP162)
SDEE (443)
sftp (TCP22) scp (TCP22) ftps (TCP20 TCP21)
SNMP event sources
SDEE event sources
IDS/IPS
File event sources
log file
log file
log file
Syslog event sources
*In Log Collection, Remote Collectors send events to the Local Collector and the Local Collector sends events to the Log Decoder.

Figure 1.8: Log collection architecture[11]

Log Parsing

A crucial step in the log anomaly detection process is log parsing[13] . Logs are typically semi-structured, meaning they can have different formats, variable information, and irrelevant elements for analysis. To enable more accurate and efficient analysis, it is essential to structure the logs in a way that retains only the relevant parts.



```
2018.1.24 09:15:13 PacketResponder 1 for block blk_38865049064139660 terminating
2018.1.24 09:15:45 Received block blk_3587508140051953248 of siz e 67108864 from /10.251.42.84
2018.1.24 09:16:22 10.251.39.192:50010 Served block blk_-5341992729755584578 to /10.251.39.192
2018.1.24 09:17:01 Receiving block blk_5792489080791696128 src: /10.251.30.6:33145 dest:/10.251.30.6:50010
2018.1.24 09:18:17 PacketResponder 0 for block blk_-6952295868487656571 terminating
2018.1.24 09:19:19 Received block blk_5402003568334525940 of siz e 67108864 from /10.251.214.112
......
```

```
Event Template1: PacketResponder <*> for block <*> terminating
Event Template2: Received block <*> of siz e <*> from <*>
Event Template3: <*> Served block <*> to <*>
Event Template4: Receiving block <*> src: <*> dest: <*>
......
```

```
2018.1.24 09:15:13 Event Template1
2018.1.24 09:15:45 Event Template2
2018.1.24 09:16:22 Event Template3
2018.1.24 09:17:01 Event Template4
2018.1.24 09:18:17 Event Template1
2018.1.24 09:19:19 Event Template2
......
```

Figure 1.9: Log parsing example[13]

Feature Extraction

Once the logs have been collected and structured, the next crucial step in the anomaly detection process is feature extraction. At this stage, we have log messages in textual form, but to apply machine learning or deep learning techniques, it is necessary to transform them into a numerical representation.

Anomaly Detection

This step involves training an anomaly detection model. It is a crucial step where the numerical features are used to create a model capable of distinguishing normal logs from suspicious logs. This distinction is crucial for identifying abnormal behaviors in the logs and taking appropriate actions.

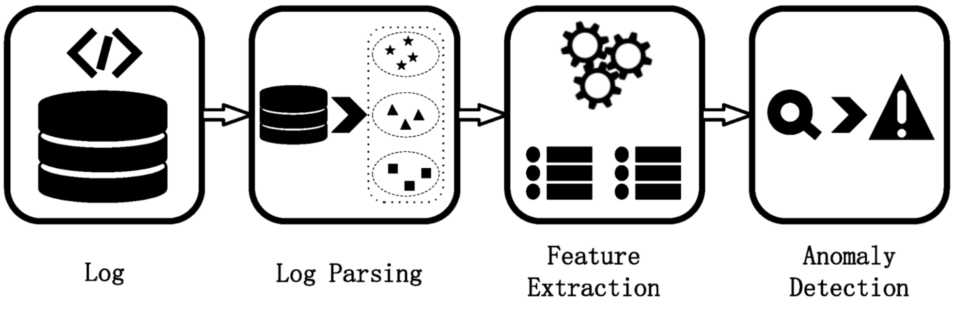


Figure 1.10: Architecture of an anomaly detection system[14]

The techniques used for anomaly detection in logs can be classified into supervised and unsupervised approaches:

Supervised

In supervised anomaly detection, when we have a dataset of labeled logs, we can train a deep learning or machine learning model knowing whether each log is normal or suspicious.

Unsupervised

Labeling logs can be challenging and time-consuming since there are often a large number of logs. In such cases, unsupervised techniques such as clustering are used to circumvent this problem. Clustering groups similar logs together and detects anomalies based on these groupings.

1.5 Conclusion

We have shed light on the fundamental principles of machine learning. We have explored various learning approaches, including supervised and unsupervised techniques, which enable us to extract valuable insights from log data. Additionally, we have discussed the crucial components of log collection, log parsing, feature extraction, and anomaly detection, highlighting their significance in achieving accurate and efficient detection of anomalies. By leveraging the power of machine learning, we can automate the process of log analysis and uncover hidden patterns or abnormal behaviors that may indicate system failures or security threats. We have set the stage for further exploration and experimentation, as we aim to use the capabilities of machine learning in log anomaly detection in our system.

Chapter 2

Methodology

2.1 Introduction

the methodology employed for the behavioral analysis in our system goes into needing to discuss the steps taken for preprocessing the data, which involved data cleaning and transformation to ensure its readiness for analysis. we explore the process of feature selection, where we identify and choose relevant features that contribute to the anomaly detection model and exclude the ones that are deemed unuseful for our particular case.

2.1.1 Our system's conception

Our conceptual system encompasses several interconnected components that form the foundation of our work: log data, data preprocessing, feature selection, model training, and behavioral analysis.

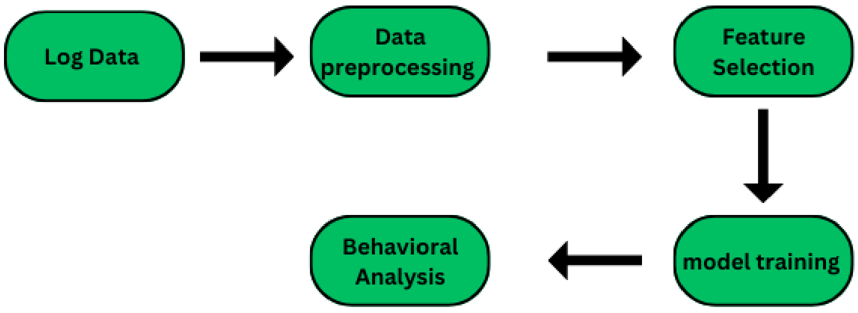


Figure 2.1: concept of anomaly detection system

2.2 Steps taken for preprocessing

2.2.1 Data Cleaning:

Remove duplicates, irrelevant or missing data. This ensures that our data is clean and ready for further analysis. First, we remove null data points found in the dataset. After visualizing and inspecting the data, we discovered that there are not only missing values but also values that are not usable (in our case filled with the character ”-”), which we also removed by deleting the rows. After this step, we are left with 747,169 usable log lines.

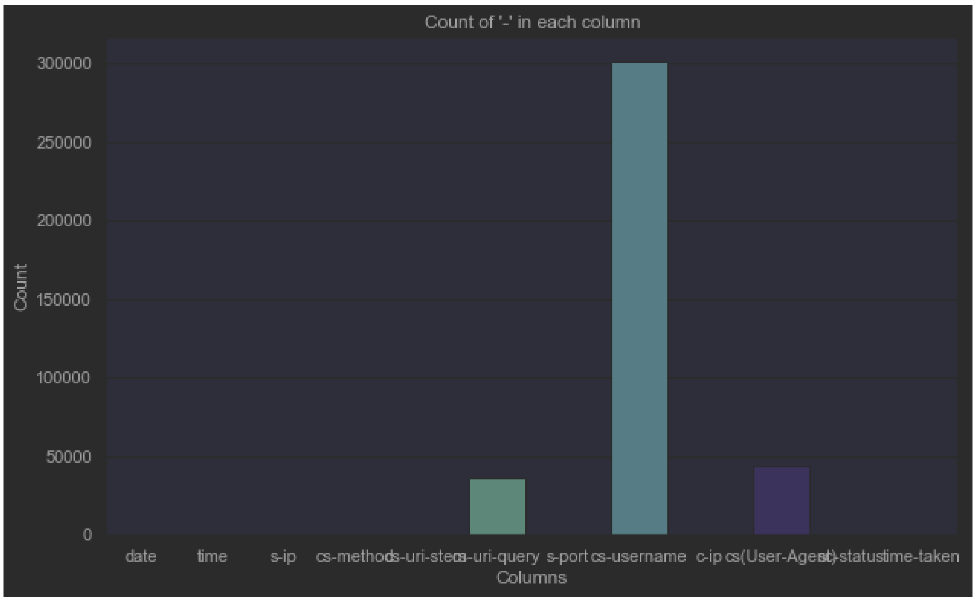


Figure 2.2: visualization of the number of rows with “-” character found in each column

2.2.2 Data Transformation:

Since we have unlabeled data, we will transform it based on the specific use of unsupervised models. This will include normalization and conversion of categorical variables into numerical variables. For the ”method” column, we used one-hot encoding to encode the distinct values of the method. One-hot encoding involves creating binary columns for each unique category in a categorical feature, with a value of 1 indicating the presence of that category and 0 indicating its absence. This technique is used to convert categorical data into a numerical format that can be used in machine learning algorithms. For columns where we have value ranges, we used scaling to normalize the values, which involves transforming the feature values into a standardized range, typically between 0 and 1. This technique is used in machine learning to ensure that features with different ranges are on the same scale, which helps algorithms perform better. The normalization formula involves subtracting the minimum value of the feature from each value, dividing by the range (the difference between the maximum and minimum values), and then scaling to the desired range. While normalization can remove the effect of outliers, it can also result in smaller standard deviations [15].

For IP addresses, we used a mechanism to convert them into integer values.

For columns where we have text that requires more than the previous techniques offered, there are different techniques that can be used depending on the specific nature of the data. A common technique is to use text preprocessing methods such as tokenization, stemming, and lemmatization[16] . Tokenization involves splitting the text into individual words or tokens, while stemming and lemmatization involve reducing each word to its base or root form. The technique we used is text vectorization, such as bag of words or term frequency-inverse document frequency (TF-IDF), to represent the textual data numerically. These techniques can help reduce the dimensionality of the data and improve the performance of machine learning models that use textual data as input[17] .

After representing the text with a vectorization technique, we would obtain a large vector that would take up too much space and require too many resources, so we need to reduce these vectors using dimensionality reduction algorithms such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-Distributed Stochastic Neighbor Embedding (t-SNE). But as the use of dimensionality reduction can affect variance, it is important to choose an appropriate method that preserves the integrity of the data. One such method is Principal Component Analysis, which can reduce the dimensionality of the data while retaining the most important information. Careful analysis of variance is essential for selecting an optimal value for our model, balancing complexity and accuracy to ensure robustness without sacrificing performance.

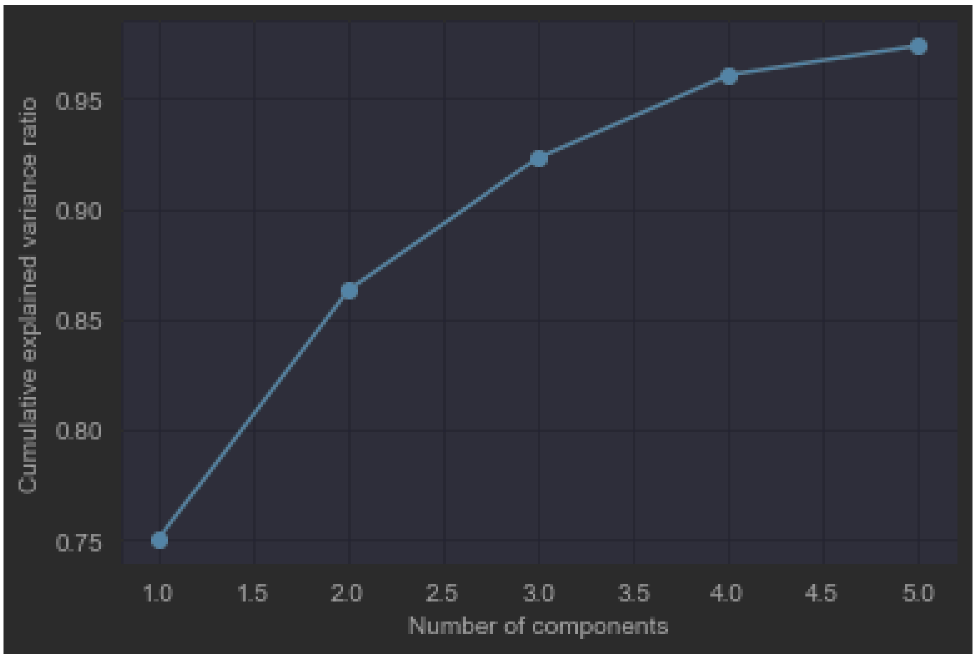


Figure 2.3: visualization of the number of components and the variance ratio

2.2.3 Feature Selection:

Select only relevant features that contribute to the anomaly detection model. This helps reduce the dimensionality of the dataset and improves the performance of the model.

After using the initial collected data and conducting experiments and observations, we concluded that the use of these 6 columns would be sufficient in terms of variation and importance to build robust training data:

datetime:

The datetime column indicates the timestamp of each log entry. It can be important for analyzing trends and patterns over time. For example, we could identify anomalies or detect specific periods when certain activities occur more frequently, or when the user logs in at unusual times.

cs-uri-stme:

The cs-uri-stem column represents the root part of the requested Uniform Resource Identifier (URI). It provides information about the specific webpage or resource that was requested. Analyzing this column can help identify popular pages, spot potential vulnerabilities or attack patterns, and gather insights into user behavior.

cs-username:

The cs-username column contains the username associated with each log entry. It can be valuable for user-specific analysis and monitoring. Since our main objective is user behavioral analysis, this column is of crucial importance.

cs(User-Agent):

The cs(User-Agent) column captures the user agent string, which provides information about the client device or browser used to access the web application. Analyzing this column allows us to understand the distribution of client devices or browsers and detect abnormal or malicious user agent patterns.

sc-status:

The sc-status column represents the status code returned by the server in response to a request. It indicates the success or failure of the request. Analyzing this column helps determine error-prone behaviors. By examining different status codes such as 200 for success, 404 for not found, or 500 for internal server error, we can identify abnormal behaviors that require special attention. This can contribute to improving system quality and reliability by addressing recurring issues and optimizing performance.

Time-taken:

The time-taken column records the duration or time taken for server processing of a request. It provides information about the performance of the web application and can be useful for identifying slow or resource-consuming requests, optimizing system performance, and detecting anomalies related to response time. By analyzing this column, we can identify abnormally long response times, bottlenecks, or system usage peaks, which can contribute to performance optimization and improving the user experience.

As for the non-selected columns:

IP addresses:

IP addresses were not selected because they are not relevant for our analysis due to the low number of distinct IP addresses available in the dataset. Without a significant quantity of distinct IP addresses, it is difficult to extract useful information or detect IP-related anomalies. Therefore, we chose not to include these columns in our analysis.

s-port:

The s-port column, with only two values, exhibits low variance, making it less useful for our analysis. Since there is no significant diversity in the values of this column, it does not provide discriminative information for detecting patterns or anomalies. Therefore, we decided not to include this column in our analysis.

Cs-method:

The HTTP request method used is not a determining factor for identifying abnormal patterns or behaviors. Therefore, we chose not to include this column in our analysis.

cs-uri-query:

The cs-uri-query (URI query) column was not selected as it is redundant for our analysis. This column contains information combined from the cs-uri-stem and cs(User-Agent) columns. Additionally, the size of the cs-uri-query column is too large to be efficiently encoded in our analysis. Therefore, we chose not to use this column in our study.

2.2.4 2nd Set of Collected Data:

2.2.5 Preprocessing Steps:

Data Cleaning:

Similar to the first time, we had to perform the same data cleaning steps, which left us with 3 million usable logs in this dataset.

Data Transformation:

We used the same steps, but not for each column, as in this case, we have a lot more data, and therefore, more dimensions to consider. So, a change in the dimensionality reduction algorithm is necessary since the PCA algorithm poses a memory demand issue that is not available on our machine. Therefore, we opted for an algorithm that would give us the necessary dimensionality reduction while avoiding memory shortage, which is the TruncatedSVD algorithm. TruncatedSVD has other advantages in our case, especially when the output from the vectorization using the TF-IDF algorithm is sparse. TruncatedSVD is more optimal in this case as it is specifically designed to handle sparse matrices and performs a truncated singular value decomposition that retains the most important features of the original matrix. This allows us to efficiently reduce the dimensionality of our data while preserving the most significant information for our anomaly detection task.

2.2.6 Discussion of Challenges Faced During Data Collection and Preprocessing:

Data Labeling Unavailability

Anomaly detection often requires labeled data to train and evaluate algorithms. However, in our research, the dataset provided by Sonatrach was unlabeled, which posed a major challenge. The absence of labeled anomalies made it difficult to develop supervised anomaly detection models. Therefore, alternative approaches had to be explored to effectively address the problem.

Limited Computing Power

Another challenge was the limited computing power available for data processing and analysis. Anomaly detection on large datasets can be computationally intensive, requiring significant computing resources. Due to the constraints of the machine used, processing time and memory limitations posed additional difficulties in handling and analyzing the dataset.

Data Noise and Inconsistencies

Raw log data often contains noise and inconsistencies, making it difficult to extract meaningful information. Log entries may have missing or erroneous values, variable formats, or irrelevant data fields. These inconsistencies had to be addressed during the preprocessing phase to ensure accuracy and reliability in the subsequent anomaly detection analysis.

High-Dimensional Feature Space

Log data can be high-dimensional, with numerous features contributing to the complexity of the dataset. The large number of dimensions can lead to issues such as the curse of dimensionality, increased computational requirements, and reduced algorithm performance. Dimensionality reduction techniques had to be employed to mitigate these challenges and extract the most informative features for anomaly detection.

Trade-offs in Data Preprocessing

Preprocessing decisions involved trade-offs between preserving important information and reducing noise and irrelevant features. Determining the optimal balance between data cleaning, feature selection, and dimensionality reduction was challenging, as different preprocessing techniques can affect the performance and interpretability of anomaly detection algorithms in different ways.

To overcome these difficulties, a deep understanding of the data characteristics, expertise in the field, and experimentation with different preprocessing techniques were necessary. By carefully addressing these challenges, our goal was to create a high-quality processed dataset capable of effectively detecting anomalies and supporting accurate anomaly detection in the subsequent stages of our research.

2.3 Explanation of Algorithm Selection

The decision to use the clustering algorithms k-means, DBSCAN, GMM, and isolation forest in this study was motivated by the need to effectively analyze an unlabeled dataset using reliable clustering techniques. Due to their relevance to unsupervised learning tasks and their ability to identify underlying patterns in the data, these algorithms were specifically chosen.

The k-means clustering method was selected for its simplicity and efficiency in dividing data points into coherent clusters based on their proximity. It attempts to minimize the sum of squared intra-cluster distances by iteratively updating the cluster centroids, making it easy to distinguish distinct groups.

DBSCAN, on the other hand, excels in locating dense areas in the dataset while effectively handling outliers and noise. It defines clusters as dense regions containing a minimum number of neighboring points, enabling it to capture clusters of all sizes and shapes.

GMM (Gaussian Mixture Model), a probabilistic model, was chosen because it can represent each cluster as a Gaussian distribution, allowing it to capture complex data distributions. With the ability to handle clusters of varied shapes and densities, it provides a more nuanced view of the underlying structure of the data.

Additionally, the Isolation Forest algorithm was used to separate outlier instances by constructing random binary decision trees as an anomaly detection tool. It assigns anomaly scores and separates outlier values from typical data points by calculating the average path length required to isolate an instance.

By incorporating these different clustering algorithms into our methodology, we sought to gain useful insights, identify significant patterns, and spot potential anomalies within the unlabeled dataset. This enhanced our understanding of the underlying data distribution and contributed to the field of anomaly detection.

2.4 Conclusion

This chapter has provided a comprehensive overview of the methodology employed in our log anomaly detection system. We have discussed the preprocessing steps involved in preparing the data for analysis, including data cleaning to remove duplicates and irrelevant information, as well as data transformation techniques such as normalization and one-hot encoding. The use of dimensionality reduction algorithms, such as Principal Component Analysis (PCA) and TruncatedSVD, has been highlighted to effectively reduce the dimensionality of the data without compromising its integrity. We have also addressed the challenges encountered during data collection and preprocessing, such as the absence of labeled anomalies, limited computing power, data noise, and inconsistencies, and the high-dimensional feature space. Overcoming these challenges required careful consideration and experimentation with various preprocessing techniques.

Chapter 3

Implementation

3.1 Introduction

In this chapter, we talk about the tools, libraries, and algorithms we utilized to build our anomaly detection system. We describe the dataset that was used in our project. To provide insights into the dataset's distribution and patterns, visualizations of user logs by distinct characteristics are presented. We describe how the algorithms used for anomaly detection work and how parameters are chosen.We examine the benefits and drawbacks of each approach and show visuals of the clustering findings.

3.2 Tools and Libraries Used:

Python:

We used the Python programming language as the foundation of our project. Python offers a wide range of libraries and frameworks that are well-suited for data analysis and machine learning tasks[20].

Scikit-learn:

Scikit-learn is a popular machine learning library in Python that provides a comprehensive set of tools for data preprocessing, clustering, and anomaly detection. We leveraged its functionalities to implement the algorithms discussed in Chapter 3[21].

NumPy and Pandas:

NumPy and Pandas are two essential Python packages for data manipulation and analysis. Pandas offers data structures and functions for handling structured data, such as dataframes, while NumPy supports efficient numerical computations. These libraries allowed us to preprocess and efficiently organize the log data[22], [23].

Matplotlib and Seaborn:

Matplotlib and Seaborn are visualization libraries in Python that enable us to create various types of graphs and charts. We used these libraries to visualize the results of our analysis, providing a clear understanding of detected anomalies and observed patterns[24], [25].

PyQt5:

A robust framework called PyQt is used to create graphical user interfaces (GUIs). It enables developers to easily produce interactive and aesthetically pleasing applications. PyQt offers a smooth user experience with a variety of built-in widgets and numerous customization options. By using PyQt, developers can create user-friendly interfaces that enhance the functionality of their applications[26].

3.2.1 Work Environment:

Table 3.1: Machine Specifications

Machine	**OS**	**CPU**	**GPU**	**RAM**
Laptop	Windows 10	i7-7700HQ	Nvidia GTX 1060	16GB

3.2.2 Description of the dataset in our project

Our dataset was collected from a Windows Exchange server(2016) at the headquarters of Sonatrach over a period of 24 hours, which gave us 1 million lines of logs. The columns found in our dataset are as follows:

date:

The date on which the event occurred, in the format YYYY-MM-DD.

Time:

The time at which the event occurred, in the format HH:MM:SS.

S-ip:

The IP address of the server that processed the request.

Cs-method:

The HTTP method used for the request (`GET`, `POST`, `PUT`, `HEAD`, `OPTIONS`, `RPC_IN_DATA`, `RPC_OUT_DATA`).

Cs-uri-stem:

The part of the URI of the request, which usually specifies the accessed endpoint (e.g., /api/users).

Cs-uri-query:

The query string of the request, which typically contains additional parameters or filters.

S-port:

The port number on which the server processed the request.

cs-username:

The username associated with the request, if any.

C-ip:

The IP address of the client that made the request.

cs(User-Agent):

The client's user agent string, which usually specifies the type and version of the client software used.

sc-status:

The HTTP response status code (e.g., 200, 404, 500).

Time-taken:

The time (in milliseconds) taken to respond to the request. -After using the log data from a single day and having only 1 million logs available for training, we requested more data, and a dataset spanning 4 days was collected for our use, consisting of 4 million log entries.

3.2.3 Visualization of each characteristic distribution in our dataset by user:

As part of our analysis, we present a visualization of our dataset. The dataset consists of five key features: datetime, HTTP response, software used, endpoint, and time taken. These features offer valuable insights into the activities and interactions within the network.

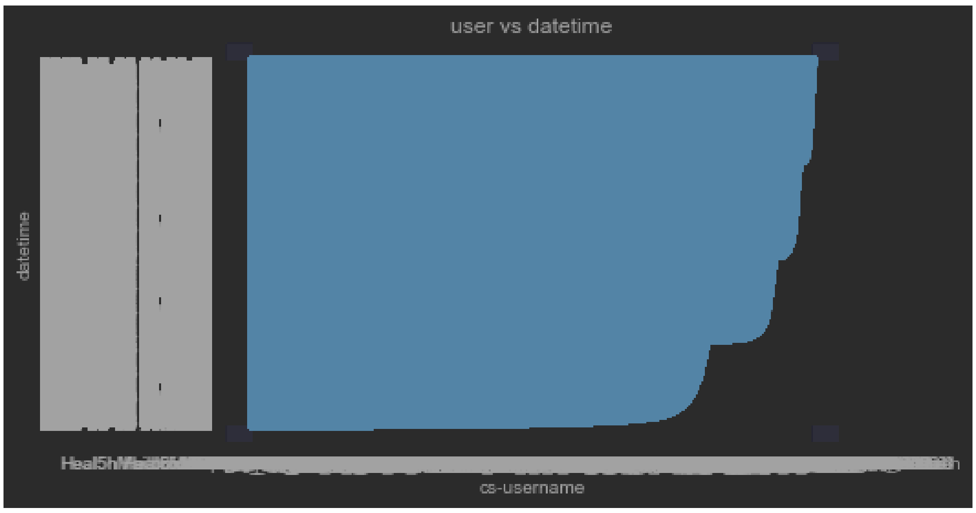


Figure 3.1: Users logs by date and hour

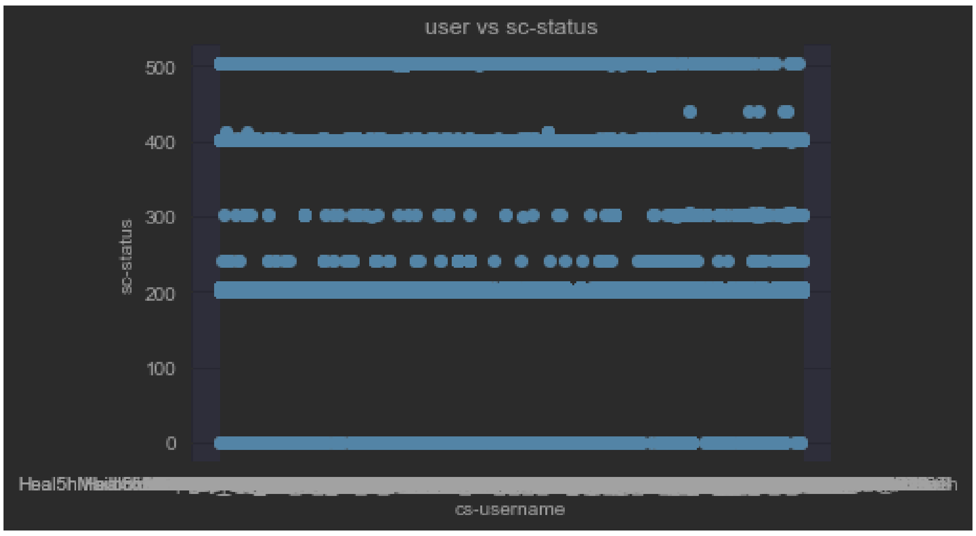


Figure 3.2: HTTP responses by user's logs

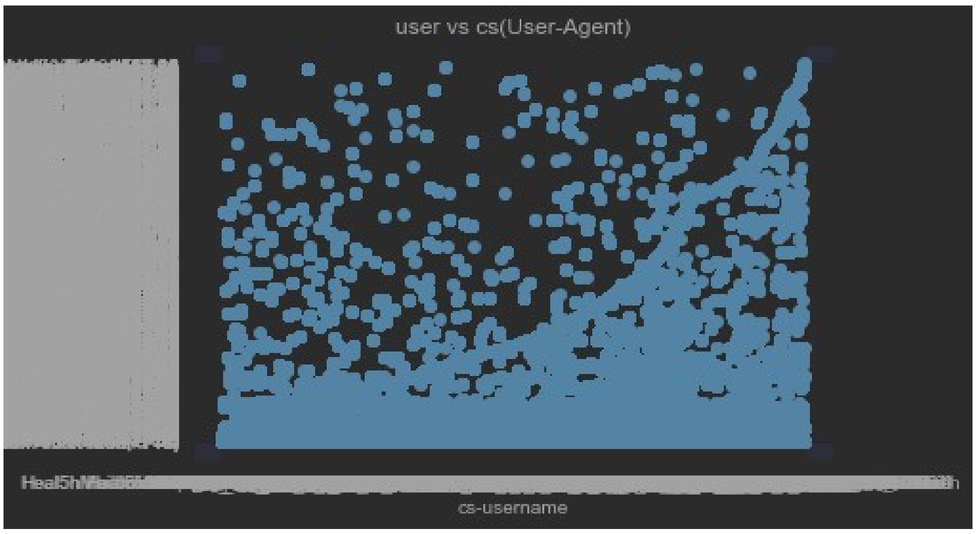


Figure 3.3: Users logs by software used

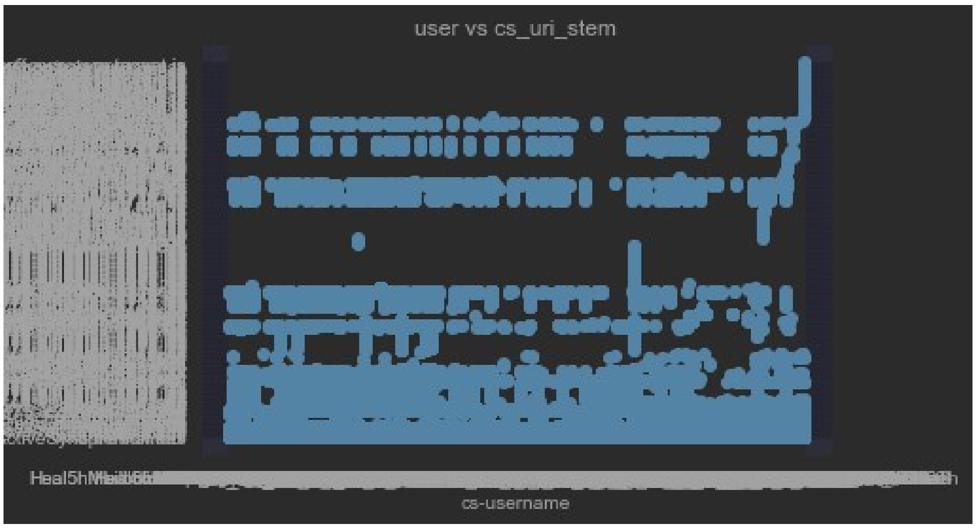


Figure 3.4: Users logs by endpoint

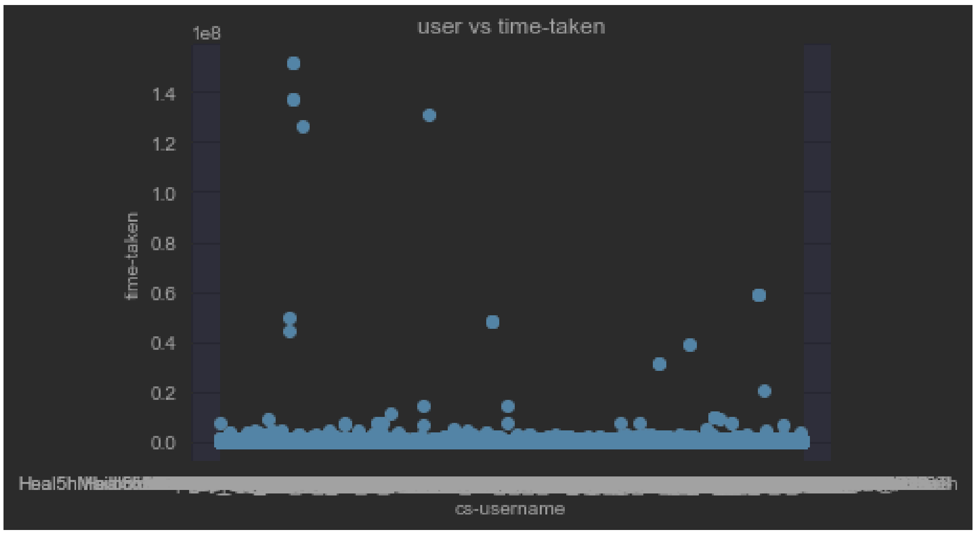


Figure 3.5: Time taken for each log by users

3.3 Description of the algorithms used for anomaly detection in our logs

3.3.1 The functioning of K-means is as follows:

Initialization: Randomly select k data points to serve as initial cluster centroids.

Assignment: Determine the distance between each data point and the cluster centroids. Each data point is assigned to the cluster with the nearest centroid.

Recalculation of centroids: After assigning each data point to a cluster, calculate new centroids by taking the average of the coordinates of all data points belonging to each cluster.

Iteration: Repeat steps 2 and 3 until a termination condition is met, such as when the centroids no longer change significantly or when the maximum number of iterations is reached.

Parameter selection:

To choose a value of k for K-means, we would calculate the Sum of Squared Errors (SSE), also known as the Within-Cluster Sum of Squares (WCSS). SSE is calculated by summing the squared distances between each data point and the centroid of the cluster it is assigned to. It measures the total variance in the data explained by the clusters. Specifically, SSE is calculated as follows:

$$SSE = \sum (\text{distance}(\text{point}, \text{centroid}))^2$$

A lower SSE value indicates that data points are closer to their respective centroids, which is desirable as it indicates better intra-cluster cohesion. On the other hand, a higher SSE value may indicate significant dispersion of data points or poor partitioning.

After obtaining SSE values for a range of k values, we plot the values and use the elbow method, which involves finding the point on the graph that resembles an elbow.

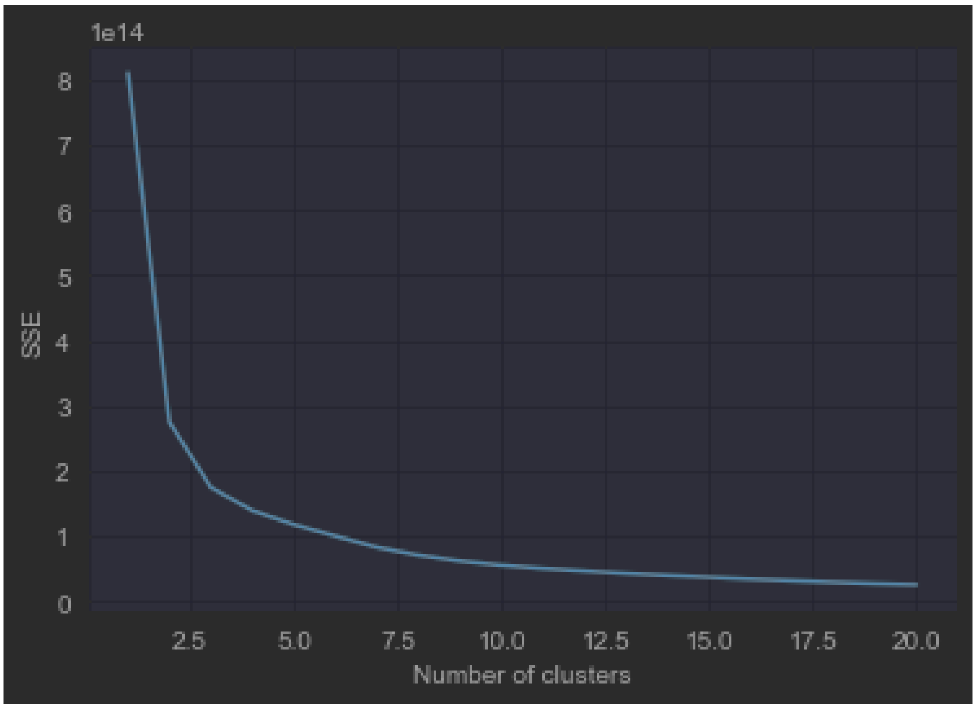


Figure 3.6: a graph of SSE values by cluster

Analysis of the SSE graph: It appears that there is a minimal decrease in SSE values from k values 2 to 3, which we could consider as the optimal value of k.

Advantages of applying K-means: - Ease of implementation: K-means is a straightforward algorithm to use. - Computational efficiency: K-means can handle large datasets and is computationally efficient.

Limitations of K-means include: - K-means assumes that clusters have isotropic shapes with similar variation in all dimensions. However, anomalies often have different shapes and variations, making reliable identification by K-means challenging. - K-means struggles to handle outliers as it treats them as normal data points and assigns them to the nearest cluster. Outliers can weaken the representation of the cluster or create new clusters, resulting in poor anomaly detection efficiency. - K-means is limited to linear boundaries as it assumes clusters can be divided by these boundaries. Anomalies, however, can exhibit complex and nonlinear correlations that K-means cannot detect. Due to this limitation, K-means struggles to correctly identify anomalies in our dataset.

3.3.2 The functioning of DBSCAN:

When K-means fails to handle non-convex problems, DBSCAN, also known as Density-Based Spatial Clustering of Applications with Noise, is a powerful technique that succeeds. The idea is simple: a cluster is a region of high density that can take any shape and is surrounded by a region of lower density. There is no need to determine in advance how many clusters are expected as this statement is generally true. The process starts by examining a small area (technically, a point surrounded by the minimum number of possible samples). If there is sufficient density, it is considered part of a cluster.

Epsilon ( ) and min_samples are two essential factors on which DBSCAN relies: The maximum distance or radius within which points are considered neighbors is defined by the epsilon (eps) parameter. It effectively specifies the level of spatial density at which a point is considered a member of a cluster. The granularity of the clusters found by DBSCAN can be adjusted by varying the value of eps. A larger epsilon value allows the inclusion of more distant points, resulting in larger clusters, while a smaller epsilon value leads to tighter and more compact clusters.

However, min_’samples indicates the minimum number of neighbors a point must have to be classified as a core point. Core points in the DBSCAN algorithm are crucial as they serve as the foundation for cluster creation. A point is considered a core point if it has min_samples neighbors within its epsilon neighborhood. Any point within an epsilon radius of a core point is also considered a member of the same cluster. We can control the minimum density threshold required to identify a cluster by modifying min_samples. By increasing min_samples, we will obtain the identification of denser and more significant clusters.

Unlike other algorithms used in our analysis, DBSCAN posed higher resource requirements, making it difficult to efficiently explore a wide range of parameter combinations. Despite these limitations, we conducted several experiments with different parameter values to discover potential clusters.

Choice of parameters:

The process of selecting parameters for DBSCAN often involves some trial and error as the algorithm does not have a built-in method for determining the best parameter values. Due to the absence of a precise method for determining the best parameter settings, it is necessary to conduct experiments and rely on domain expertise to guide the parameter selection process.

3.3.3 The functioning of GMM:

Due to their ability to simulate complex data distributions, Gaussian Mixture Models (GMMs) have proven to be useful tools in anomaly detection. A probabilistic model called GMM assumes that the data has been generated using a combination of Gaussian distributions. Each Gaussian component in the data indicates a cluster or group.

Choice of parameters:

The parameters used for the GMM algorithm were configured to obtain two distinct clusters, one representing normal data and the other representing abnormal data. The chosen value for the parameter k was 2 because we wanted to have exactly two clusters. This allowed for clear differentiation between normal observations and those considered abnormal. By having these two distinct clusters, it was easier to take targeted actions based on the nature of the detected anomalies. This approach resulted in more accurate results and a better understanding of the different behaviors present in the data.

3.3.4 The functioning of Isolation Forest:

An anomaly detection approach called Isolation Forest isolates outlier values in a dataset. It works by building a random forest of isolation trees, where each tree is constructed by randomly selecting a feature and then splitting the data on a randomly chosen value within the range of that feature. Each data point receives an anomaly score from the algorithm based on the number of splits required to isolate it.

In comparison to other algorithms, the main advantage of the Isolation Forest algorithm lies in its efficiency in detecting anomalies, especially in high-dimensional datasets. It is an unsupervised learning technique that does not require labeled training data, which is beneficial in our situation. Additionally, Isolation Forest is not affected by the presence of irrelevant features and can handle outliers of all sizes and shapes.

Choice of parameters:

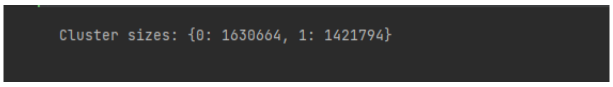
When using Isolation Forest for anomaly detection, there is an important parameter known as the threshold. This parameter plays a crucial role in Isolation Forest as it determines the cutoff point for classifying instances as normal or abnormal. By setting an appropriate threshold, we can control the sensitivity of the algorithm in detecting anomalies. A lower threshold will classify more instances as anomalies, while a higher threshold will detect fewer anomalies.

Determining a good threshold when we don't have ground truth can be difficult and highly subjective.

3.4 Results of Each Algorithm:

3.4.1 K-means:

In our evaluation, the k-means algorithm performed poorly in both anomaly detection and clustering analysis. Simply dividing the log file into two without considering the subtleties of the data proved to be insufficient for our objectives. Therefore, we decided to exclude k-means from our actual detection process.

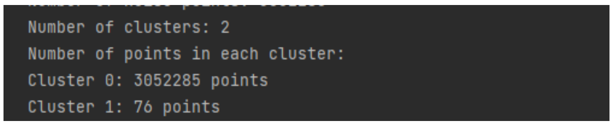


```
Cluster sizes: {0: 1630664, 1: 1421794}
```

Figure 3.7: clusters by k-means

3.4.2 DBSCAN:

We used the DBSCAN clustering algorithm to identify patterns and anomalies in the logs. Applying DBSCAN resulted in the formation of two clusters, one of which was significantly smaller than the other. This result poses a challenge as it suggests that the algorithm may have missed some irregularities present in the data.



```
Number of clusters: 2
Number of points in each cluster:
Cluster 0: 3052285 points
Cluster 1: 76 points
```

Figure 3.8: clusters by dbscan

Distribution of suspicious connections according to DBSCAN clustering:

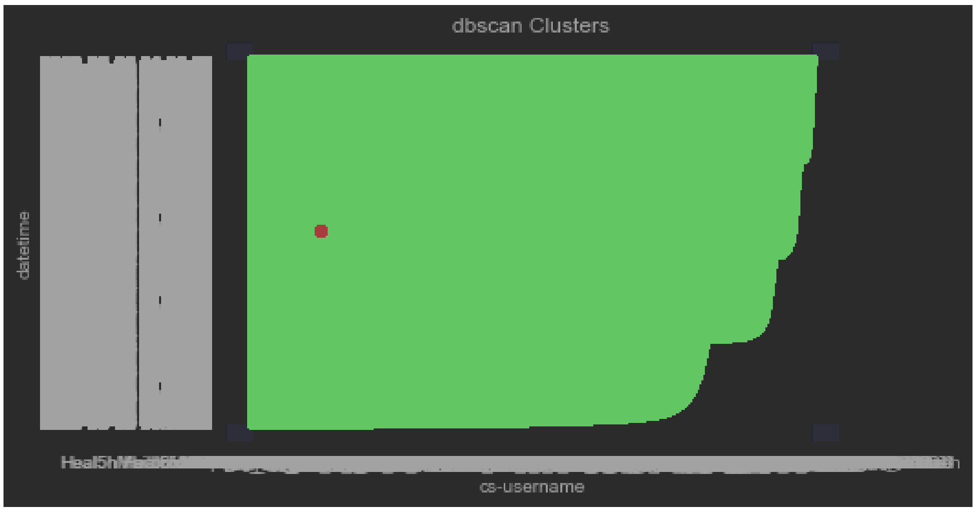


Figure 3.9: DBSCAN clustering users by datetime logs

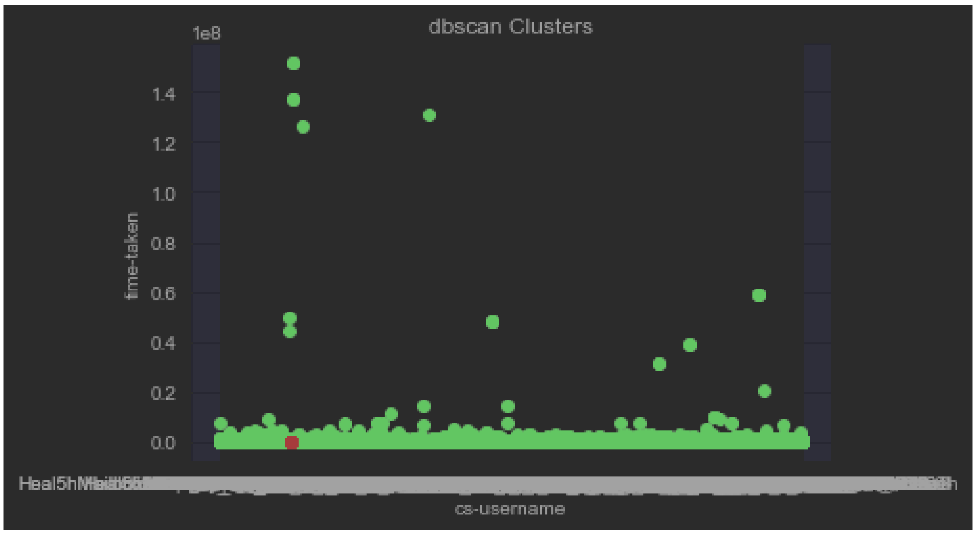


Figure 3.10: DBSCAN clustering users by time-taken logs

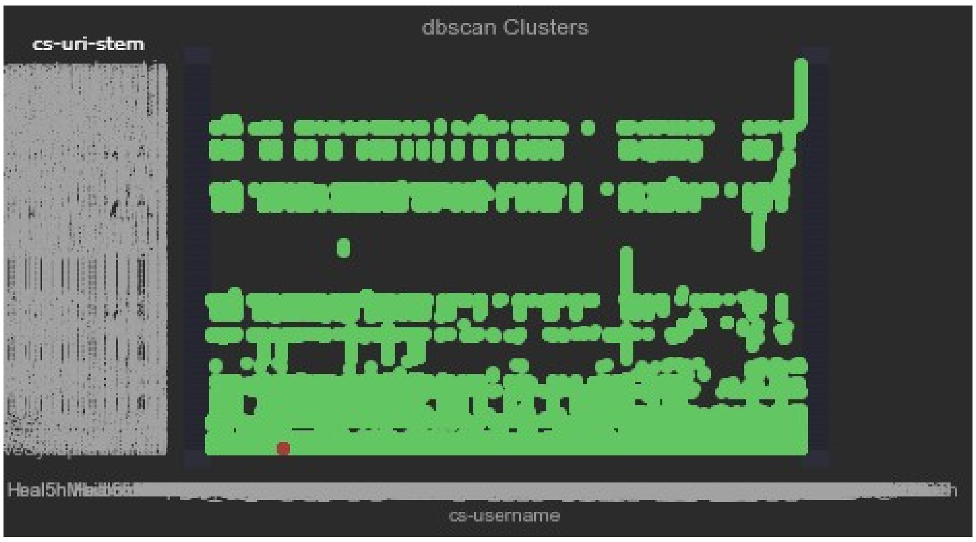


Figure 3.11: DBSCAN clustering users by endpoint logs

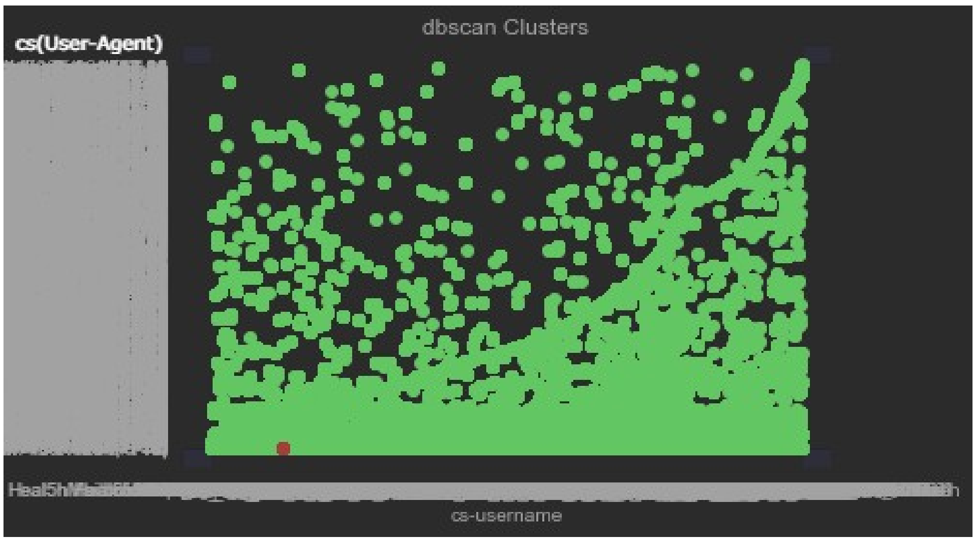


Figure 3.12: DBSCAN clustering users by software logs

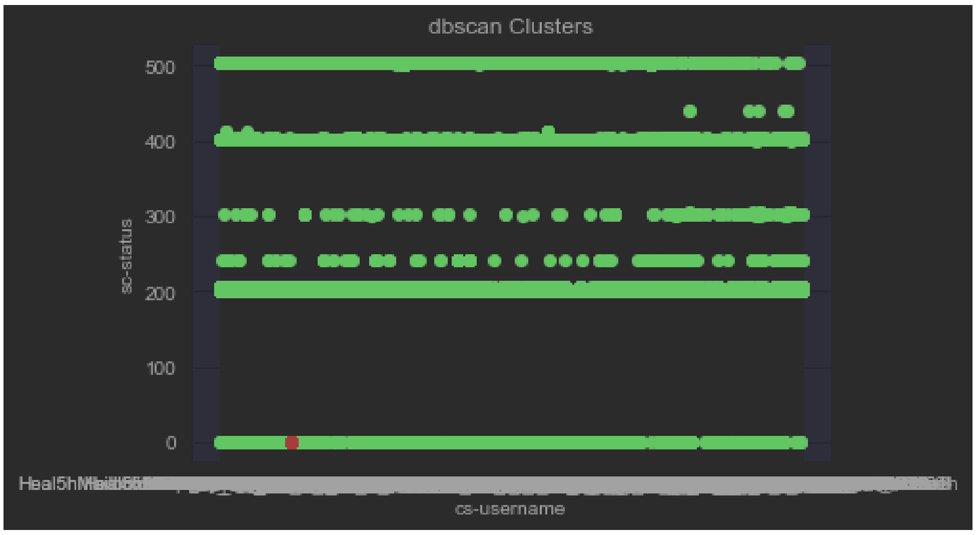
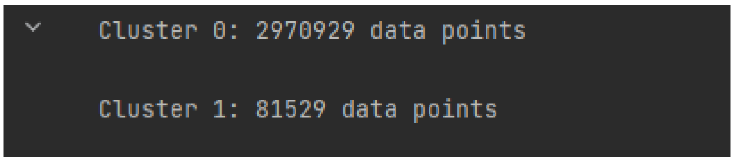


Figure 3.13: DBSCAN clustering users by HTTP status response logs

3.4.3 GMM:

On the other hand, when we applied the Gaussian Mixture Model (GMM) to our log data, the results were quite different. GMM successfully detected a larger number of logs as anomalies compared to DBSCAN. This suggests that GMM has the potential to capture a wider range of irregularities and anomalies in the data. Such capability could be highly beneficial in the process of anomaly detection and resolution, as it provides a more comprehensive understanding of system behavior.



```
Cluster 0: 2970929 data points

Cluster 1: 81529 data points
```

Figure 3.14: GMM clusters

Distribution of suspicious connections according to GMM clustering:

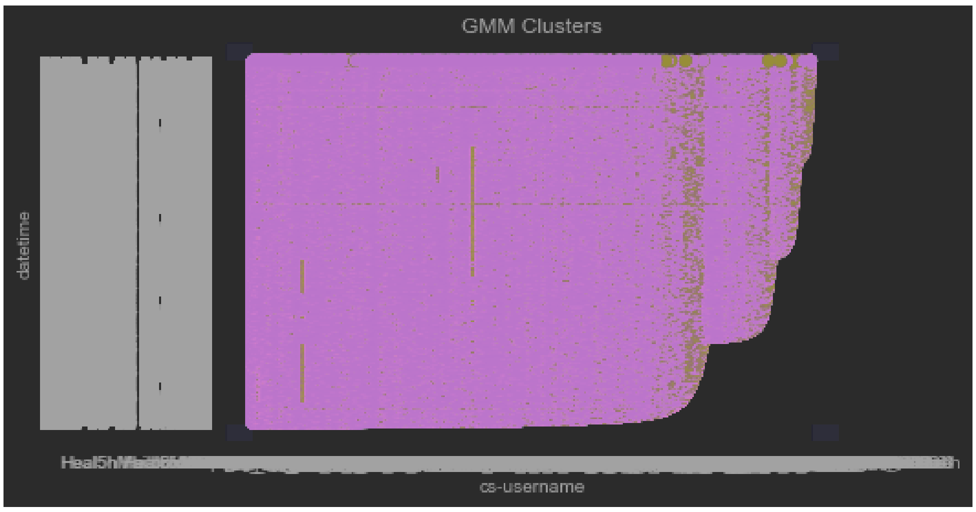


Figure 3.15: GMM clustering users by datetime logs

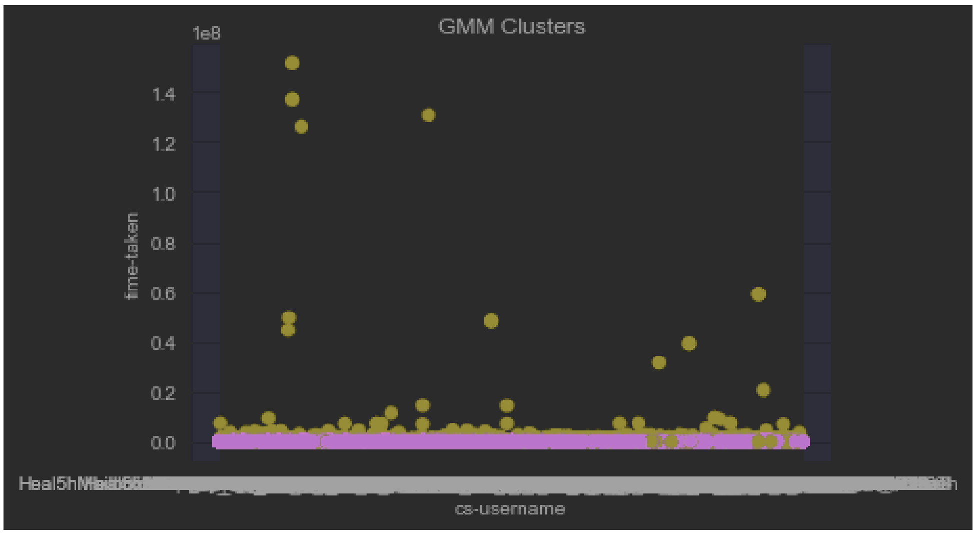


Figure 3.16: GMM clustering users by time-taken logs

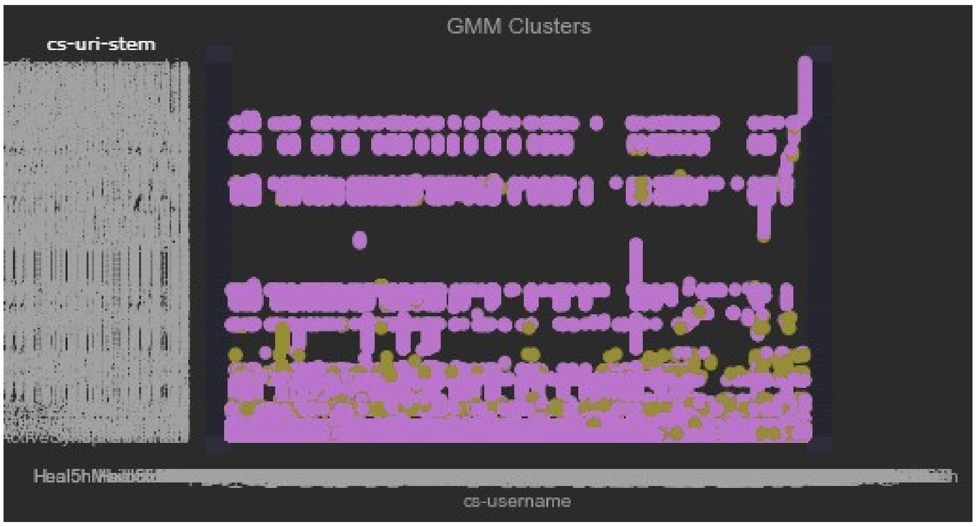


Figure 3.17: GMM clustering users by endpoint logs

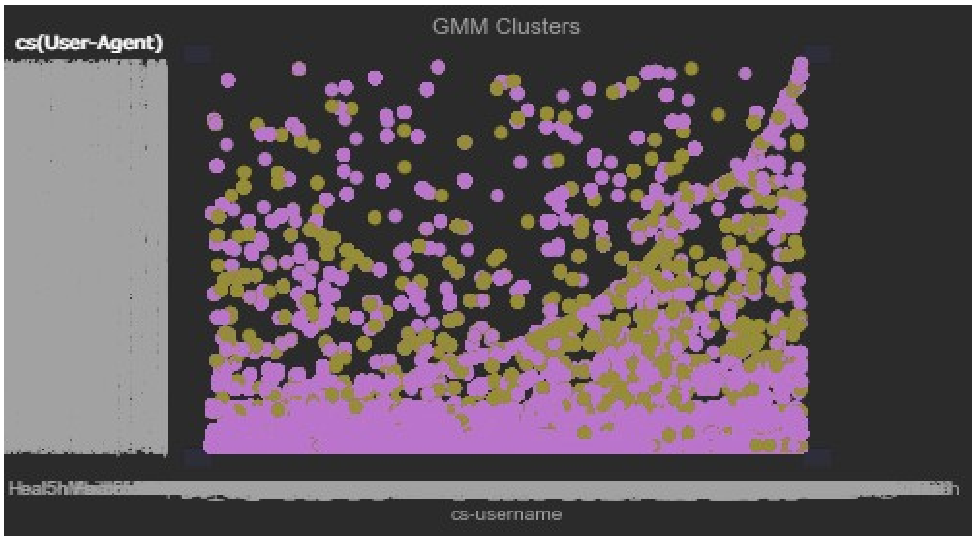


Figure 3.18: GMM clustering users by software logs

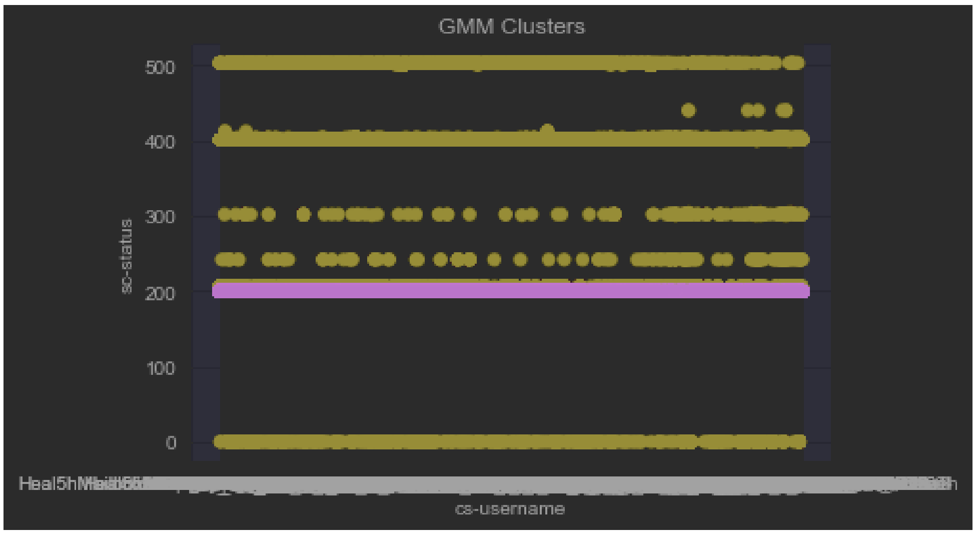
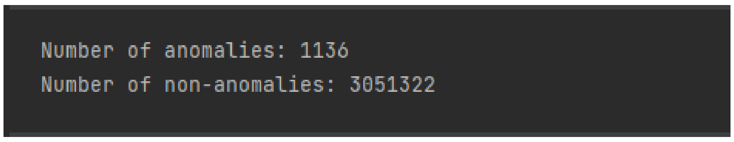


Figure 3.19: GMM clustering users by HTTP status response logs

3.4.4 Isolation Forest:

The Isolation Forest algorithm proved to be more sensitive in capturing a greater number of anomalies compared to DBSCAN. However, it did not show the same level of sensitivity as GMM, which detected even more irregularities in the log data. A noteworthy observation is that the anomalies detected by the Isolation Forest did not follow a recognizable pattern or distribution.



```
Number of anomalies: 1136
Number of non-anomalies: 3051322
```

Figure 3.20: isolation forest clusters

Distribution of suspicious connections according to Isolation Forest clustering:

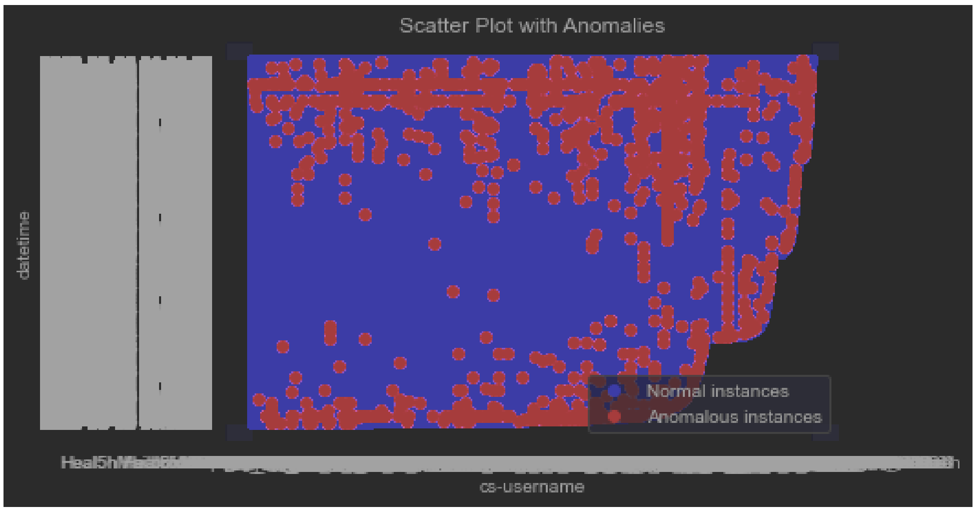


Figure 3.21: Isolation Forest clustering users by datetime logs

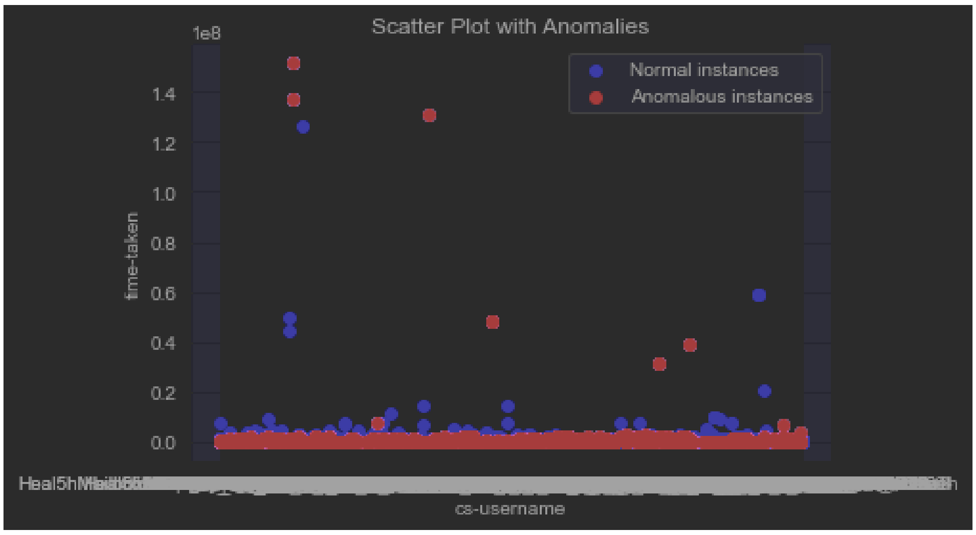


Figure 3.22: Isolation Forest clustering users by time-taken logs

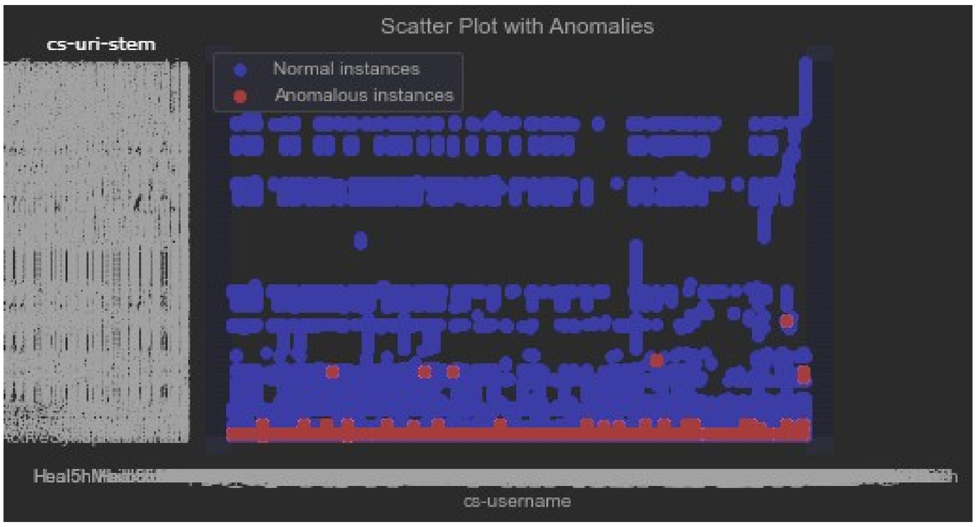


Figure 3.23: Isolation Forest clustering users by endpoint logs

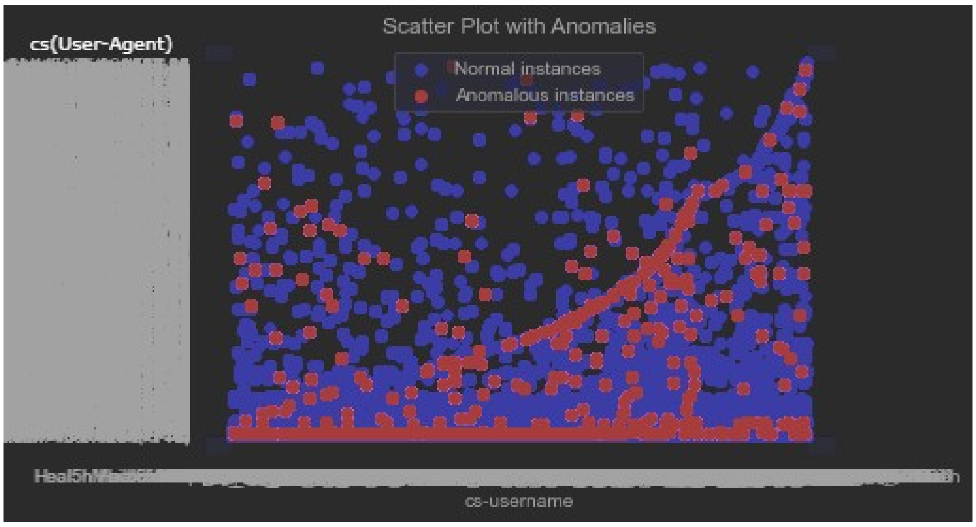


Figure 3.24: Isolation Forest clustering users by software logs

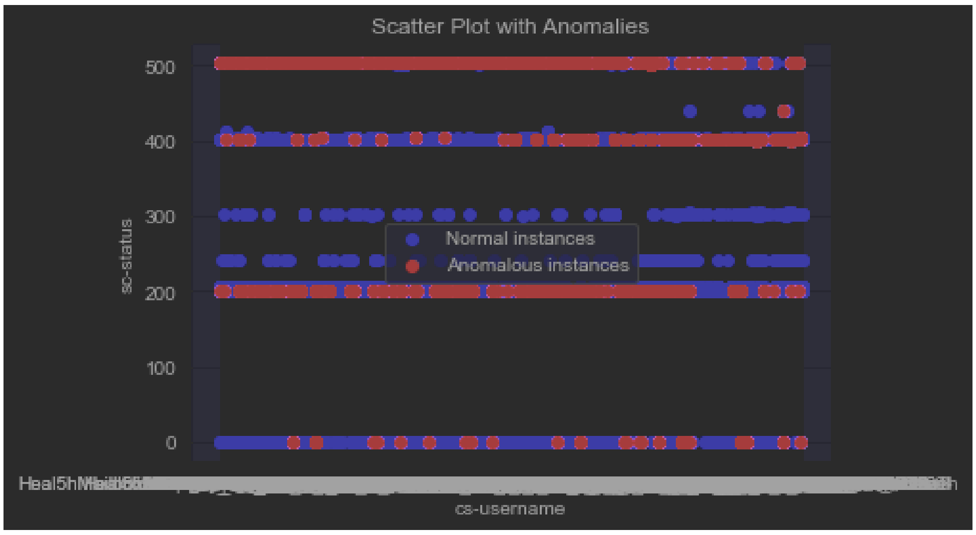


Figure 3.25: Isolation Forest clustering users by HTTP status response logs

3.5 The Application:

Our application is simple: it takes a log file as input, and you have the choice to apply one of the three used algorithms. Depending on the chosen algorithm, you will receive a list of users whose abnormal behavior has been detected. Once the algorithm is applied to the log file, our application extracts the users who exhibited abnormal behavior. These users can then be further examined to understand the reasons behind their atypical behavior.

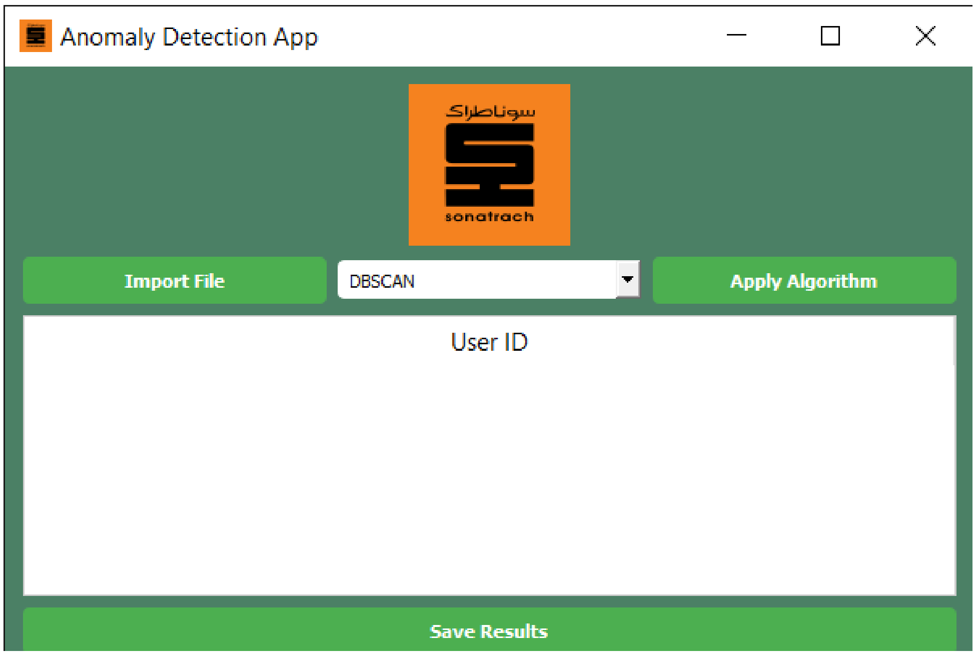


Figure 3.26: anomaly detection application GUI

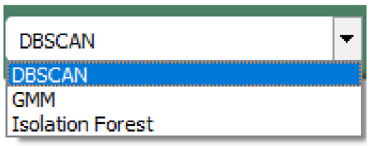
Import File:

Import a log file in Excel format for use by the application.

Apply Algorithm:

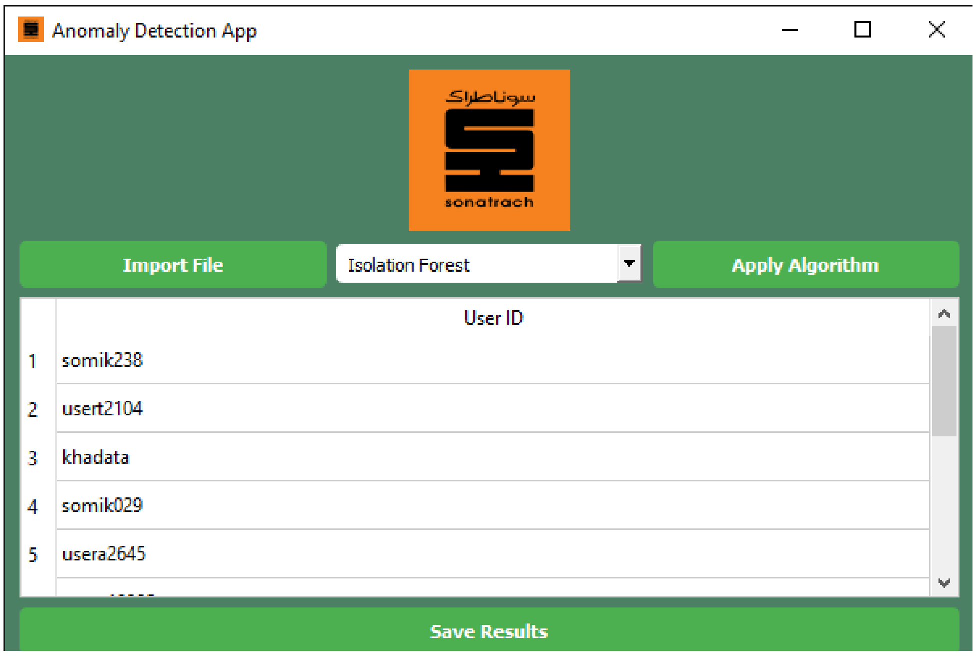
Apply the selected algorithm to the log file to detect anomalies.

Available Algorithms:



Example of Application Operation:

After importing a log file and applying an algorithm, users with abnormal behavior are displayed. To analyze the abnormal logs separately in more detail, they can be saved to an Excel file.



	User ID
1	somik238
2	usert2104
3	khadata
4	somik029
5	usera2645

Figure 3.27: Example result of the application

3.6 Conclusion:

In this chapter, we presented the algorithms and the parameters used as well as the results and analysis of our project,. We discussed the libraries and tools used throughout our experimentation process and examined the obtained results. Our main objective was to gather information about patterns, anomalies, and potential cybersecurity threats present in the dataset. To showcase the application of the algorithms, we developed a simple GUI-based application that takes a log file as input and applies the chosen algorithm to detect abnormal user behavior. The application provides a list of users exhibiting abnormal behavior, which can be further examined to understand the reasons behind their atypical actions. In conclusion, we explored and analyzed multiple anomaly detection and clustering algorithms in log data. Through our evaluation, we found that DBSCAN, GMM, and Isolation Forest offer varying levels of performance and sensitivity.

General Conclusion

Detecting anomalies and efficiently analyzing logs in large companies as sonatrach can be a daunting task manually, so there's a need for an automated and time-saving way to go about it which is presented by the use of machine learning.

In this thesis, we focused on the behavioral analysis of log data for anomaly detection and clustering in the domain of cybersecurity. We evaluated and compared various machine learning algorithms, including K-means, DBSCAN, GMM, and Isolation Forest, to identify patterns, anomalies, and potential threats in the unlabeled sonatrach log dataset. The evaluation results revealed that K-means performed poorly in both anomaly detection and clustering analysis, mainly due to its simplistic approach of dividing the log file without considering subtle irregularities. Therefore, it was excluded from our actual detection process. On the other hand, DBSCAN successfully identified clusters, but one of them was significantly smaller, suggesting that it may have missed certain irregularities in the data. GMM demonstrated a higher sensitivity in detecting anomalies compared to DBSCAN, indicating its potential to capture a wider range of irregularities. Isolation Forest, while sensitive in capturing anomalies, did not match the performance level of GMM. Notably, anomalies detected by Isolation Forest did not exhibit recognizable patterns or distributions. Moreover, we developed an application that allows the network engineer to input log files and apply the selected algorithms to detect abnormal behavior. This application facilitates the analysis of abnormal logs, enabling a deeper understanding of the reasons behind such behavior.Through our evaluation, we have found that DBSCAN, GMM, and Isolation Forest offer varying levels of performance and sensitivity. For further examination and use of deep learning models and techniques ,sonatrach would need to do a labeling process for the logs to have precision and more applicability to the work.

Bibliography

. [1] M. Mohssen, Machine Learning Algorithms and Applications.
[2] R. Susmita, “A Quick Review of Machine Learning Algorithms”

[3] “decision tree.” https://www.saedsayad.com/decision_tree.html

[4] “naive bayes.” https://kdagiit.medium.com/naive-bayes-algorithm-4b8b990c7319

[5] B. Mahesh,“Machine Learning Algorithms - A Review”

[6] “support vector machine.” https://www.sciencedirect.com/topics/computer-science/support-vector-machine
[7] “unsupervised learning.” https://www.tibco.com/reference-center/what-is-unsupervised-learning
[8] Y. Zhang, New Advances in Machine Learning.

[9] “reinforcement learning.” https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292
[10] Zhuangbin et al, “Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection”
[11] “log collection.” https://community.netwitness.com/t5/netwitness-platform-online/log-collection-architecture/ta-p/669276
[12] “Tools and Benchmarks for Automated Log Parsing”, doi: https://doi.org/10.48550/arXiv.1811.035

[13] “log parsing.” https://www.researchgate.net/figure/A-simple-example-of-log-parsing-Log-parsing-converts-unstructured-log-messages-into_fig1_338606640
[14] “anomaly detection structure.” https://www.mdpi.com/2076-3417/13/8/4930

[15] “Feature Engineering: Scaling, Normalization, and Standardization.” https://www.analyticsvidhya.
scaling-machine-learning-normalization-standardization/
[16] “Text Normalization for Natural Language Processing (NLP).” https://towardsdatascience.com/text
normalization-for-natural-language-processing-nlp-70a314bfa646
[17] “Understanding TF-IDF: A Traditional Approach to Feature Extraction in NLP.” https://towardsdatascience.com/understanding-tf-idf-a-traditional-approach-to-feature-extraction-in-nlp-a5bfbe04723f
[18] G. Bonaccorso, Machine Learning Algorithms.

[19] “Anomaly Detection Using Isolation Forest in Python.” https://blog.paperspace.com/anomaly-detection-isolation-forest/

[20] "python website." https://www.python.org/

[21] "Scikit-learn." https://scikit-learn.org/

[22] "NumPy." https://numpy.org/

[23] "Pandas." https://pandas.pydata.org/

[24] "Matplotlib." https://matplotlib.org/

[25] "Seaborn." https://seaborn.pydata.org/

[26] "Pyqt5." https://doc.qt.io/