Doctoral Thesis

Towards a more sustainable anomaly detection: new methods and practical applications

Jorge Meira

2023

UNIVERSIDADE DA CORUÑA

Towards a more sustainable anomaly detection: new methods and practical applications

---

Jorge Meira

Doctoral Thesis

January 2023

PhD Advisors:
Amparo Alonso Betanzos, Goreti Marreiros, Verónica Bolón Canedo

PhD Program in Computer Science

UNIVERSIDADE DA CORUÑA

Amparo Alonso Betanzos
Catedrática de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

Goreti Marreiros
Profesora Coordinadora
Departamento de Ingeniería Informática
Instituto Superior de Ingeniería de Oporto

Verónica Bolón Canedo
Profesora Titular de Universidad
Departamento de Ciencias de la Computación y Tecnologías de la Información
Universidade da Coruña

CERTIFICAN

Que la memoria titulada "*Towards a more sustainable anomaly detection: new methods and practical applications* " ha sido realizada por D. Jorge Meira bajo nuestra dirección en el Departamento de Ciencias de la Computación y Tecnologías de la Información de la Universidade da Coruña, y concluye la Tesis Doctoral que presenta para optar al grado de Doctor en Ingeniería Informática con la Mención de Doctor Internacional.

En A Coruña, a 6 de enero de 2023

Fdo.: Amparo Alonso Betanzos
Directora de la Tesis Doctoral

Fdo.: Goreti Marreiros
Directora de la Tesis Doctoral

Fdo.: Verónica Bolón Canedo
Directora de la Tesis Doctoral

Fdo.: Jorge Meira
Autor de la Tesis Doctoral

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Goreti, Amparo, and Verónica, for their unwavering guidance, professionalism, and encouragement throughout my doctoral journey. Their invaluable expertise, support, and mentorship have been instrumental in shaping my research and academic progress.

I am also grateful to my colleagues and friends at GECAD, particularly to Diogo, Luís, João and Alda, for their constant support, assistance, and good company. Their friendship and camaraderie have made my time in the lab an unforgettable experience.

I would also like to extend my thanks to my colleagues and friends at LIDIA, especially Laura, Isaac, Carlos, and Eva, for the welcoming and collaborative environment they have created during my time in Coruña. Their generosity, knowledge, and kindness have been a source of inspiration for me.

I am also deeply grateful to my family, especially my parents, and my brothers, for their unwavering love and support throughout my academic journey. Their encouragement and belief in me have been a constant source of motivation.

Lastly, I would like to express my deepest love and gratitude to my wife, for her constant love, support, and encouragement. Her unwavering belief in me has been my greatest source of strength and inspiration during the most challenging times.

I would also like to express my thanks to any other people, institutions, or research groups that helped me during the research process, like LIAAD-Inesc-Tec in Porto, School of Engineering in Manchester, and INL in Braga, professors or collaborators, for their contributions and support.

This journey would not have been possible without the support and contributions

of all of the people I have acknowledged here. I am deeply grateful for their help.

Jorge Meira

Success is not final;
Failure is not fatal:
It is the courage to continue that counts.

Winston Churchill

Resumo

A detección de anomalías é un problema crítico en moitos campos, con aplicacións que van desde a detección de intrusións ata o diagnóstico de fallos e o mantemento predictivo. Os métodos non supervisados gañaron unha gran popularidade debido á súa capacidade para aprender de datos sen requirir exemplos etiquetados. Esta tese doutoral presenta unha visión xeral completa dos métodos de detección de anomalías, cun enfoque específico en técnicas non supervisadas e as súas aplicacións en varios dominios.

A tese tamén enfatiza a sustentabilidade ao presentar métodos que están deseñados para ser escalables, eficientes e capaces de manexar grandes e complexos conxuntos de datos. Os mecanismos de afinación automática dos hiperparámetros, combinados coas propiedades distribuídas de algúns dos métodos, permiten un procesamento eficiente e minimizan a necesidade de afinación manual, que pode ser consumidora de tempo e recursos. Isto resulta nun enfoque máis sustentable e eficiente para a detección de anomalías, reducindo o risco de sobrecarga de sistemas e minimizando a pegada de carbono do procesamento implicado.

Estes enfoques aplícanse a varios conxuntos de datos e dominios, incluíndo un conxunto de datos de detección de intrusións de IoT, un fluxo de datos de sistema ferroviario e as preferencias turísticas baseadas no conxunto de datos de reseñas de TripAdvisor. O rendemento dos métodos avalíase utilizando unha variedade de métricas, como a precisión de clasificación, precisión, recall, curva ROC, tempo de procesamento e tests estatísticos como o test post hoc Nemmenyi, amosando resultados de vangarda.

A investigación presentada nesta tese fai unha contribución significativa á detección de anomalías ao introducir novos métodos máis eficientes para lidiar con

conxuntos de datos grandes e complexos. Ademais, os métodos son escalables e sostibles, o que son factores importantes para a súa implementación en aplicacións do mundo real. En xeral, o traballo nesta tese proporciona unha visión detallada e actualizada dos métodos de detección de anomalías, co enfoque nas técnicas non supervisadas e as súas aplicacións prácticas, especialmente coas novas tendencias cara unha intelixencia artificial máis verde.

Resumen

La detección de anomalías es un problema crítico en muchos campos, con aplicaciones que van desde la detección de intrusiones hasta el diagnóstico de fallos y el mantenimiento predictivo. Los métodos no supervisados han ganado una gran popularidad debido a su capacidad para aprender de los datos sin requerir ejemplos etiquetados. Esta tesis doctoral presenta una visión general completa de los métodos de detección de anomalías, con un enfoque particular en las técnicas no supervisadas y sus aplicaciones en una amplia variedad de dominios.

Además, la tesis hace énfasis en la sostenibilidad al presentar métodos que están diseñados para ser escalables, eficientes y capaces de manejar grandes y complejos conjuntos de datos. Los mecanismos de ajuste automático de hiperparámetros, combinados con las propiedades distribuidas de algunos de los métodos, permiten un procesamiento eficiente y minimizan la necesidad de ajuste manual, que puede ser tardado y requerir recursos intensivos. Esto resulta en un enfoque más sostenible y eficiente para la detección de anomalías, reduciendo el riesgo de sobrecarga de los sistemas y minimizando la huella de carbono del procesamiento involucrado.

Estos enfoques se aplican a varios conjuntos de datos y dominios, incluyendo un conjunto de datos de detección de intrusiones de IoT, un flujo de datos de sistema ferroviario y las preferencias turísticas basadas en el conjunto de datos de reseñas de TripAdvisor. El rendimiento de los métodos se evalúa utilizando una variedad de métricas, como la precisión de clasificación, la precisión, el recall, la curva ROC, el tiempo de procesamiento y los tests estadísticos como el test post hoc Nemmenyi, mostrando resultados de vanguardia.

La investigación presentada en esta tesis hace una contribución significativa al campo de la detección de anomalías al introducir nuevos métodos más eficientes

para tratar con conjuntos de datos grandes y complejos. Además, los métodos son escalables y sostenibles, lo cual son factores importantes para su implementación en aplicaciones del mundo real. En general, el trabajo en esta tesis proporciona una visión detallada y actualizada de los métodos de detección de anomalías, con un enfoque en técnicas no supervisadas y sus aplicaciones prácticas, especialmente con las nuevas tendencias hacia una inteligencia artificial más verde.

Abstract

Anomaly detection is a critical problem in many fields, with applications ranging from intrusion detection to fault diagnosis and predictive maintenance. Unsupervised methods have gained widespread popularity due to their ability to learn from data without requiring labeled examples. This doctoral thesis presents a comprehensive overview of anomaly detection methods, with a particular focus on unsupervised techniques, and their applications in a wide variety of domains.

The thesis also emphasizes sustainability by presenting methods that are designed to be scalable, efficient, and able to handle large and complex datasets. The automatic hyperparameter tuning mechanisms, combined with the distributed properties of some of the methods, enable efficient processing and minimize the need for manual tuning, which can be time-consuming and resource-intensive. This results in a more sustainable and efficient approach to anomaly detection, reducing the risk of overloading systems and minimizing the carbon footprint of the processing involved.

These approaches are applied to various datasets and domains, including an IoT intrusion detection dataset, a railway system data stream, and tourist preferences based on the TripAdvisor reviews dataset. The performance of the methods is evaluated using a range of metrics, such as classification accuracy, precision, recall, area under the curve ROC, processing time, and statistical tests such as the Nemmenyi post hoc test, showing state-of-art results.

The research presented in this dissertation makes a significant contribution to the field of anomaly detection by introducing new methods that are more efficient for dealing with large and complex datasets. Moreover, the methods are scalable and sustainable, which are important factors for their deployment in real-world applications. Overall, the work in this thesis provides a detailed and up-to-date

overview of anomaly detection methods, with a focus on unsupervised techniques and their practical applications, specially with the new tendencies towards a greener AI.

Contents

List of Tables

List of Figures

List of Acronyms

ABD Average Bucket Distance.

ABS Average Bucket Size.

AD Anomaly Detection.

ADMCN Anomaly Detector for Mixed Numerical and Categorical Inputs.

APU Air Production Unit.

AUC Area under the ROC Curve.

AutoML Automated Machine Learning.

Bary Barycentric Coordinates.

BC Bucket Count.

BiLSTM Bidirectional Long-Short-Term Memory.

CB Content-Based Filtering.

CD Critical Difference.

CESGA Centre of Supercomputing of Galicia.

CF Collaborative Filtering.

CFA Cross-feature Analysis.

DDoS Distributed Denial of Service.

DF Demographic Filtering.

DOC-SVM Distributed One-Class Support Vector Machine.

DoS Denial of Service.

EADMNC Explainable Anomaly Detection on Mixed Numerical and Categorical spaces.

EF Equal Frequency.

FPR False Positive Rate.

HNS Hide and Seek.

HS-Trees Half Space Trees.

IDS Intrusion Detection Systems.

IoT Internet of Things.

KDD Knowledge Discovery and Data Mining.

LOCI Local Outlier Correlation Integral.

LOF Local Outlier Factor.

Loop Local Outlier Probability.

LSH Locality Sensitive Hashing.

ML Machine Learning.

NLP Natural Language Processing.

OC-SVM One-Class Support Vector Machine.

OCKNN One-Class K Nearest Neighbour.

PA-I Passive-Aggressive Kernel.

PDBS Piecewise Density-Biased Sampling.

PdM Predictive Maintenance.

POI Points of Interest.

RDD Resilient Distributed Datasets.

ROC Receiver Operating Characteristic.

RS Recommender System.

SCH Scaled Convex Hull.

SOM Self-Organizing Map.

SotA state-of-the-art.

SVD Singular Value Decomposition.

SVM Support Vector Machine.

TF-IDF Term Frequency-Inverse Document Frequency.

TPR True Positive Rate.

UNIDS Unsupervised Network Intrusion Detection System.

Chapter 1

Introduction

It is possible to find several definitions of anomaly detection in the literature, such as the one by Chandola et al. [43], where AD is defined as the problem of finding patterns in data that do not conform to expected behaviour. Foorthuis [74] describes anomalies has occurrences in a dataset that are in some way unusual and do not fit the general patterns. Anomalies can encompass a broad range of rare and unique occurrences that can involve both static entities and time-based events. They can be single instances or occur in groups, and can be either desired or undesired observations. The definition of anomalies covers a diverse array of phenomena.

The field of AD has been approached from diverse research areas, and different models have been proposed in many application domains in the last few years. This field has received considerable attention from the ML and data mining communities, as the ability to identify anomalous patterns in data is an important capacity to solve a vast variety of problems in many research fields. Additionally, with the increasing emphasis on sustainability and the need to reduce waste and minimize environmental impact, it is important to ensure that the methods used for anomaly detection are also sustainable. This means considering not only the accuracy and efficiency of the methods, but also the resources they consume and their long-term impact on the environment.

1.1. Types of Anomalous Patterns

Understanding the types of anomalies is critical for obtaining the most value from the insights generated. Those insights are responsible for avoiding major problems not only in the performance of processing and confirmation of results but also in the conclusions obtained by interpreting and analyzing data from a specific context. These problems can lead to incorrect strategies or decisions that can cause immediate economic damage, such as production faults or system defects. Anomalies can be classified in distinct ways depending on the domain or type of data. For instance, in [43], the authors classify anomalies into three types (Figure 1.1):

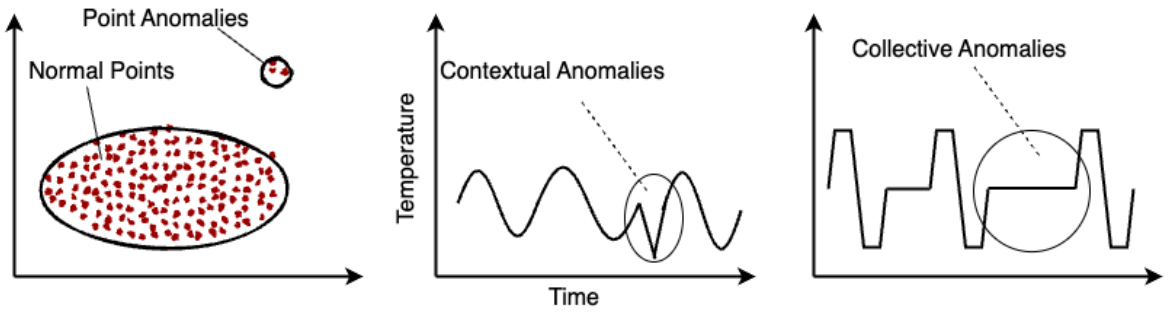


Figure 1.1: Types of anomalies Examples based on [43].

- **Point or Global anomalies:** It is defined as a Point or Global anomaly when an observation or group of instances deviates from the rest of the data. It is represented by a deviation or irregularity occurring randomly and not associated with normal data behaviour. This anomaly is the main focus of broad research work in the AD field [116]. An example in the credit fraud transaction field might be a high amount spent on an individual's credit card transaction compared to its normal range of transaction values.

- **Contextual Anomalies:** Also known as Conditional Anomaly, it occurs when an observation or group of instances deviates from other observations that exist in the same context. These data points can be abnormal in a particular context but may not be anomalous in another. This type of anomaly is commonly used in time-series data [22]; for instance, a specific tempera-

ture might be normal during the winter at a particular place, while the same temperature during the summer at the same location would be anomalous.

- **Collective Anomalies:** They occur when a subset of data points deviates from the whole data pattern. The individual data instances in a collective anomaly may not be anomalies by themselves. However, their occurrence together as a collection is considered anomalous. For instance, a consecutive 40-day period of cold temperatures could be viewed as a collective anomaly. These temperatures are unusual as they occur together and are likely caused by the same underlying weather event.

Another way to classify the types of anomalies is presented in [74], which proposed a framework for the typology of anomalies divided into five fundamental data-oriented dimensions: data type, the cardinality of relationship, anomaly level, data structure, and data distribution. The first three dimensions represent a classificatory principle that describes a key characteristic of the nature of data. Together, these dimensions differentiate between nine basic anomaly types shown in Table 1.1.

The first dimension, **types of data**, describes the behaviour of occurrences. The attributes or features are responsible for the deviant character of a given anomaly type. Those attributes can be: *Quantitative*, numerical values that capture the anomalous behaviour; *Qualitative*, categorical values that capture the anomalous behaviour; *Mixed*, the variables that capture the anomalous behaviour can be *Quantitative* or *Qualitative*.

The second dimension, **cardinality of relationship**, retracts the relation between attributes when describing anomalous behaviour. It can be classified as: *Univariate* when each attribute is independent and no relation between attributes exists to describe anomalous patterns; in *Multivariate*, the opposite occurs as there is a relationship between attributes that can describe the deviant behaviour of the anomaly.

The third dimension, **anomaly level**, distinguishes between the types of anomalies mentioned above, point anomalies (atomic, individual or low-level cases) versus collective anomalies (aggregate, or groups).

The fourth dimension, **data structure**, is used to distinguish between typolo-

Table 1.1: Typology of anomalies from [74]

		Types of Data				
		Quantitative Attributes	**Qualitative Attributes**	**Mixed Attributes**		
Cardinality of Relationship	**Univariate**	Type I Uncommon number anomaly	Type II Uncommon class anomaly	Type III Simple Mixed data anomaly	**Atomic**	**Anomaly Level**
	Multivariate	Type IV Muldimensional numerical anomaly	Type V Muldimensional categorical anomaly	Type VI Muldimensional mixed data anomaly		
		Type VII Aggregate numerical anomaly	Type VIII Aggregate categorical anomaly	Type IX Aggregate mixed data anomaly	**Aggregate**	

gies. A given cell in Table 1.1 can contain several anomaly subtypes, which have characteristics that can be traced back to the specific data formats that host them (i.e. graphs and time series).

The last and fifth dimension, **data distribution**, refers to the collection of feature values and their dispersion throughout the data space [98, 165]. The distribution of the dataset is an important factor to consider when detecting anomalies. The distribution strongly depends on the mentioned classificatory factors but focuses on density and other dispersion-related aspects of the set.

Anomalies or outliers can also be characterized in different types for specific domains. The work in [22] categorizes outliers/anomalies in three different types in the time series domain. The authors defined the first two outlier types based on the taxonomy from [43]:

- *Point outliers*: An observation that shows an unusual behaviour in a specific time instant when compared to other observations in the time series (global observations) or to its neighbour observations (local outlier). They can be univariate or multivariate regarding the impact provided in time-dependent variables.
- *Subsequence outliers*: Refers to the successive unusual behaviour of collective observations in time. Each observation individually is not necessarily an outlier but analyzing the behaviour of a consecutive set of observations makes it anomalous. This definition is very similar to the definition from [43] known as *collective anomalies*.
- *Outlier time series*: Occurs when entire time series can also be outliers. However, they can only be identified when the input data are a multivariate time series.

1.2. Applications

A wide variety of applications in the AD field have been proposed and presented in the literature over the years. It still remains a hot topic in the Artificial Intelligence domain due to its great contributions to a high diversity of applications [150, 66, 190]. Some of the AD applications fields commonly developed and studied are Health Care, Predictive Maintenance, Intrusion detection, Text and speech AD, and Fraud Detection. This section describes recent works presented in the literature for the fields addressed in this thesis and explains the context of anomalies respectively.

1.2.1. Predictive Maintenance

PdM, framed as an important issue in Industry 4.0, is able to anticipate problems or emergencies before they happen, bringing huge advantages to industries that allow them to optimize their results and obtain greater efficiency and profitability of the equipment. In PdM, an anomaly represents a failure that must be prevented. AD techniques have been extensively applied in this domain to detect such failures.

PdM can be applicable to all sectors where machines produce significant amounts of data and require maintenance or fine-tuning of their parameters. PdM applications are already gaining traction in some industries such as:

- **Airlines:** The aviation industry is grasping for opportunities to reduce costs. Security risk management has shifted from post-accident investigations and analyses to pre-accident warnings in an attempt to reduce flight risks by identifying untracked flight events and effectively preventing risks before they occur [160]. Monitoring sensor data from planes together with AD methods allow to increase passenger safety [9, 92, 18].
- **Transportation:** Although airlines lead the group in terms of the complexity of their equipment, other means of transportation, such as trains, also involve complex machinery that can benefit from predictive maintenance.
- **Ports:** Exposed to adverse conditions, the conditions of port equipment deteriorate rapidly. For instance, deviations in port container handling can be detected by AD techniques [164].
- **Automotive:** Automotive companies operate some of the largest robot parks in the world. In automotive industries, pricing anomalies may occur for components of various products, despite their similar physical features, which raises the total production cost of the company. AD methods have an important role in order to reduce production costs [84].
- **High-tech manufacturing:** Operating complex equipment at optimal parameters is the main challenge for improving the efficiency of high-tech manufacturers, such as semiconductor manufacturers. In high full automation manufacturing, unexpected equipment breakdown results in throughput loss. In order to capture the failure or deviation as early as possible, the time-to-failure or remaining useful life of each equipment should be predicted. The work from Hsu et al. [90] addressed the predictive maintenance and AD in high-tech manufacturing by applying deep learning.
- **Oil and gas:** Despite the increase in green energy, oil and gas is still one of the largest industries. Both extraction and refining involve expensive equipment that can cause risks to health and the environment in the event of failure,

as for example, the Deepwater Horizon oil spill in 2010 [131]. The stakes are high to prevent such disasters with better analysis and maintenance.

It is also addressed in Chapter 5 a real case scenario where a data-driven predictive maintenance framework is proposed for the air production unit (APU) system of a train of *Metro do Porto* [136].

1.2.2. Intrusion Detection

Nowadays, there are more and more types of computer attacks that are performed in large numbers where organizations, individuals, society, and even nations are affected. Several methods have been proposed to secure the host or network against malicious behaviour, such as IDS [105]. These systems work as a layer of protection by detecting intrusion events. Usually, two main approaches are used for IDS namely, signature-based and anomaly-based. The first type uses rules in the detection process. However, in a network environment, a large distributed network would require a large number of rules for an IDS, which could be costly and time-consuming. Additionally, if the rules are not sufficiently described, attackers might be able to access the network [63].

To overcome the mentioned gap, anomaly-based systems have been proposed based on ML methods to improve the AD performance significantly with reasonable computational resources [63].

The authors in [207] presented an anomaly-based approach to detect network attacks from flow-based features. They employed Autoencoder and Variational Autoencoder together with OC-SVM as anomaly detectors trained in a semi-supervised learning manner.

Another example is the work from Al-Turaiki et al. [6] which proposed two models based on deep learning, more specific convolutional neural networks, to address the binary and multiclass classification of network attacks. In addition, the authors applied a hybrid two-step preprocessing approach to extract meaningful features.

A study of adversarial attacks against network IDS is proposed by Aiken and Scott-Hayward [3]. The authors investigate the viability of adversarial attacks against classifiers. They implemented an anomaly-based IDS, Neptune, as a tar-

get platform that uses several ML classifiers and traffic flow features. With the development of an adversarial test tool, the authors showed that with the perturbation of a few features, the detection accuracy of a specific SYN flood DDoS attack by Neptune decreases from 100% to 0% across a number of classifiers.

Chapters 2 and 4 of this thesis present a comparative analysis of SotA unsupervised methods to deal with intrusion detection. Chapter 2 shows a comparative evaluation of network intrusion detection using two benchmark public datasets. In chapter 4 we added two novel methods in our study to detect attacks on IoT devices in a network environment. Both new methods can deal with large datasets: one, recently proposed and published by members of LIDIA lab [24], provides explainability to the obtained results, while the other (also a novel proposal of this doctoral work) provides automatic hyperparameter tuning.

1.2.3. Text and speech anomaly detection

The application of AD in text mining allows the detection of novel topics, new stories, or events in a collection of articles, documents, or web social media platforms. In this domain, anomalies can be considered as new interesting events or anomalous topics [43] (e.g. fake news, log events).

Examples of applications in this field are the work from Souza et al. [55] which proposed a network-based approach using a one-class and transductive semi-supervised learning algorithm that performs classification by first identifying potential interest and non-interest documents into unlabeled data and then propagating labels to classify the remaining unlabeled documents. They applied their approach by comparing it with four One-Class classification algorithms and analysing the performance impact of each method.

In [198] it is presented an offline feature extraction approach, named LogEvent2vec. The authors take the log event as input of word2vec method to extract the relevance between log events and vectorize log events directly. Then they transform the log event vector to the log sequence vector by applying Bary and TF-IDF techniques and trained three classical supervised methods (Random Forests, Naive Bayes, and Neural Networks) to detect the anomalies.

NLP is a component of text mining that performs linguistic analysis that essentially gives a machine the ability to understand the text and spoken words. It combines linguistics-rule-based modeling of human language by applying statistical or ML techniques. It is used for several tasks namely, speech recognition, sentiment analysis, natural language generation, and named entity recognition. Several works applied anomaly detection with the help of NLP methods to solve specific tasks such as hate speech detection [142, 108] or sentiment detection in social media platforms [180, 134]. In chapter 6 we describe another contribution of this thesis, consisting in an NLP approach that combines AD methods to improve predictions about tourists' preferences using the TripAdvisor dataset.

1.3. Anomaly Detection Challenges

Although significant improvements have been achieved in AD applications there are still challenges to be achieved. A generalization of the AD challenges that are present and common for the mentioned domains is described in this section:

- **Big data**: The growth of smart devices and sensorization of industrial activities contributes to the generation of high volumes of data previously unseen. The size of collected datasets has been steadily growing, sparking interest in ML methods. The capacity of these methods to learn and perform complex tasks was restricted by the scarcity of data. Due to the increase in data availability, the complexity of the learned tasks is now bounded by the capability of the ML method to extract relevant insights. The high computational complexity of ML methods makes it impractical to process large volumes of data. Even though there are several solutions presented in the literature on distributed and parallelized algorithms to deal with large datasets, existing approaches for AD with distributed characteristics are scarce. The proposed method in Chapter 3 addresses the issue of scalability and enables efficient processing of large data, making it more sustainable in the long term. Furthermore, by comparing its performance with a recent scalable method and traditional AD techniques in the IoT Intrusion Detection field in Chapter 4, we demonstrate the sustainability of our proposed method in terms of its ability to perform

well while dealing with large data.

- **Unlabeled data**: The data collected and available from smart devices or sensors is usually unlabeled. Labelling data demands substantial effort and time as it is usually done manually by an expert in the application domain. This is where unsupervised learning methods can be useful, as they can assist in the discovery of hidden patterns and relationships in the data without the need for labels. However, one of the main challenges with unsupervised learning is the difficulty to validate the quality of the model, since there are no labels to which to compare the predictions. Throughout this thesis, our proposals are tested in several application domains: intrusion detection (chapter 2 and 4), predictive maintenance (chapter 5) and text classification (chapter 6), with the main focus being the use of unsupervised AD methods that can deal with unlabeled data.

- **Defining Anomalies**: Anomalies that arise due to malicious activity are often changing and adapting. For instance, a model is trained to recognize malicious patterns in e-mail messages but if new patterns appear that did not exist before, the model might lose performance as they are not recognized as abnormal. This type of change in data is known as concept drift. It is a change in the statistical properties of the data that a ML model is trained on. This can happen over time as the data distribution changes, or as the goals or needs of the model's users change. Normal behaviour is continuously evolving, and the notion of normal behaviour now may not be sufficiently representative of future behaviours. The challenge of AD algorithms design is studied using non-anomalous samples only. As it would not be feasible to develop a generic framework to cover all the aforementioned applications, several AD models are developed so that each one deals with a specific domain. Although the anomalies may be different depending on the domain or context, the novel AD model proposed in Chapter 3 is not only able to deal with big datasets, as mentioned above but is also capable of being generic and independent of the domain application. To demonstrate its effectiveness, it has been validated and tested in several application fields.

- **Hyperparameter Tuning**: Hyperparameter tuning can be especially challenging in the context of AD. The performance of an AD model can be highly

sensitive to the choice of hyperparameters. The definition of what constitutes an anomaly can vary depending on the application, making it difficult to evaluate the performance of a model. Also, the space of possible hyperparameters can be large and the optimization process is time-consuming, making it challenging to find the best set of hyperparameters for a given dataset. Additionally, in many cases, the domain expert who is tasked with tuning the hyperparameters of an AD model may not have a deep understanding of the underlying data or the characteristics of anomalous behaviour. It is also a challenge when evaluating the performance of an AD model, as in most occasions, there is no ground truth to which to compare the model's predictions. This can make it difficult to know whether the chosen hyperparameters are optimal and whether the model is able to accurately detect anomalous behaviour. To address this issue the proposed method in Chapter 3 mentioned above has also an automatic tuning mechanism capable of adjusting its hyperparameters independently of the domain application. The automatic hyperparameter tuning capabilities of the proposed method can help ensure that the method is being used in the most resource-efficient way possible, making it more sustainable.

- **Explainability**: Explainability is particularly important in critical fields such as healthcare, finance or network intrusion, where the consequences of making incorrect decisions based on the results of an AD algorithm can be severe. By providing users with a better understanding of how the algorithm is working and why it is detecting certain anomalies, explainability can help improve the trustworthiness and reliability of the results. However, most of the proposed methods are not transparent and lack interpretability. In chapter 4 we describe a comparative study of AD methods in the field of intrusion detection. Among them, an explainable method is used, and an evaluation with an expert user of the results obtained is presented.

In this doctoral thesis, our main goal is to address the presented challenges by proposing not only methodologies or frameworks to deal with specific AD domains but also a novel distributed AD method with auto hyperparameter tuning. All AD methods are described and evaluated in the following five chapters of the thesis. The main objective is to provide a comprehensive understanding of the different methods and their AD effectiveness. Additionally, we aim to test these methods in

different application fields to demonstrate their versatility and utility.

1.4. Thesis Outline

This thesis is divided into 6 chapters. In this section, we provide a short summary of the contents of each of the chapters:

- Chapter 1 provides an introduction to the field of AD. It defines AD and discusses its importance in a wide range of applications including intrusion detection, text and speech AD, and PdM. This chapter outlines the main challenges and provides an overview of this thesis.
- Chapter 2 explores the use of unsupervised methods for AD in the intrusion detection field. It discusses the applications of unsupervised methods in intrusion detection systems, comparing their advantages and limitations using two benchmark datasets widely used in the literature.
- Chapter 3 presents a novel AD method that is designed to be distributed, capable of dealing with large datasets, and equipped with an automatic hyperparameter tuning mechanism. The chapter describes the method in detail and compares it with other methods in terms of its characteristics and performance. It also discusses the advantages and limitations of the method, and its potential applications.
- Chapter 4 applies several traditional unsupervised methods and two novel methods (one is the new distributed model proposed in chapter 3 while the other has explainable characteristics and has been published in **??**) to an IoT intrusion detection dataset. The performance of these methods is evaluated concerning their AD performance, processing time, and the usefulness of explanation trees. The chapter presents the results of the evaluation and discusses the strengths and weaknesses of the applied methods.
- Chapter 5 presents a data-driven approach for detecting anomalies in a railway system using a data stream model. The model is designed to handle streaming data in real-time and has limited memory, a forgetting mechanism, and

incremental learning, which allows it to perform well with limited resources. The chapter discusses the potential applications of this model and compares it with other SotA data stream methods.

- Chapter 6 investigates strategies for predicting tourist preferences based on their reviews. The chapter begins by describing the data and the prediction problem, which involves using Natural Language Processing strategies to predict whether a review is positive or negative and the rating assigned by users on a scale of 1 to 5. It then applies a range of NLP supervised methods combining them with unsupervised AD techniques to improve predictions on tourist preferences. The chapter presents the results and discussion of the evaluation performance.

- Chapter 7 concludes the thesis by summarizing the main findings and contributions of this dissertation and provides some possible lines for further work.

- Appendix A includes additional or supplementary information. It describes the datasets, the AD methods, and evaluation metrics that were used in this thesis.

- Appendix B provides a detailed abstract of the thesis written in Spanish. This abstract provides a summary of the research presented in the thesis, including the main challenges and contributions.

Chapter 2

Performance evaluation of unsupervised techniques in cyber-attack anomaly detection

Computer systems play a major role in modern everyday life. Almost everything from personal calendars to financial records and e-commerce operations is done with resources to a computing device with a network connection. Important information is stored and sent in all sorts of devices, from small low-power smartwatches to huge data centers. This creates an extensive attack vector that intended individuals and/or organizations may try to outbreak. Attackers use a variety of different techniques to try to exploit safety flaws in systems. This may result in sensible data breaches, stolen user accounts or taking control over the system.

To combat these attacks, system administrators and security experts often need to use safety measures to eliminate these attacks or at least mitigate their effects. One of these safety measures are IDS. These systems perform cyber-attack detection, using a variety of techniques to discover failures and malicious activity in computer systems. IDS tend to follow one of two different approaches: (a) signature-based, or (b) anomaly-based. Signature-based detection requires prior knowledge of an attack before being able to identify it; on the other hand, techniques based on AD work by acquiring knowledge of the patterns that represent "normal" or "attack" data and then classify new data accordingly to their resemblance to those patterns.

This latter approach gives the IDS the possibility of detecting attacks, even if the attack is not currently known (a zero-day attack, that is, an attack that is unknown or unaddressed yet, and thus can be exploited to adversely affect the computer or network), because these new attacks may present more similarities to other previous attacks rather than to "normal" data.

Within anomaly-based approach IDS, different algorithms may be used. Supervised learning algorithms are suitable for problems in which a set of already existing and previously classified samples can be used as a training dataset. On the other hand, when novel vulnerabilities and attacks are involved, there are no classified examples for a supervised algorithm to learn from it. One possibility in order to deal with this problem is the use of unsupervised learning algorithms. Unsupervised learning techniques can learn what is normal for a given set of data and then are capable of finding deviations in new unclassified data, which in this scenario would indicate a possible attack that until now was unknown.

In this chapter, we will explore the use of unsupervised ML techniques for AD. In order to be able to make a comparison among them, we have selected one of the traditional fields for the use of these models, IDS for detecting anomalies in network traffic. The work presented in this chapter is published in the Journal of Ambient Intelligence and Humanized Computing [133]. This use case is very important nowadays, as datasets are increasingly large, making it impossible to apply supervised models. Overall, this chapter will provide a comprehensive overview of the role of unsupervised methods and IDS in detecting anomalies and securing networks against threats.

The motivation for this study comes from the SASSI (Decision Support System for Security in Computer System) project (ANI — P2020 17775), which objective is the development of an Intelligent Decision Support System that centralizes, structures and allows the visualization of information regarding the activity of computer networks and the individual machines in given networks, allowing the automatic detection, prediction and prevention of anomalies, cyber-attacks and possible security risks. This platform aims to support computer network administrators who are increasingly faced with critical decision-making tasks regarding security problems that cannot be detected by typical anti-malware protection systems. This work, which was published in the Journal of Ambient Intelligence and Humanized Com-

puting volume [133], focuses on cyber-attack and AD using unsupervised learning algorithms, and explores six of these algorithms: Autoencoder, One-Class Nearest Neighbor, Isolation Forest, One-Class K-Means, One-Class SCH and One-Class Support Vector Machines, over two different public datasets the NSL-KDD [185] and the ISCX datasets [175].

Our results show that the techniques used are capable of archiving high-performance results in the classification tasks tested in our case study and consequently are candidates for future implementation in an IDS.

This chapter has the following structure: Section 2.1 presents some related work on this topic, Section 2.2 describes the workflow used including the pre-processing techniques applied in our approach, Section 2.2.3 indicates all of the unsupervised algorithms tested and which hyperparameters where used in our application, Section 2.3 presents a comparative evaluation of the results, and finally, Section 2.4 draws the conclusion and ideas for future work.

2.1. Related Work

As IDS's classification problems are a frequent topic of study in the literature, many authors have proposed and studied interesting techniques to deal with the problem of unknown attacks. The task of identifying if a new instance belongs to the class of the data that has been used for training the classifier, or whether it is an outlier, is known as one-class classification. This means that the classifier only learns the data patterns of one class (target class) in the training phase. There are other names called to this field like novelty or outlier detection, and concept learning [104]. One-class algorithms were proven to be an important tool for several domains as in disease detection [78], intrusion detection [81], text/document classification [124], or PdM [174].

Fernández-Francos et al. [67] presented a novel One-Class classification algorithm purposed for targeting distributed environments called One-Class Convex Hull-Based Algorithm. Their results showed that this method was accurate in one-class classification problems and efficient in big data scenarios due to the distributed nature of the approach. Castillo et al. [39] proposed a DOC-SVM method for clas-

sification problems. They experimented with different datasets and their results demonstrated that the proposed DOC-SVM was able to achieve accurate results and with a reduction in the necessary training time when compared to other classifiers known in the literature. Chen et al. [44] introduced the autoencoder ensembles for unsupervised outlier detection. They presented the random edge sampling technique which randomly drops connections in a neural network retaining a certain level of control on the connection density between several layers, so in this way, they can create various models with different types of density. The mentioned method was used in conjunction with the adaptive data sampling approach where the authors applied the RMSprop [191] optimization method to speed up the learning process. Their method, named as RandNet, which stands for Randomized Neural Network for Outlier Detection, showed robustness in avoiding the overfitting problem, and it was competitive with respect to other neural network techniques.

In the intrusion detection field, Goldstein et al. [82] presented a comparative evaluation of unsupervised algorithms used in the context of AD. The algorithms were applied to a group of different datasets, one of each was the KDD 99, described in Section A.1.2, however, the analyses only used part of the dataset regarding HTTP traffic. It is important to note that an improved version of this dataset called NSL-KDD is presented and used in this paper.

Aleroud et al. [8] explored the detection of zero-day attacks, with an approach that combines already existing methods with linear data transformation techniques such as discriminant functions that separated the data in normal patterns from attack patterns, and AD techniques using the One Class Nearest Neighbor algorithm to identify the zero-day attacks. Their approach consisted of a system of several static components and processes. The first component was the network data repository where they used the NSL-KDD dataset. The second component represented the pre-processing methods applied in the NSL-KDD dataset, where they converted numeric features into bins. The third module, Misuse detection, consisted in identifying attacks that are relevant to a particular context and also identifying normal activities in the network to reduce the false positives alerts. This module used conditional entropy to create known attack context profiles using patterns from historical data. Finally, the last component represented the AD module which used the 1-NN algorithm to detect deviation from normal activity and also used the SVD tech-

nique to reduce the data dimensionality. They showed good performance in their approach, detecting zero-day attacks with a low false positive rate.

Casas et al. [38] presented the concept of an UNIDS, using Sub-Space Clustering and Multiple Evidence Accumulation techniques for outlier detection. Their unsupervised security system consisted in analyzing packets captured in continuous times slots of fixed length running in three consecutive steps. In the first step, it was performed the clustering analysis to detect anomalous time slots. The second step used a multi-clustering algorithm based on a combination of several techniques [154] to rank the degree of abnormality of all the identified outlying flows. The third step used a simple threshold detection technique to flag the top-ranked outlying flows as anomalies. Their evaluation of this system included its application to the KDD 99 dataset. Noto et al. [144] studied AD using an approach called FRaC, feature regression and classification. The FRaC technique built a model of normal data and the distances of its features and used the learnt model to detect when an anomaly occurred. They also compared their approach with other commonly used techniques, such as LOF, OC-SVM and CFA.

Our work intends to show and compare the behaviour of several one-class classification algorithms (some of them already mentioned in this section) and apply them in two recent intrusion datasets with the purpose of identifying if these techniques could be integrated in an IDS inside the SASSI project.

2.2. AD Methodology

In this chapter, we study the behaviour of several unsupervised algorithms based on one-class classification, in order to verify if these techniques are a viable solution to discover and detect unknown attacks. In this section, we describe the network AD methodology, as shown in Figure 2.1. We present the datasets used and the pre-processing techniques applied to them before feeding the algorithms, as well as the unsupervised techniques employed.

In our exploration, we analyzed the NSL-KDD [185] and the ISCX datasets [175] (Consult Appendix A section A.1.1 and A.1.2). These datasets contain samples from normal activity and from simulated attacks in computer systems and are commonly

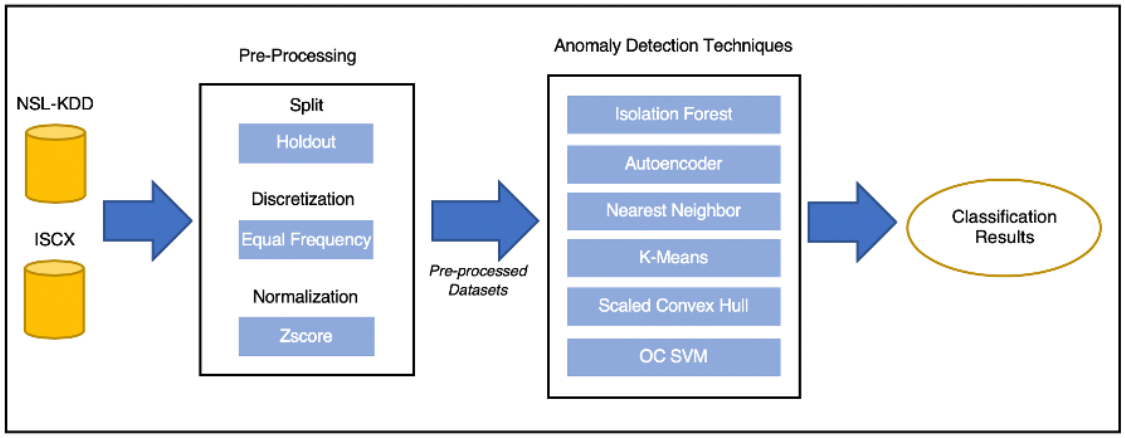


Figure 2.1: AD methodology – The datasets were splitted, normalized and discretized through pre-processing techniques before being applied in the algorithms learning and testing phase.

used in the literature. Before using the learning algorithms, we employed some pre-processing methods to prepare the data.

2.2.1. NSL-KDD dataset pre-processing

As we are testing one class classification algorithm, it was selected a portion of normal data from the training set and a portion of both normal and attack data from the test set, where the attack data contains all four attack categories and represents 10% of the test set.

Some pre-treatment techniques were applied to the dataset before performing the discretization and normalization operations, as shown in Figure 2.1. Some features were removed namely: 'Num_outbound_cmds', 'Is_hot_login', 'level_difficult', 'Land', 'Wrong_fragment', because they have redundant values in at least one of the subsets. In the case of the 'level_difficulty' feature, it represents the level of difficulty of attacks' detection by learning algorithms. This feature was removed because its information is not relevant to a real-world AD problem. Another pre-treatment operation to the data was the conversion of nominal features to numerical features since the algorithms to be employed afterwards cannot handle non-numerical data.

After performing the cleaning of the subsets, two different pre-processing techniques were applied to the data. First, the data with continuous features was discretized with the equal frequency technique. With this technique, the values of the features were divided into k bins in a way that each bin contains approximately the same number of samples. Thus, each bin has $\frac{n}{k}$ adjacent values. The value of k is a user-defined parameter, and to obtain this value we used the heuristic n where n is the number of samples. This discretization technique can provide better accuracy and fast learning in certain AD algorithms since the range of values is smaller [119].

The second pre-processing technique was data normalization, to have all the features within the same scale. This operation prevents some classification algorithms to give more importance to features with large numeric values. Once the features are all on the same scale, the classifiers assign the same weight to each attribute. The Z-Score and MinMax were the normalization techniques applied to the data. The Z-score technique transforms the input, so the mean is zero and the standard deviation is one. On the other hand, the MinMax transform the original input data to a new specific set where the values range are between 0 to 1. We tested the algorithms with each pre-processing technique and with both combined to evaluate which techniques improve the performance of the algorithms. Then we made 5 experiences with each algorithm with the best pre-processing techniques and calculate all the performance metrics mean to compare their results.

2.2.2. ISCX dataset pre-processing

For this dataset, we did the following changes before applying the pre-processing techniques shown in Figure 2.1 and described in the NSL-KDD dataset:

- All nominal features were converted to numerical – the algorithms used cannot handle non-numeric features;
- All “Payload” features were removed – These are string features, so it is not possible to train and test the algorithms with these features;
- The source and destination IP address features were removed – There is no interest in training the algorithms with these features since the IP addresses are constantly changing;

- A new feature was created to represent the time interval of an operation on the network, defined as the difference between the features "stop date time" and "start date time".

2.2.3. Unsupervised Methods

Unsupervised learning algorithms are suitable for scenarios where the objective is to perform outlier detection on a dataset. Some of these algorithms follow the basic idea of learning from a training dataset that only contains normal samples, and in the classification, the output is either "normal" if it resembles the learned set or "outlier" if it does not. These algorithms are named one-class classification methods and appear to be good candidates for the problems of discovering unknown attacks since every attack can be considered an outlier. In this work, we applied a set of 6 different one-class algorithms, namely Autoencoder, Nearest Neighbor, K-Means, Isolation Forest, Support Vector Machines, and Scaled Convex Hull, where performance was evaluated over the NSL-KDD and ISCX datasets. The description of the methods can be consulted in Appendix A, Section A.2.

For the Autoencoder, the Area Under the Curve metric was employed to compare the performance of the algorithm with different hyperparameters values. The hyperparameters values used were:

- Hidden c (50,5,50) - defines the number of hidden layers and units of the neural network, in this case, the vector c (50, 5, 50) contains 3 values and each value corresponds to the number of neurons per layer;
- Activation: Tanh - we define the activation function hyperbolic tangent;
- Epochs = 20 - Specify the number of times to iterate the dataset.

We used the h2o.anomaly function after the model finalized its training process. This function is intended to detect anomalies in a dataset. The function reconstructs the original dataset using the training model and calculates the MSE for each point in the test set. Then we created a graphic that represents the reconstruction of the mean square error as shown in Figure A.1. This graphic represents an example of a test made on a test sample of the ISCX dataset. It turns out that at a certain point

the MSE increases. This means that the model could not correctly identify these records, which could be considered an anomaly. So, we drew a threshold, in this case, equal to 0.002, where all records above this threshold are treated as anomalies.

For AD in the NSL-KDD and ICSCX datasets, and for the OCNN algorithm, the value of k was chosen to be equal to 1 since this value obtained the highest performance in the data classification.

Regarding the OC K-Means, the silhouette analysis that measures how close each point in one cluster is to points in the neighbouring clusters was used. This measure gives us information about the best parameter (number of clusters) to apply. In both the NSL-KDD and ISCX datasets the ideal number of clusters was set to 4.

In our tests for the Isolation Forest, we used the default algorithm parameter of 100 trees in both datasets, since experimentally the variation of this parameter did not show any substantial impact on the performance.

In the SCH experiments we found that the best hyperparameters for this algorithm were:

- A value of $\lambda = 1,22$ in the NSL-KDD and a $\lambda = 1,11$ in the ISCX dataset;
- Around 2000 projections;
- A center type that uses the average of the CH vertices in the projected space.

Finally, for the OC-SVM, the tests performed allowed us to obtain the best following hyperparameters in AD for the NSL-KDD and ISCX datasets:

- The radial base kernel function was used;
- The $\gamma = 0.3$ in the NSL-KDD and $\gamma = 4.2$ in the ISCX (parameter used for the radial basis kernel);
- The $v = 0.01$ in the NSL-KDD and $v = 0.005$ in the ISCX.

2.3. Performance Evaluation

All combinations of the pre-processing techniques with the unsupervised learning algorithms were tested and we present the results of the best techniques applied to each algorithm for NSL-KDD and ISCX datasets in table 2.1. To evaluate the performance of the classifiers we used several metrics described in Appendix A Section A.3.

As we can see in table 2.1, the algorithms One-class K-means and 1-Nearest Neighbor had the best performance applying the Z-Score techniques. The Isolation Forest algorithm had the best results without any kind of data transformation as it uses binary trees in the process of data recursive partitioning. In the case of the Autoencoder, SCH and v-SVM had the best performances in detecting anomalies by applying MinMax and EF techniques in the pre-processing phase.

Table 2.1: Comparative results using mean AUC ($\times$100) for each algorithm using the best combination of pre-processing techniques, in NSL-KDD and ISCX datasets

Best Pre-Processing Techniques	**OC Algorithms**	**NSL-KDD (AUC)**	**ISCX (AUC)**
No pre-processing	Isolation Forest	81.71	90.70
Zscore	K-Means	84.76	77.06
Zscore	1-Nearest Neighbor	84.85	**95.20**
Equal Frequency + MinMax	Autoencoder	83.65	80.44
Equal Frequency + MinMax	Scaled Convex Hull	**85.30**	85.95
Equal Frequency + MinMax	Support Vector Machines	83.14	91.63

Looking at the NSL-KDD results, the SCH classifier had the best performance with an AUC value of around 85. The other algorithms obtained very close results ranging between 81 and 84 AUC, where the 1-Nearest Neighbor was the second-best classifier with an AUC close to 85. Regarding the ISCX dataset, analyzing the table, we can observe that the 1-Nearest Neighbor algorithm obtained the highest AUC result, followed by v-SVM. In this dataset, the AUC results were higher compared to the NSL-KDD. One of the reasons for this is the fact that the NSL-KDD has 38 different types of attacks compared to the ISCX with only 4 different types.

To verify if there is a significant difference between the performance of the classifier in both datasets we applied the Nemenyi post-hoc statistical test (See Appendix

A Section A.4) and presented a critical difference diagram [56] as shown in Figure 2.2. As we can see all the algorithms are connected to each other (thickest horizontal line underneath the critical difference scale), meaning that they are not significantly different (at level $\alpha = 0.10$).

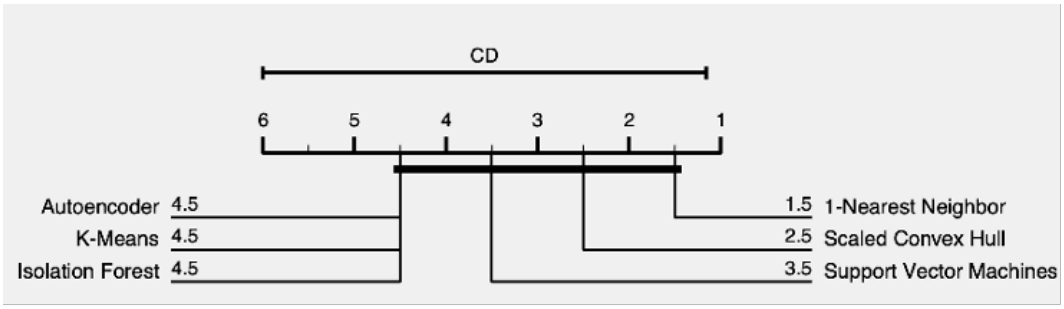


Figure 2.2: Critical difference diagram, Nemenyi post-hoc test.

The non-significant difference between algorithms can be explained because we only used two datasets to test the classifiers due to the lack of good datasets in the cybersecurity field. Even though the classifiers are not significantly different to each other, we can see that on average Nearest Neighbor, SCH and v-SVM have a high score compared to the other three algorithms.

Since the test set has unbalanced classes, we plotted the performance of the algorithms using other metrics that can measure the errors more in detail. These metrics are: Recall, Precision and F1 score.

Starting with the NSL-KDD dataset, observing Figure 2.3, looking at the F1 score metric as it represents the harmonic mean combining the two other metrics, we can see that all algorithms showed similar results. The isolation Forest and K-Means with 53% and 55% respectively and the others ranging between 60% to 66%, being SCH the algorithm with the highest F1 score. We can look also at precision and recall metrics as to have a better perception of the false positives and false negatives costs. Few false negatives represent a higher value of recall and vice-versa, and we can also say the same regarding precision with respect to the false positives. Observing the graphic in Figure 2.3, all algorithms except SCH and v-SVM had a recall value much higher than precision, so the false positives were much higher than the false negatives in these cases. In cybersecurity, it is important to have a low false negative rate since it represents the worst-case scenario, where data is predicted as a

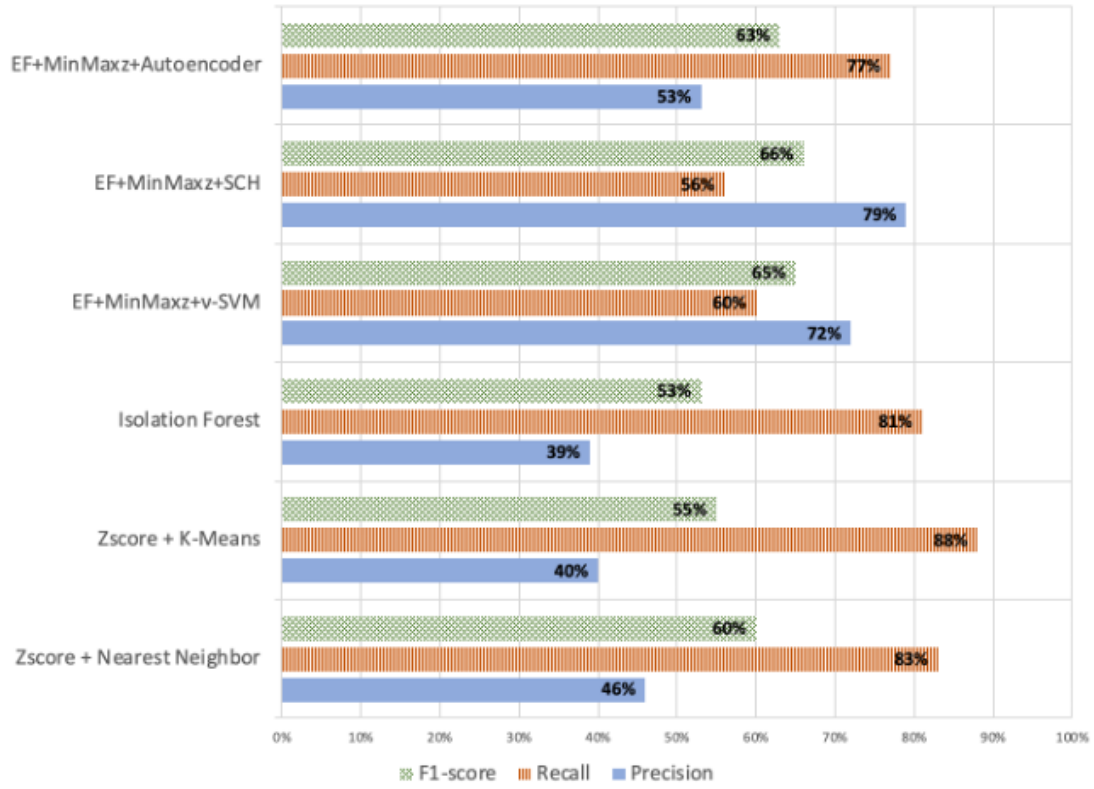


Figure 2.3: Anomaly detection results in NSL-KDD.

normal activity, while in fact, it represents malicious or abnormal activity. Regarding the SCH and v-SVM, they both had the highest F1 score compared to the other AD techniques but at the same time, they had more misclassified observations that represent false negatives than misclassified observations representing false positives.

Analyzing Figure 2.4, concerning the ISCX dataset, we observe that Nearest Neighbor, SCH and v-SVM have much better performance results than those obtained for the NSL-KDD. On the other hand, the Isolation Forest and K-means algorithms remained with approximately the same results as in NSL-KDD. Another fact that can be observed is that the algorithm SCH generates fewer false negatives and increases the false positives when trying to detect the four different types of attacks contained in the ISCX dataset.

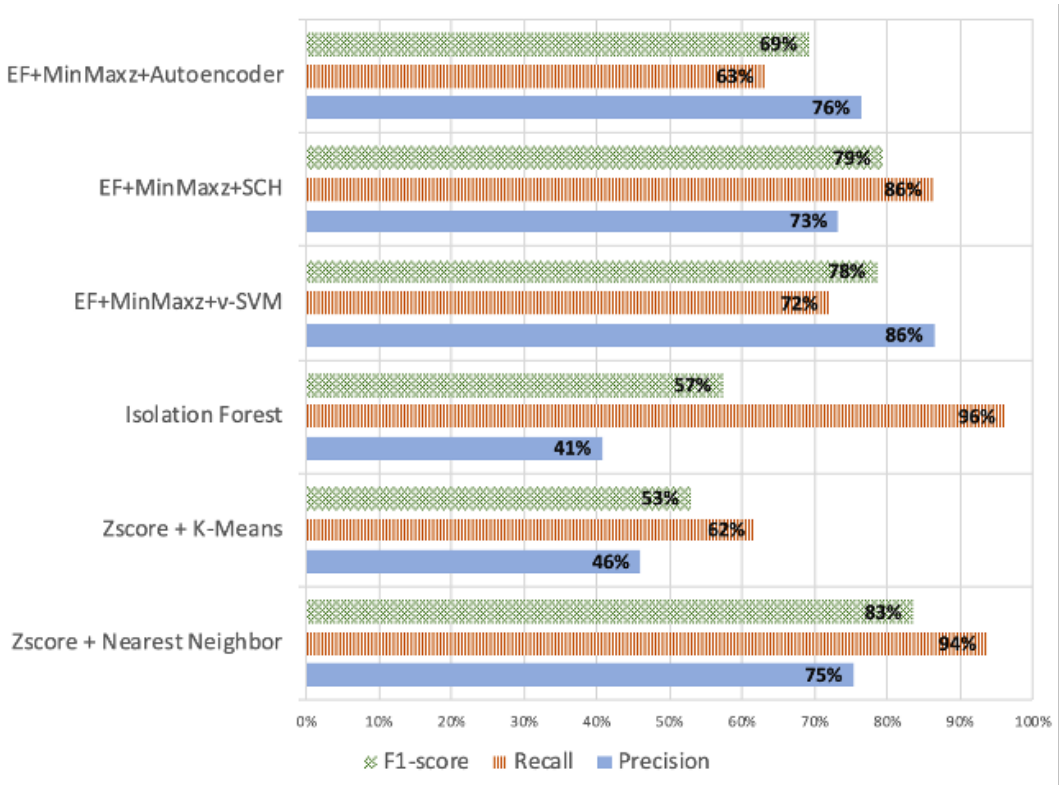


Figure 2.4: Anomaly detection results in ISCX.

2.4. Conclusion

Threats in information systems have become increasingly intelligent and they can deceive basic security solutions such as firewalls and antivirus. Anomaly-based IDSs allow monitored network traffic classification or computer system calls classification in normal activity or malicious activity. The efficiency of intrusion detection depends on the techniques used in these systems. As mentioned, the work carried out was motivated by the SASSI project. The goal was to verify if any of the unsupervised techniques presented in this paper could be implemented in an IDS to support Systems administrators in the decision-making process of anomaly and novelty detection tasks. We can conclude that all algorithms could detect most of the anomalies and also showed that they managed to separate adequately the data between classes even though they were unbalanced (to represent a more realistic environment). To choose the best method, we focus not only on the overall perfor-

mance but also on the type of errors generated. Analyzing the performance metrics, we conclude that the 1-Nearest Neighbor, SCH and v-SVM presented the highest results in both datasets but the SCH and v-SVM generated more false negatives than false positives errors in the NSL-KDD dataset. Being this type of error an undesirable scenario in cybersecurity, we suggest the implementation of the 1-Nearest Neighbor since it is capable of detecting most of the anomalies and moreover it was also one of the fastest unsupervised techniques in the computing process of AD. Although unsupervised learning methods are great to generalize, detecting unknown patterns and also handle unlabeled data problems, they have also some constraints. These methods can't be too specific about the definition of the data, leading to less accuracy (generating a high number of false positives for this specific problem), also most implementations can't deal with large datasets due to their high computational power.

Chapter 3

Fast Anomaly Detection with Locality-Sensitive Hashing and Hyperparameter Autotuning

In this chapter, we will introduce a new AD method that addresses several limitations of traditional unsupervised methods, some of which were presented in the previous chapter. Specifically, this novel method is designed to be distributed, allowing it to scale to large datasets and handle high-dimensional data efficiently. The proposed method was published in The Information Sciences Journal [135]. It also includes an automatic hyperparameter tuning mechanism, allowing for improved performance through the optimization of model parameters. By leveraging its distributed architecture and hyperparameter tuning capabilities, it is able to detect anomalies with greater accuracy and efficiency than many traditional approaches. Overall, this chapter will provide a detailed overview of the design and performance of this novel AD method, highlighting its advantages and potential applications in a variety of settings.

As mentioned in Chapter 1, anomalies are events that differ sufficiently from most of the data to indicate that they have been generated from a different process. Their minority nature is problematic, as this hinders the use of supervised ML methods, given that it is difficult to find or build data with these labelled events. As a solution, unsupervised techniques can be used that can be trained to model

normal data on unlabeled data, thereby enabling patterns that deviate from the normal to be detected.

The literature records a wide variety of AD methods that can be categorized according to a given approach [43]. Proximity-based algorithms detect anomalies by measuring their proximity to normal data points, such that elements distant from all others can be regarded as anomalies. This category includes distance-based methods, which rank elements according to their distance from neighbours, and density-based methods, which compare the density around a data point with that of local neighbours. Our proposed method is a density-based method. With density-based methods, the working assumption is that points located in low-density regions have a high probability of being anomalies: the density around a normal point is similar to the density around its neighbours, but is considerably different from the density around an anomaly [111]. Our method (described in detail in Section 3.3) corresponds to this category since its main characteristic is to randomly split data into different density groups and then analyze the density of each data point so as to infer an anomaly score. Several density-based anomaly detection methods have been described in the literature, including LOF [27], and some of its variations [96, 184], LOCI [151] and Loop [110].

AD models are becoming increasingly popular, partially due to increasingly large datasets in a Big Data context, and unlabeled data are increasingly common, mainly because sources vary greatly, e.g., connected devices such as cell phones, fleets of vehicles, or industrial machinery; anomalies, for instance, could derive from a machine on the verge of malfunctioning, or a vehicle that has experienced unusual environmental conditions. AD for large quantities of data is a difficult task, as it requires considerable computational resources. One solution is the development and application of distributed AD methods.

When dealing with large datasets, a distributed paradigm allows for parallel computation to distribute data across different nodes, with each node operating on the data in parallel. The immutable nature of distributed operations, such as in the Apache Spark framework, helps ensure consistency in computations. To exemplify, assume that we have a task such as summing all n elements of a given dataset, and the time for a single operation is t units. In the case of sequential execution by a single processor, the summation time required will be $n * t$, but if execution is by 4

processors, time would be reduced to $(n/4) * t$ plus merging overhead in time units. Scalability is becoming a must for this type of task, although at present only a few algorithms are able to cope with large datasets [77, 62, 61].

Another field that has emerged in recent years is AutoML [16]. Almost every ML method has hyperparameters, and thus a key task is the optimization of these hyperparameters so as to maximize algorithm performance. AutoML automatically sets these hyperparameters to optimize performance, thereby reducing human effort and obtaining a more rapid and simple solution.

The method that is presented in this chapter describes a novel density-based method for AD, called LSHAD, based on the LSH technique. LSHAD was developed to address the above-described problems regarding the difficulty of processing very large datasets composed of data generated daily, and the lack of unsupervised methods for AD problems capable of automatically adjusting hyperparameters. Therefore, the main contributions of this work are the following:

- Adaptation of the LSH technique to AD in large datasets.
- Autotuning of hyperparameters. ML success heavily relies on humans to select appropriate hyperparameters, a very complex and time-consuming task that becomes even more critical when it has to be carried out by ML non-experts rather than experts, which happens quite often. There is therefore a great need for AutoML methods [16].
- A distributed algorithm, since development is in the Apache Spark framework using the MapReduce approach for distributed environments

This method is rapid and effective when the objective is to process large quantities of data in search of anomalies. It achieves a similar (in some cases better) performance in AD compared to other methods. It also has the advantage over alternative methods that it is rapidly configured, as there is no need to tune hyperparameters, and can handle large datasets. By being able to process large amounts of data efficiently, this method can help reduce the computational resources needed for anomaly detection, which can reduce the environmental impact. Furthermore, the automatic hyperparameter tuning capabilities of the LSHAD method can help ensure that the method is being used in the most resource-efficient way possible, which can further

contribute to its sustainability. The scalable and efficient nature of the LSHAD method makes it a valuable tool for promoting sustainability in anomaly detection.

The rest of this chapter is organized as follows: Section 3.1 describes the LSH technique developed by Indyk and Motwani [94] and applied to our algorithm; Section 3.2 reviews SotA methods used for AD; Section 3.3 explains LSH detailed functionalities and describes 4 different types of estimators. Section 3.4 describes our LSHAD algorithm, explains the automatic hyperparameter tuning process, and describes an experiment to identify the best estimator. Section 3.5 evaluates our method and compares it to other algorithms in terms of AD and execution time. Finally, Section 3.6 summarizes our main conclusions.

3.1. Background

The basic concept underlying LSH, introduced by Indyk and Motwani [94], is to identify approximate nearest neighbors through the use of hash functions. The underlying principle that two points in the feature space that are close to each other are very likely to have the same hash function. LSH is formally defined by Indyk and Motwani [94] as follows:

Definition 1. *Given a space* $\mathbb{R}^{dim}$*, and distance thresholds* r_1, r_2*, a family* $\mathcal{H} = \left\{h : \mathbb{R}^{dim} \to U\right\}$ *is called* (r_1, r_2, P_1, P_2)*-sensitive if for any two points* $p, q \in \mathbb{R}^{dim}$ *it satisfies:*

- *if* $\|p - q\| \leqslant r_1$ *then* $P_{\mathcal{H}}[h(q) = h(p)] \geqslant P_1$,
- *if* $\|p - q\| \geqslant r_2$ *then* $P_{\mathcal{H}}[h(q) = h(p)] \leqslant P_2$.

The first condition above states that nearby objects within distance r_1 will collide in the same bucket with a high probability, whereas the second condition states that distant objects will be hashed to the same bucket with a small probability. In order for a family $\mathcal{H}$ to be useful it has to satisfy $P_1 > P_2$ and $r_1 < r_2$.

Generated from $\mathcal{H}$ is a h hash function by the concatenation of various L random projections (a user-specified parameter explained in Section 3.3), $h =< proj_1, proj_2, ..., proj_L >$. As shown in Definition 1, the method is probabilistic, so

the problem of false neighbor detection needs to be dealt with. A common practice to make the hashes more specific by increasing L. However, if hashes are very specific, many points may end up in different buckets from their neighbors. Therefore, T hashes are generated for each point. The impact of L and T on algorithm performance is studied in Section 3.4. LSH speeds up the search for neighbors in requiring much less computational effort than the brute-force approach of measuring every possible pairwise distance. LSH and variants have already been successfully applied in practical scenarios such as computer vision [115], recommender systems [49], and linguistics [1]

In implementing the LSH technique in our AD algorithm, the goal is to rapidly retrieve neighbor counts to be used as a ranking score for AD. We assume that points with few neighbors are very likely to be anomalous. An advantage of using this technique is that it rapidly processes data in high-dimensional spaces, which, when combined with distributed implementation in Apache Spark, makes our LSHAD algorithm highly scalable.

3.2. Related Work

Below existing work related to AD and outlier detection algorithms, with very similar definitions [178] is described. We first describe frequently used and recent general methods, then we focus on density-based methods, and lastly, some LSH variants.

Liu et al. [118] developed their Isolation Forest algorithm that works with binary trees. Each tree is created by partitioning instances recursively and randomly selecting a split value for a specific attribute. Tree path length is used as an anomaly score, with data points with shorter path lengths considered anomalies.

OC-SVM [171], a variant of the classical SVM algorithm, is another method that can be applied to AD problems. It relies on finding the smallest hypersphere containing all training examples after mapping by a kernel function. Different approaches to fitting an SVM model are training with data from different classes, training with data from unknown classes, and training with data from a single class. In the OC-SVM method, all the data in the training set are represented by only one

class. In AD problems the method is usually used to train data belonging to the non-anomalous class, as these data are commonly available. The algorithm separates all data points from the origin and maximizes the distance from the hypersphere to the origin, resulting in a binary function that captures regions in the input space where the data density probability is high [171].

Martínez-Rego et al. [126] proposed a modification of the One-class classification with a PA-I algorithm combining it with a Bernoulli CUSUM chart to deal with stream change problems. With this adaptation, the method is capable of accurately fitting the support of normal data in an online fashion. Thus, it can dynamically adapt to changes in data distribution.

Deep learning is still a hot topic, with numerous applications and approaches described in the literature in fields such as computer vision [25], speech recognition [152], natural language processing [85, 79], etc [89, 91]. Deep-learning methods are also widely used in AD problems [42], especially the autoencoder architecture [40]. This method is trained in order to make output features the same or very similar to input features [80]. Autoencoders are composed of two parts: the encoding layer(s) compress(es) the input into a latent-space representation, and the decoding layer(s) reconstruct(s) the output from this representation. The anomaly ranking score is computed from the reconstruction error metric, which measures the difference between input and output data.

Eiras-Franco et al. [62] recently proposed the ADMCN algorithm, which, as the name indicates, targets data with both categorical and numerical variables. The model is trained through a maximum-likelihood objective function optimized with stochastic gradient descent. It is capable of dealing with large quantities of data since implemented in Apache Spark, the algorithm lends itself well to parallel computation.

Concerning density-based methods, Breunig et al. [27] proposed the LOF algorithm, which searches for anomalous data points by measuring the local deviation of a given point from its neighbours. The same concept inspired other developments, such as LOCI [151], which aims at fast outlier detection using the local correlation integral. This improved method can identify not only outliers but also groups of outliers, providing an automatic cutoff to determine whether or not a point is an out-

lier. Its main drawback is its quadratic complexity, which makes it computationally expensive, and thus prohibitive for very large datasets.

LSH methods have recently been successfully applied to AD problems. Wang et al. [199] proposed an LSH framework for ranking points according to the likelihood that they are anomalous. The data is first split in clusters and then a ranking of points is computed by building LSH tables. Each point is next evaluated according to its rank to isolate a certain number of anomalies. This ranking mechanism is based on the number of points hashed to the same bucket on the assumption that points in buckets with few elements are likely to be anomalies. The authors reported that they could isolate the top anomalies very quickly, usually by scanning less than 3% of the dataset, and in their empirical study their method outperformed other AD methods, although the comparison was with just two other methods.

Pillutla et al. [158] presented an approach in which LSH is used to prune non-outlier data points according to their redundancy in a hash table. The algorithm then processes the data using the pruned points, which makes this approach computationally less costly. The authors developed a distributed system for their algorithm and evaluated their method in terms of AD and communication time, but did not compare their method with other algorithms.

Zhang et al. [208] proposed a density-biased sampling approach using LSH to count neighbors and obtain a scalable density estimate. They also proposed a parameter tuning rule, specific to AD for LSH. They formally investigated density-biased sampling for AD, suggesting that, given the different importance of data points according to density, this approach to sampling would have a higher impact on AD performance compared to uniform sampling, and conducting an empirical study to compare the approaches.

The works by Wang et al.[199], Pillutla et al.[158], and Zhang et al.[208] described in this section use LSH techniques for AD problems. We identified the following differences with our method:

- Although the results reported by Wang et al.[199] showed that their method is more scalable than others included in their study, they did not mention whether their method is capable of performing distributed computing (as was the case for Pillutla et al.[158]). Implementation of our method in Apache

Spark enables distributed data processing across various processor cores, and thereby enabling larger datasets to be handled than handled by competitors.

- Our method adjusts hyperparameters automatically, relieving the user of this time-consuming task and contributing to the AutoML field.
- Our experimental study (described in detail below) is much broader, as we compare our method across a wide range of datasets and with different AD methods.

Table 4.3 summarizes the different methods, considering hyperparameter autotuning and distributed computing capabilities.

Table 3.1: Characteristics of different anomaly detection algorithms

Methods	**Auto-Hyperparameter**	**Distributed**
One Class SVM	Yes[1]	No
LOF	Yes[1]	No
LOCI	No	No
Pillutla et al.[158] method	No	No
Wang et al.[199]	No	No
Zhang et al.[208]	No	No
PA-I	No	No
Autoencoder	No	Yes
IForest	No	No
ADMNC	No	Yes
LSHAD	Yes	Yes

3.3. Proposed Method

Before we describe the use of the LSH technique for AD, we explain the automatic hyperparameter tuning mechanism implemented in our method and the impact of each hyperparameter on the process of generating random projections and creating

[1]It has several hyperparameters, with only one tuned automatically

groups of neighbors. We also describe several density estimators that measure the number of neighbors for each data point.

The main idea behind our method is to obtain an estimate of the density of the different input space regions rapidly and inexpensively thanks to distributed computation using the MapReduce approach implemented in Apache Spark [205]. Our proposed method leverages the LSH technique by applying hash functions to group data points in buckets with their neighbors. The number of neighbors in each bucket is then used to compute several evaluation metrics that score and rank elements according to level of anomaly. This process is described in Algorithm 1. First, a suitable set of hyperparameters is obtained using the tuning procedure described in Section 3.4 (Line 1). Then a hasher, consisting of $L * T$ hyperplanes, is created to obtain the hashes for each element in the training dataset D. The number of elements corresponding to each hash is counted and used to compute an estimator (Line 3). Finally, the estimator values are used to establish a threshold below which a point is considered an anomaly. The threshold is selected so that the number of elements that fall below it corresponds with the anomaly ratio for the training data, which is provided by the user.

Algorithm 1: Pseudocode for LSHAD. Training phase.

Input : $D \leftarrow$ Set of training points,
$anomalyRatio \leftarrow$ Fraction of the dataset expected to be anomalous
Output: $hasher \leftarrow$ set of hyperplanes to obtain hashes,
$estPerHash \leftarrow$ dictionary associating each hash with its estimator value,
$threshold \leftarrow$ estimator value used to deem an element to be anomalous

1 $L, T, w \leftarrow tuneHyperparameters(D)$;
2 $hasher \leftarrow$ new HASHER(L, T, w);
3 $estPerHash \leftarrow hasher$.HASHANDESTIMATEPERHASH$(D)$;
4 $threshold \leftarrow$ COMPUTETHRESHOLD$(estPerHash, anomalyRatio)$;

The model, consisting of an estimator value for each hash, a set of projection hyperplanes, and a threshold value, is fitted to the training data, and assessing whether a test point p is an anomaly follows the process described in Algorithm 2. First, the hashes for p are computed (Line 1), then the estimator values corresponding to the

assigned hashes are accumulated. If the resulting value fails to reach the threshold established by the learned model, then p is an anomaly.

Once the model is trained, predictions can be made using the Algorithm 2. Checking a test point requires generating all its hash values with the hasher. An estimator is calculated using the precomputed counts in the model, which represent the properties of the training data distribution.

Algorithm 2: Pseudocode for LSHAD. Detection phase.

Input : $p \leftarrow$ Test point,
$hasher \leftarrow$ Hasher of the trained model,
$estPerHash \leftarrow$ Estimator dictionary,
$threshold \leftarrow$ Estimator threshold
Output: Boolean value indicating whether p is an anomaly

```
1 hashes ← hasher.HASH(p);
2 estimator ← 0;
3 foreach h ∈ hashes do
4 |   estimator ← estimator + estPerHash[h];
  end
```

Result: $estimator < threshold$

3.3.1. Hashing

Although the LSH techniques that we use can draw on many LSH hash families, since our implementation is based on the Euclidean distance, we selected the corresponding classical hash function, formally computed as follows:

$$proj(\boldsymbol{x}|\alpha,\beta) = \lfloor \frac{\boldsymbol{x}\cdot\alpha+\beta}{w} \rfloor \tag{3.1}$$

The projection $proj(x|\alpha,\beta) : \mathbb{R}^d \rightarrow \mathbb{Z}$ maps a d dimensional vector $\boldsymbol{x}$, representing each data point, onto the set of integers, where $\boldsymbol{\alpha}$ is a random vector drawn from a Gaussian distribution, and where β is a real number uniformly chosen from the interval $[0 : w]$. This scalar projection is then quantized into a set of hash buckets, grouping all elements that are close together in the original space in the same bucket. The user-specified hyperparameter w in Equation 3.1 represents the

resolution of the quantization. Figure 3.1 shows one such hash function, consisting of a random projection in 2 dimensions with a specific w value.

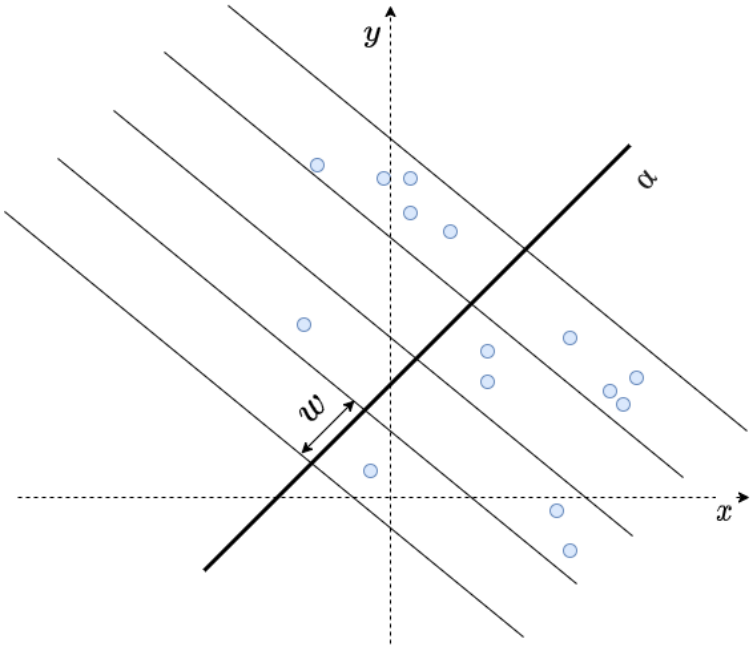


Figure 3.1: Random projection in 2 dimensions. Axis $x, y \in \mathbb{Q}$ and the blue dots are composed by random values from $\mathbb{Q}$.

A hash function, represented by such L random projections, defines the hash value (Equation 3.2):

$$H(x) =< proj_1(x|\alpha_1, \beta_1), ..., proj_L(x|\alpha_L, \beta_L) > \tag{3.2}$$

Where $proj_i(x|\alpha_i, \beta_i)$, $1 \leq i \leq L$ (from Equation 3.2) is computed by Equation 3.1. After all the hash functions are generated, observations with the same hash values are grouped together.

Using the same notation as used in Definition 1 in Section **??**, in order for a family $\mathcal{H}$ to be useful it has to satisfy the condition that the probability of P_1 is much higher than that of P_2. Hash functions $H(x)$ will, in some cases, not fulfill this condition, especially as they are generated at random. To ensure that $P_1 > P_2$ while taking into account the probabilistic properties of $H(x)$, T hash tables are created, each one indicating the hash of each data point. As a result, each point $\boldsymbol{x}$

will receive a set of hashes $\{H_1(x), H_2(x)...H_T(x)\}$. When grouping elements with the same hash, the method creates groups of elements that have a high probability of being close together. However, increasing the values of parameters L and T also increases the computational complexity of the algorithm, since more hashes need to be generated. It is therefore necessary to identify suitable values for these parameters that trade off accurate AD against as little computational effort as possible.

Figure 3.2 shows an example of a hash table $H(x)$, with data points on the left and the hash table on the right. The rows represents different hash values and the righthand column shows the collisions, which occur when data points share the same hash value.

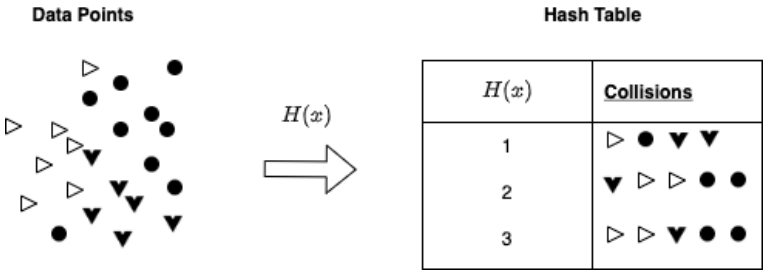


Figure 3.2: Hash table example

3.3.2. Anomaly level estimation

Once a suitable hasher has been found, the next step is to estimate the density of the regions of the input space represented by each hash. Points hashed to low estimator buckets will be deemed anomalous. We explored 4 different density estimators, as follows. Let D be the input dataset and let $b_h = \{x \in D, H(x) = h\}$ be the set of points with hash h in one of the tables t. We define the neighbors of point x as the set of elements in the dataset that share a hash with x across all T tables: $neigh(x) = \bigcup_{t=1}^{T} b_{H_t(x)}$:

- Estimator A represents the number of points in the bucket:

$$E_A(h) = |b_h| \tag{3.3}$$

- Estimator B is the average number of neighbors of the points contained in the bucket:

$$E_B(h) = \frac{\sum_{x \in b_h} neigh(x)}{|b_h|} \tag{3.4}$$

- Estimator C represents the ratio between $E_A(x)$ and $E_B(x)$:

$$E_C(h) = \frac{E_A(x)}{E_B(x)} \tag{3.5}$$

- Estimator D represents the sum of the inverse of the number of neighbors of all points in the bucket:

$$E_D(h) = \sum_{x \in b_h} \frac{1}{neigh(x)} \tag{3.6}$$

In Section 3.4 we analyze the 4 estimators to determine which one gives the best anomaly ranking score.

3.3.3. LSHAD framework

LSHAD is implemented in the Apache Spark framework, designed for fast performance using RAM for caching and MapReduce for processing data. Parallel computation is enabled by the use of RDD, an immutable partitioned collection of records with partitions that can be operated in parallel. Even though RDD are immutable, they can be transformed into other RDD using functions such as mapping, filtering, joining, groupBy, etc. The immutability ensures consistent computations since any changes in RDD are permanent; the fact that data can be safely shared across various processes and threads enhances the computation process by caching RDD. Figure 3.3 shows how LSHAD makes use of RDD to compute tasks in parallel.

First, LSHAD splits data into train and test sets, and each set is transformed into an RDD in which data is partitioned according to a user-defined number of nodes/partitions that allowing task to run in parallel. In the training phase, LSHAD adjusts its hyperparameters in 2 iterative steps performed in parallel in the multiple partitions, namely, creating hashes, and retrieving specific measurements to search

for the w size values that build optimal hash tables. Section 3.4 describes this process in detail.

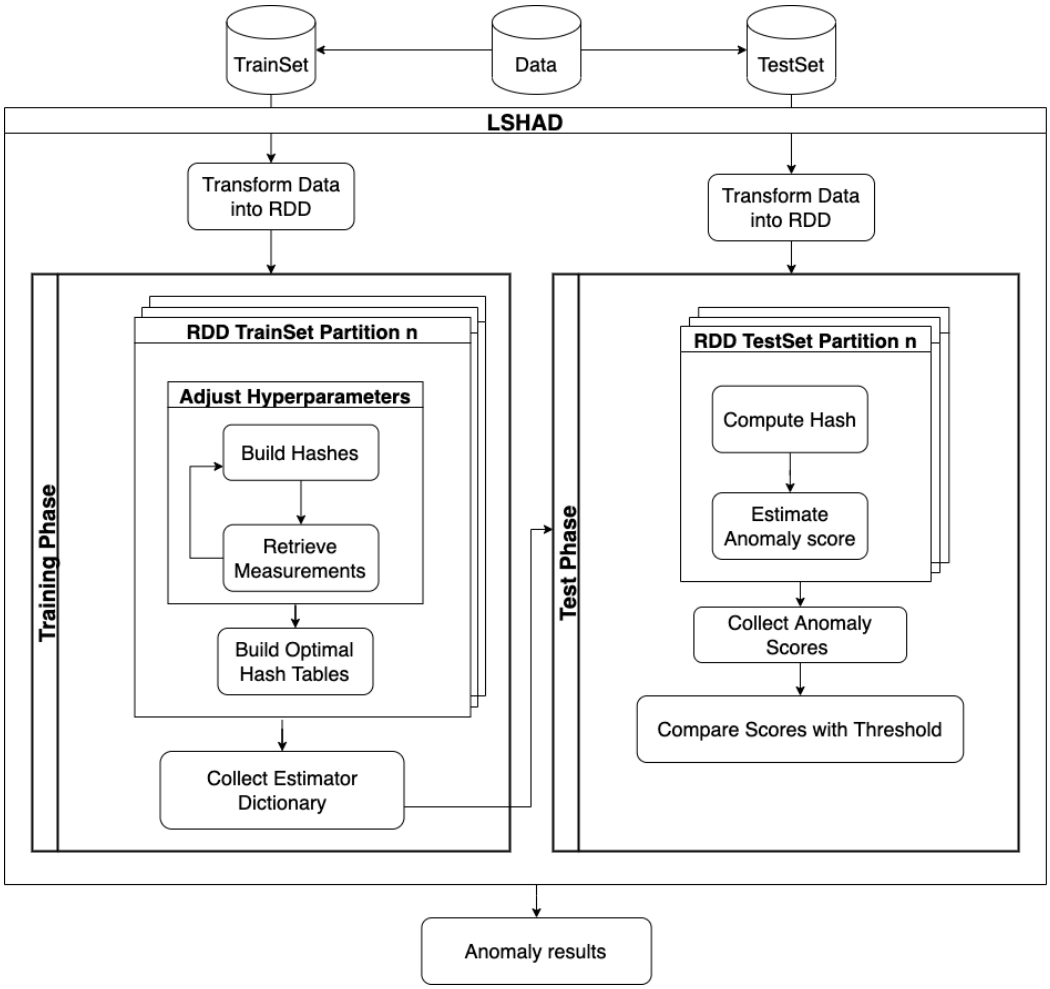


Figure 3.3: LSHAD diagram

In the testing phase, LSHAD calculates new hashes for each test point in each partition. It then accumulates the estimator values corresponding to the hashes assigned to all evaluated points. Finally, all estimator values are collected and compared to a threshold value to determine whether an observation is normal or anomalous.

3.4. Hyperparameter Tuning and Experimentation

AutoML has been a hot research topic in recent years [16], as applying traditional ML methods is time-consuming, resource-intensive, and challenging. Manual hyperparameter tuning is challenging, as besides being very computationally expensive, hyperparameter tuning has a great influence on the final algorithm results.

Given the possible difficulties faced by a non-expert user in tuning hyperparameters when only unlabeled data is available, for LSHAD, we implemented automatic hyperparameter tuning, studying the behavior of each user-specified parameter, that is, the resolution of quantization buckets w, the number of random projections L, and the number of tables T.

3.4.1. Datasets

To analyze the hyperparameters and evaluate various AD methods, we selected several datasets widely used in the literature for classification tasks but adapted for AD tasks. The datasets, presented in Table 3.2, were downloaded from the UCI ML [60], Zenodo[2] and Stratosphere Research Laboratory[3] Repository.

Regarding the UCI ML Repository datasets, we used a version of Abalone, which contains data on abalone shell characteristics that predict its age (number of rings of a cut shell). The idea is to observe whether an algorithm can identify differences in specific age ranges. Thus, for Ab. 1-8, Ab. 9-11, and Ab.11-29, classes considered anomalous are 1-8, 9-11, and 11-29, respectively, while all other classes are considered non-anomalous

Also used was a sample of 20% of the CoverType dataset, composed of cartographic variables that classify different types of forest cover. In this dataset, class 2 instances (Lodgepole Pine) were considered normal, while class 4 instances (Cottonwood/Willow) were considered anomalous.

Other datasets selected from the same repository were German Credit, Arrhyth-

[2]https://zenodo.org/
[3]https://www.stratosphereips.org/

Table 3.2: Datasets used to analyze hyperparameter tuning and anomaly detection evaluation

Synthetic datasets	**Samples**	**Features**
2 banana clusters (2BC)	1,000	2
2 circular clusters (2CC)	1,000	2
2 point clouds with variance (2PV)	1,000	2
3 anisotropic clusters (3AC)	1,000	2
3 point clouds (3PC)	1,000	2
Real datasets: small	-	-
Abalone 1-8 (Ab. 1-8)	4,177	11
Abalone 9-11 (Ab. 9-11)	4,177	11
Abalone 11-29 (Ab. 11-29)	4,177	11
Arrhythmia (Arrhyth)	420	278
German Credit (GC)	1,000	20
Heart	270	14
Pima Diabetes (Pima)	768	9
Breast Cancer (Breast)	683	10
Real datasets: medium	-	-
CoverType (CT)	56,911	12
KDDCup99 (KDD99)	44,000	41
KDDCup99 (http) (KDD99h)	64,293	40
KDDCup99 (smtp) (KDD99s)	97,23	40
IDS 2012	42,301	27
IOT-23 sample (ID dataset: 1)	44550	18
Real datasets: large	-	-
IOT-23 (ID: 1)	1,008,749	18
IOT-23 (ID: 3)	156,101	18
IOT-23 (ID: 7)	11,454,723	18
IOT-23 (ID: 9)	6,378,294	18
IOT-23 (ID: 17)	54,659,864	18
IOT-23 (ID: 33)	54,454,592	18
IOT-23 (ID: 35)	10,447,796	18
IOT-23 (ID: 36)	13,645,107	18
IOT-23 (ID: 39)	73,568,982	18
IOT-23 (ID: 43)	67,321,810	18
IOT-23 (ID: 48)	3,394,347	18
IOT-23 (ID: 49)	5,410,562	18
IOT-23 (ID: 52)	19,781,379	18
IOT-23 (ID: 60)	3,581,029	18

mia, Pima Diabetes, Breast Cancer, Heart, three versions of the KDDCup99 dataset, and finally, IDS 2012, an update of the KDDcup99 that solves some of its problems. For the German Credit dataset, representing people receiving bank loans classified as good or bad credit risks according to specific attributes, we considered bad credit risk as the anomalous class. For the Heart dataset, we considered patients with the disease to be an anomalous class. Regarding KDDCup99, a well-known intrusion detection dataset, we used the following versions: KDD99 (a sample of the full KDDCup99 with all cyberattacks), KDD99-SMTP (reduced KDDCup99 filtering only SMTP connections), and KDD99-HTTP (reduced KDDCup99 filtering only HTTP connections). For the KDDCup versions and the IDS 2012 datasets, we considered any sort of intrusion as anomalies.

From the Zenodo repository, we took a synthetic dataset of two-dimensional combinations of attributes of clusters of different shapes (see Figure 3.4).

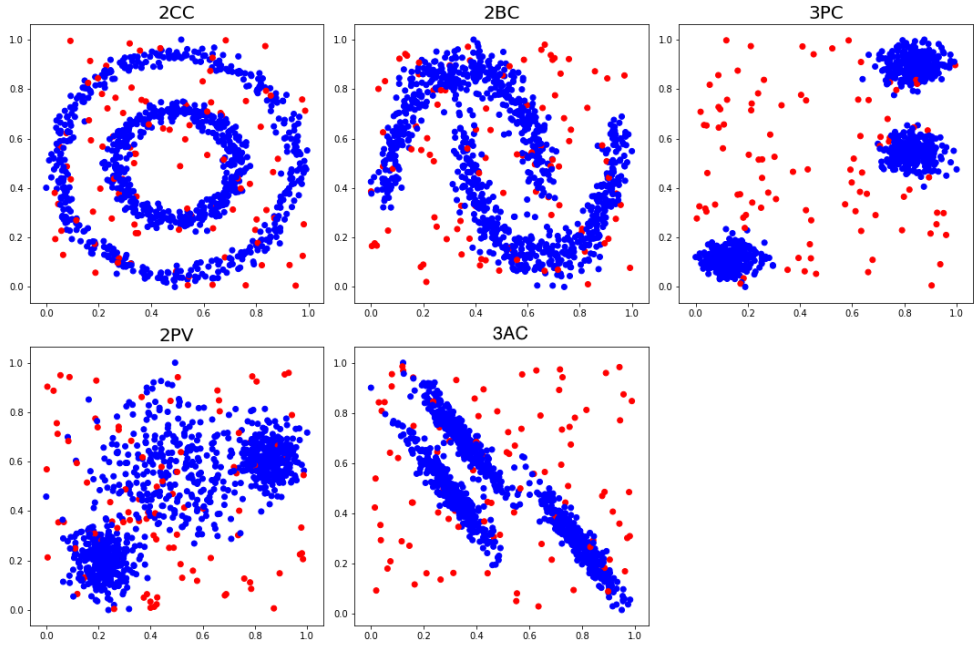


Figure 3.4: Synthetic dataset of shapes representing 2 circular clusters (2CC), 2 banana clusters (2BC), 3 point clouds (3PC), 2 point clouds with variance (2PV), and Anisotropic Clusters (3AC).

From the Stratosphere Research Laboratory repository, we took a newly released dataset called Aposemat IoT-23, containing several subsets of malicious and benign

network traffic for real IoT devices, for which we consider attacks as anomalies.

To analyze hyperparameter behaviour and test the estimators we used 8 real datasets, namely, Ab.1-8, Arrhythmia, German Credit, CoverType, all KDDCup99 versions, and IDS 2012, and to evaluate and compare the AD methods we used all datasets, as described in Section 3.5.

3.4.2. Hyperparameter analysis

To analyze the behaviour of each hyperparameter and its impact on the performance of our method, we conducted an experimental study to identify regular patterns or correlations between the performance of the LSHAD algorithm and hyperparameter values in order to build an automatic tuning mechanism.

For all the experiments we used five-fold cross-validation, computing the average for each metric used. For this particular study, the datasets were modified in order to retain 1 % of anomalies. This was done to both provide an accurate anomaly rate to the algorithm, and to ensure that normality is learned by keeping the number of anomalies low. This would not be possible in real-life as the user would have verified that the training dataset represented normality and would need to provide an estimate of the anomaly rate for the training dataset.

We first fixed a constant value for the parameter w and tested the performance of the algorithm using the metric area under the curve (AUC) for one of the estimators for different values of L and T.

Analysing the results, it was observed that by increasing the number of hash tables (T), LSHAD performance remained very similar irrespective of the number of random projections, L, to be generated. We tested values from 1 to 128 for L for each different T value; for visualization purposes, Figure 3.5 shows a plot of LSHAD AUC performance measured for 2 random projections, $L = 2$, and for different hash table values T (from 5 to 1,000). As can be observed, LSHAD performance improves as the number of hash tables increases, until stagnating at a particular AUC value. This behaviour was expected, as mentioned in Section **??**, as repeating this random process several times will increase the likelihood that 2 similar data points will collide in the same bucket. As can be seen in Figure 3.5, for most datasets the LSHAD

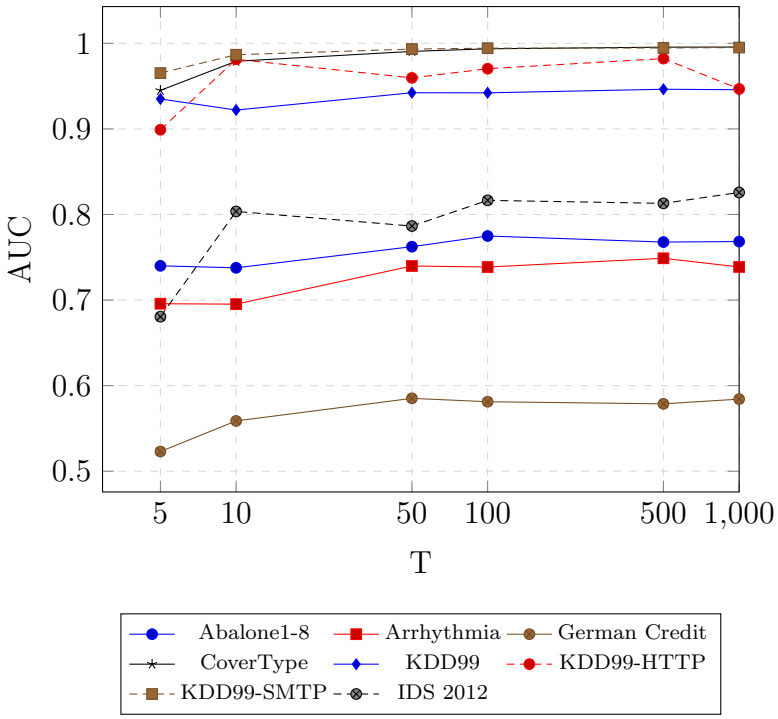


Figure 3.5: LSHAD performance (AUC) varying the hyperparameter T, the number of hash tables.

performance approached optimum at $T = 50$; for higher values, performance was maintained or slightly improved, while in some cases, the repetition resulting from a high number of hash tables deteriorated performance.

Figure 3.6 depicts the model AUC versus different values of L (for a fixed T=50), showing that different L values have a small impact on LSHAD performance for the CoverType dataset[4] and all the KDD99 dataset versions. Performance deteriorated greatly for Abalone 1-8 with more than 8 random projections and for Arrhythmia, with more than 32 random projections, and improved greatly for IDS 2012 with more than 16 random projections. No conclusions could be drawn regarding the effect of L in the German Credit dataset; therefore, we fixed the parameter $L = 4$, as an acceptable value to trade off performance against computational cost. This is because shorter hashes require less memory in saving the model and so can be

[4]Performance is similar to that for KDD99-SMTP but this is not visible in Figure 3.6 since the corresponding line is behind the KDD99-SMTP line.

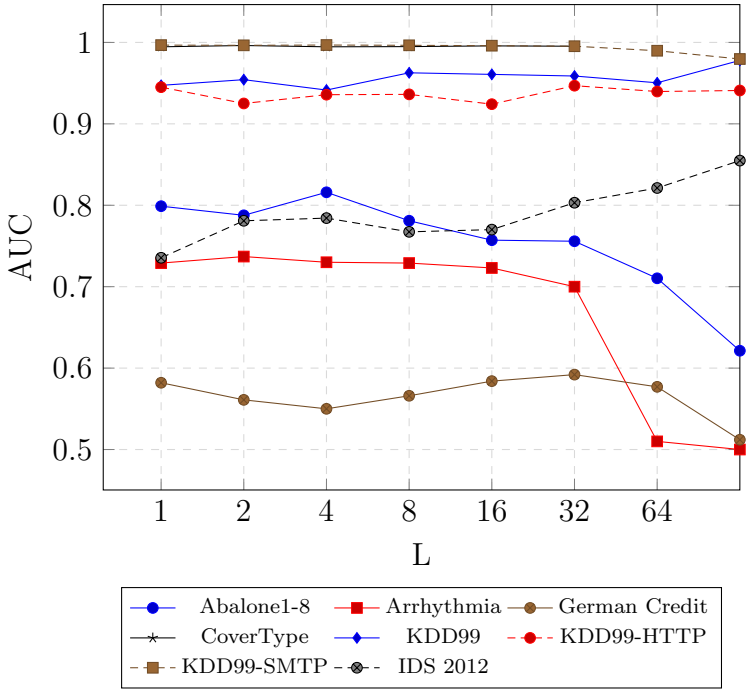


Figure 3.6: LSHAD performance changing the hyperparameter L, the number of random projections

processed faster.

In the next experimental step, for the same test approach, we fixed the values of both the T and L parameters. We set $T = 50$, because, as observed from Figure 3.5, performance improvement is not significant beyond that value, and we set $L = 4$ as the optimal tradeoff value described above.

Using these fixed values, we analyzed the effect of w, the length of the quantization buckets, with Figure 3.7 showing that w has a great impact on AD accuracy, although its optimal value depends on the characteristics of the dataset. Consequently, when $T = 50$ and $L = 4$, performance can be optimized by simply tweaking w. This simplifies the hyperparameter tuning process, which is merely a matter of finding a suitable w value for the given dataset.

For an unlabeled dataset, however, the effect of w on AD detection accuracy cannot be directly observed, since no labels are available to measure performance. To obtain more information, we thus extracted other indirect unsupervised metrics

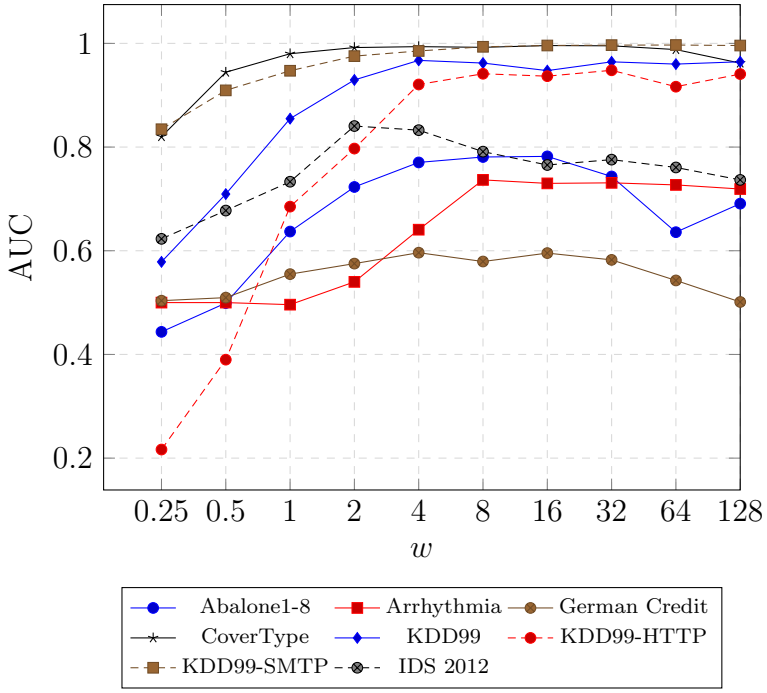


Figure 3.7: LSHAD performance changing the hyperparameter w, the quantization bucket length.

to observe if they were correlated with algorithm performance:

- BC: number of buckets generated.
- ABS: Let $|D|$ be the cardinality of the input data and let b be each bucket size from B buckets generated. Hence:

$$ABS = \frac{\frac{\sum_{b \in B} b}{BC}}{|D|} \tag{3.7}$$

- ABD: Average Euclidean distance of the first element in the bucket to its neighbours.

To assess the suitability of these metrics, we explored several w values for the datasets and plotted each metric versus the AUC. While no pattern was observed for the BC and ABD metrics, the ABS was found to contain useful information. For

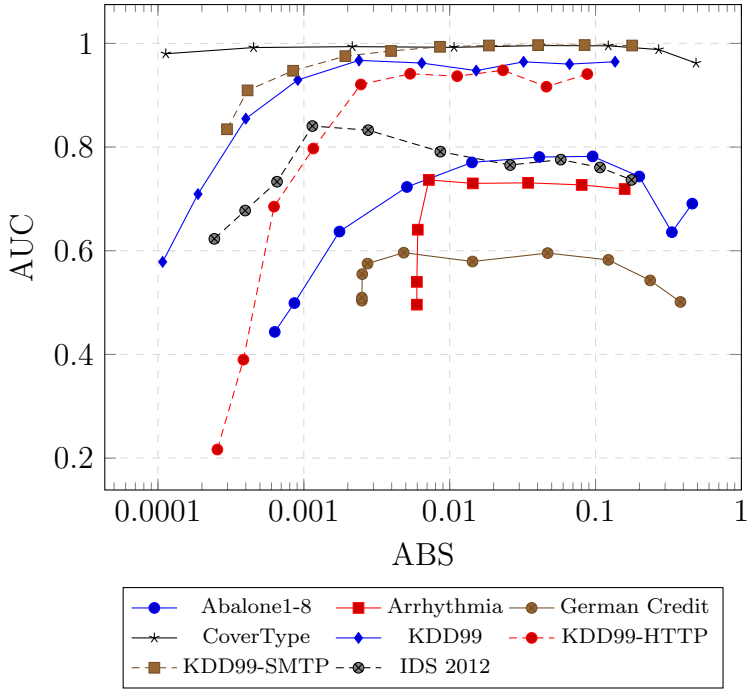


Figure 3.8: LSHAD performance for different w values, with the ABS metric on the horizontal axis.

most of the tested datasets, LSHAD performance was much improved when ABS was in the range $[0.05, 0.1]$, as can be observed in Figure 3.8. ABS can therefore be used to obtain a suitable value for w since a w value that lands ABS in the $[0.05, 0.1]$ range will be likely to achieve good AD accuracy. Moreover, since the effect of w on ABS is known (a larger w increases ABS, while a smaller w decreases ABS), the search for a suitable w can be performed efficiently.

3.4.3. LSHAD with hyperparameter autotuning

Algorithm 3 depicts our LSHAD model, which takes into account the ABS metric above. It begins by estimating a suitable value of w using a binary search.

A search interval must first be set, for which the lower threshold is always set to 1 (line 2). The upper threshold is found by doubling the w value and using it for hashing until small enough buckets result (line 3). That range is then explored

Algorithm 3: Pseudocode for LSHAD:. Hyperparameter training.

Input : $D \leftarrow$ Set of training points
Output: $L, T, w \leftarrow$ tuned hyperparameters

```
1  L ← 4, T ← 50;
2  wCandidate ← 1, avBucketSize ← 0, leftLimit ← 1, rightLimit ← 1;
3  while avBucketSize < 0.05 do
4  |   avBucketSize ← HASHGROUPANDCOUNT(D, L, T, wCandidate);
5  |   wCandidate ← wCandidate * 2;
   end
6  rightLimit ← wCandidate;
7  while avBucketSize < 0.05 or avBucketSize > 0.1 do
8  |   wCandidate ← ⌊(leftLimit + rightLimit)/2⌋;
9  |   avBucketSize ← HASHGROUPANDCOUNT(D, L, T, wCandidate);
10 |   if avBucketSize < 0.05 then
11 |   |   leftLimit ← wCandidate;
   |   end
12 |   if avBucketSize > 0.1 then
13 |   |   rightLimit ← wCandidate;
   |   end
14 |   if leftLimit >= rightLimit then
15 |   |   break;
   |   end
   end
```

Result: $L, T, wCandidate$

using a binary search (loop on line 7) to find a value for w that produces buckets with an average number of elements between 0.05 and 0.1 times the size of D. Once found, L,T and the retrieved w are reported as the tuned hyperparameters.

3.4.4. Estimator experiments

Regarding the proposed estimators in Section 3.3, namely, $E_A(h)$, $E_B(h)$, $E_C(h)$, $E_D(h)$, we compared their AUC performance for each given dataset to determine which produced the best ranking score for classification. Figure 3.9 depicts a graph showing the AUC score for each estimator, showing that they all produced similar results with small variations in the AUC for different datasets. In fact, the statistical Nemenyi post-hoc test [56] with $\alpha = 0.05$ could not significantly differentiate the

estimator scores (see Figure 3.10). We chose our algorithm to use the C estimator by default as it was the estimator with the lowest critical difference value.

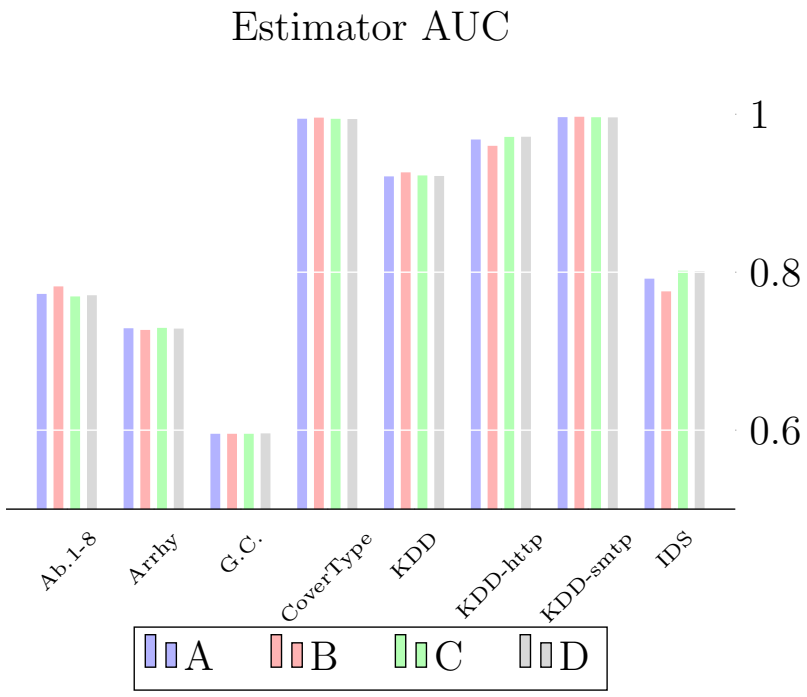


Figure 3.9: AUC scores for the different estimators

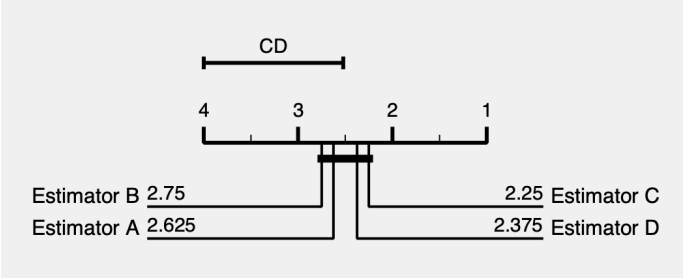


Figure 3.10: Nemenyi statistical test for the estimator AUC scores

3.5. Performance Evaluation

Below we evaluate the LSHAD algorithm compared to other methods in terms of processing time and AD performance.

3.5.1. Applied Methods

To measure and compare the performance of our method against other methods, a variety of state-of-the-art algorithms (many referred to in Section 3.2) were selected. As LSHAD is a density-based method, we first selected the well-known LOF and LOCI methods and used Euclidean (E), Jaccard (J) and Hamming (H) distances, for which we employed a Matlab implementation[5]. From the same category, we also selected the approach by Zhang et al.[6] [158], as it also uses LSH for scalable density estimation; in this case, we tested their JAVA implementation[7] of 4 algorithms using their PDBS, namely:

- **1 Sample PDBS** (1 Samp PDBS)- drawing one sample for all points to compute the k-NN distance
- **Iterative PDBS** (Ite PDBS)- drawing one sample for each point to compute the k-NN distance
- **Iterative+Ensemble PDBS** (Ite+Ens PDBS)- drawing multiple samples for each point to make ensembles for the k-NN distance
- **Isolation Forest PDBS** (IForest PDBS)- using the IForest detection method.

Other methods related to unsupervised AD were also selected for our evaluation: the Autoencoder implementation in Python using the Elephas[8] framework, an extension of Keras that allows distributed deep-learning models to be run at scale with Spark; the One-Class SVM with radial basis (SVM-R) and linear (SVM-L) kernel functions, for which we used the Matlab LibSVM interface[9]; a distributed version of SVM that can handle large datasets (DOC-SVM) [39]; an online one-class classifier with a PA-I [126], also built-in Matlab; and finally, ADMNC implemented in Scala-Apache Spark[10].

[5]https://github.com/jeroenjanssens/lof-loci-occ
[6]From the LSH methods presented in Section 3.2, this is the only algorithm available to test
[7]https://bit.ly/2ugZQ0x
[8]http://maxpumperla.com/elephas/
[9]https://www.csie.ntu.edu.tw/ cjlin/libsvm/
[10]https://github.com/eirasf/ADMNC

All the methods were tested with several datasets with different compositions in order to observe algorithm behaviour in a variety of scenarios. We first used a simple synthetic dataset[11] suite with just 3 dimensions and 1000 samples representing different shapes as described in Table 3.2. We also used the real datasets described in Section 3.4.

In our experiments we performed five-fold cross-validation, filtering around 1% of the class anomaly samples for each dataset with the aim of simulating a real AD scenario, as done in Section 3.4 to test the different hyperparameters. Note that, to overcome computational difficulties for some methods, we only used 2 folds rather than 5 folds in testing the medium datasets. In addition, as we use the AUC metric for evaluation, we ignore the threshold variable described in Section 3.4.3 and use the anomaly score provided by the estimator as the LSHAD output. Used for the experiments was a MacBook Pro laptop with 8GB of RAM memory and a 2.9GHz Intel Dual-Core i5 processor.

3.5.2. AD performance comparison

For visualization purposes, we split the tables according to the datasets' structure. Table 3.3 shows AUC results for the synthetic datasets, Table 3.4 shows algorithm performance results for small datasets with fewer than 5000 samples (Abalone, Arrhythmia, German Credit, Heart, Pima Diabetes, Breast Cancer) and finally, Table 3.5 shows algorithm performance results for medium datasets, with more than 5000 samples (CoverType, KDDCup99 datasets, IDS 2012).

3.5.2.1. Synthetic datasets

We first made a comparison of the algorithms for a simple classification task, applying 5 datasets of different shapes with 2 dimensions each, represented in Figure 3.4. Table 3.3 shows that our LSHAD method obtains state-of-the-art results in AD for the different data shapes, except for the 2 circular clusters. While LSHAD obtains the best performance for the 2-point clouds with variance, overall the best performance is achieved by the much more exhaustive LOF and LOCI methods.

[11] https://zenodo.org/record/1171077#.XkE-HBP7TOR

Table 3.3: Selected algorithm AUC results for 5 synthetic datasets

	2BC	**2CC**	**2PV**	**3AC**	**3PC**
LOF (E)	82.78	78.40	78.40	**90.20**	96.90
LOF (H)	83.03	78.60	79.58	**90.20**	96.80
LOF (J)	**83.10**	**78.62**	79.73	90.10	96.80
LOCI (E)	80.46	76.20	75.61	88.12	94.80
LOCI (H)	80.63	78.20	76.57	88.64	95.20
LOCI (J)	80.11	75.40	76.65	87.81	96.87
SVM-L	50.13	52.00	53.40	56.80	62.80
SVM-R	72.36	56.54	81.30	83.00	91.80
DOC-SVM (RBF)	55.40	51.80	59.80	66.20	60.80
PA-I	55.77	58.00	56.60	64.20	70.80
ADMNC	53.80	59.29	70.26	85.32	82.70
Autoencoder	59.80	56.70	68.54	70.32	69.79
1 Samp(PDBS)	69.71	54.38	81.47	82.70	95.08
Ite (PDBS)	68.90	55.00	80.09	83.13	95.68
Ite+Ens(PDBS)	76.18	57.68	81.22	86.51	97.06
IForest(PDBS)	61.47	56.47	66.93	78.51	72.50
LSHAD	76.34	61.07	**82.29**	89.30	**97.34**

3.5.2.2. Real datasets

For the small datasets, from Table 3.4 it can be seen that AUC scores are very variable. Although LSHAD did not obtain the best score in any dataset, its results are average state-of-the-art, and in some cases close to the best. For the medium datasets, Table 3.5 reports a similar outcome. Note that LOF and LOCI were excluded from the comparison, as their quadratic complexity made them computationally excessively costly in managing large datasets, nor was it possible to test DOC-SVM, as its Matlab implementation failed in trying to split large datasets.

3.5.2.3. Statistical test evaluation

We ran a statistical Nemenyi post-hoc test [56] (See Appendix A Section A.4) with $\alpha = 0.05$ to check for any significant statistical difference between methods. In Figure 3.11, which shows the algorithms sorted by score, it can be observed that the

[12]This is a sample of the IoT-23 Subset with ID 1

Table 3.4: Selected algorithm AUC results for small real datasets

	Ab. 1-8	**Ab. 9-11**	**Ab. 11-29**	**Arrhyth**	**GC**	**Heart**	**Breast**	**Pima**
LOF (E)	69.36	60.29	59.27	66.70	58.47	61.22	60.21	68.38
LOF (H)	69.36	60.29	59.27	69.83	56.46	69.58	59.18	68.18
LOF (J)	69.36	60.29	59.27	70.10	56.81	65.28	60.17	68.23
LOCI (E)	85.24	67.56	71.55	67.35	59.17	**86.42**	**99.51**	**73.48**
LOCI (H)	**85.26**	68.56	71.55	71.41	57.09	72.25	99.37	69.87
LOCI (J)	85.15	**68.74**	71.59	71.44	56.63	85.39	99.40	72.75
SVM-L	79.44	61.40	76.70	67.94	56.97	85.16	99.50	59.77
SVM-R	81.21	67.56	74.48	74.79	**64.52**	81.14	97.76	67.10
DOC-SVM	55.61	57.48	55.02	65.30	54.19	53.83	74.97	67.12
PA-I	84.98	65.11	71.13	69.32	62.16	71.02	69.33	55.90
ADMNC	84.53	61.20	**79.30**	61.40	62.76	72.31	91.34	59.20
Autoencoder	82.23	58.34	67.76	**79.54**	64.00	83.20	97.90	67.10
1 Samp(PDBS)	70.85	52.72	68.82	73.06	53.70	68.00	98.60	70.16
Ite(PDBS)	70.07	50.75	68.78	71.93	54.10	59.67	98.62	68.90
Ite+Ens(PDBS)	73.80	50.97	73.31	72.60	54.50	62.67	98.30	72.10
IForest(PDBS)	84.61	55.60	70.66	72.10	55.60	53.83	91.63	53.98
LSHAD	77.24	53.82	67.13	72.22	58.23	79.95	98.58	71.13

Table 3.5: Selected algorithm AUC results for medium real datasets

	CT	**KDD99**	**KDD99h**	**KDD99s**	**IDS**	**IOT-23[12]**
Autoencoder	98.95	**99.13**	**99.99**	99.69	80.44	88.05
1 Samp(PDBS)	96.59	93.29	59.90	99.68	53.64	93.83
Ite(PDBS)	95.39	84.96	59.90	99.69	54.45	93.67
Ite+Ens(PDBS)	98.88	90.93	59.40	99.69	55.29	93.60
IForest (PDBS)	99.50	96.67	94.81	99.73	92.99	93.70
PA-I	99.49	98.90	99.50	95.92	**96.50**	73.55
SVM-R	99.53	95.35	99.91	99.32	61.61	73.96
SVM-L	95.01	69.37	99.95	99.51	80.66	77.01
ADMNC	57.94	94.05	91.62	88.26	56.75	93.29
LSHAD	**99.66**	97.74	99.44	**99.85**	87.32	**93.92**

Nemenyi test divided the algorithms in 3 groups, represented by horizontal thick lines. LSHAD was placed in the group of algorithms with the best performance, for which there is no statistical difference.

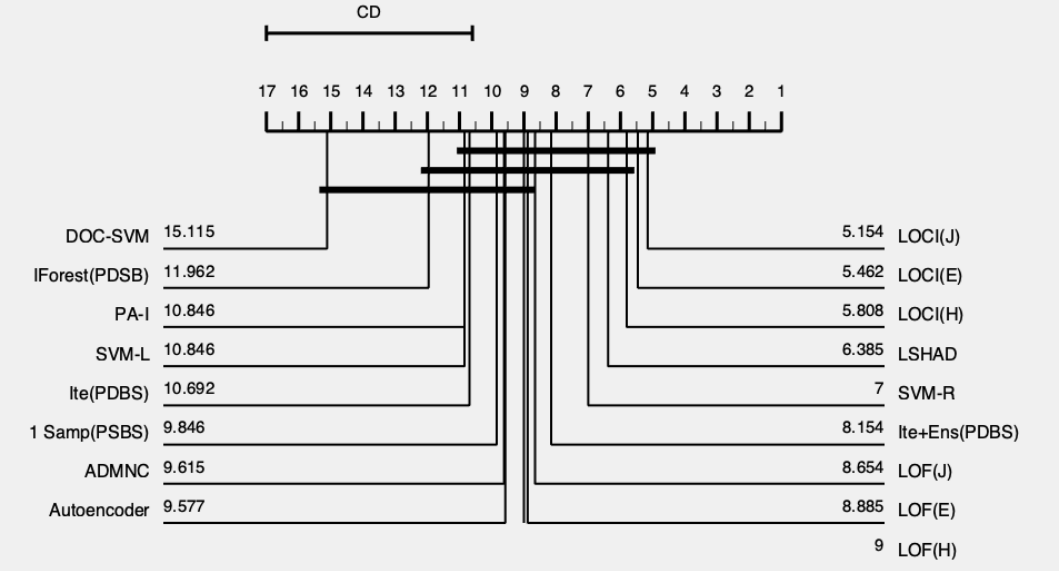


Figure 3.11: Nemenyi statistical test to evaluate AUC scores for AD methods

3.5.3. Scalability testing

To test method scalability, we used a synthetic dataset from the generator developed by Eiras-Franco et al. [62] [13]. Varying size, we started with 100 samples and increased the sample 5 times for each iteration. Since the methods are implemented on different platforms, we measured relative algorithm execution time as the ratio between the processing time for the first dataset with 100 samples and the processing time for each other specific dataset size. This allowed us to approximate the empirical time complexity of each method. Selected for this test were the LOF and LOCI methods with Hamming distance, SVM-L, SVM-R, Autoencoder, DOC-SVM, PA-I, ADMNC, and IForest(PDBS) (as the fastest of the 4 PDBS methods). Figure 4.4 depicts the execution time results of each algorithm, showing that all the algorithms process the data very rapidly for small datasets (100 and 500 samples), except LOCI and LOF (given their quadratic complexity). DOC-SVM was unable to process datasets with more than 2500 samples due to its current implementation, and needed more time to process the small datasets compared to the other algorithms; PA-I execution time started to increase significantly for datasets with more than 2500 samples, exceeding linear complexity; SVM-L exhibits quadratic

[13] http://github.com/eirasf/ADMNC/

complexity; IForest(PDBS), although showing acceptable execution times for small datasets, could not handle datasets of more than 62500 samples; and SVM-R performed adequately up to 12500 samples, then slowed down considerably, exceeding quadratic complexity.

For the larger datasets, ADMNC, Autoencoder, and LSHAD achieved the best execution times, while LSHAD showed the lowest complexity when handling the largest amount of data (1562500 samples).

An experiment was also carried out with the IoT-23 dataset since it has some large subsets in the order of 7GB, rounding 70,000,000 records [153]. Only the LHSAD, ADMNC, and Autoencoder algorithms were used, given the evidence that they could deal with large datasets, given their distributed approach. The algorithms were applied to each IoT-23 dataset subset and five-fold cross-validation was performed. The resources of the CESGA were used, consisting of 22 machines with 35GB of RAM and 22 cores each.

Table 3.6: AUC results for LSHAD, ADMNC, and Autoencoder for IoT-23 datasets

ID DATASET	**LSHAD**	**ADMNC**	**Autoencoder**
1	89.60 $\pm$0.87	**91.95 $\pm$ 1.87**	62.58 $\pm$ 0.0038
3	**99.53 $\pm$ 0.12**	95.45 $\pm$ 0.61	96.80 $\pm$ 0.0011
7	99.94 $\pm$ 0.02	99.68 $\pm$ 0.43	**99.71 $\pm$ 0.00023**
9	**99.97 $\pm$ 2.42**	64.99 $\pm$ 15.72	99.89 $\pm 8.96e-9$
17	71.76 $\pm$ 21.05	97.22 $\pm$ 1.06	**99.99** $\pm 6.79e-5$
33	76.42 $\pm$ 5.08	**83.31 $\pm$ 18.26**	51.81 $\pm$ 0.017
35	98.48 $\pm$ 1.38	**99.84 $\pm$ 0.06**	95.21 $\pm$ 0.017
36	99.77 $\pm$ 0.29	99.36 $\pm$ 1.21	**99.99** $\pm 8.99e-8$
39	97.42 $\pm$ 0.21	76.98 $\pm$ 2.80	**99.99 $\pm$ 4.54e-5**
43	91.29 $\pm$ 3.23	**99.99 $\pm$ 0.0005**	59.72 $\pm$ 0.038
48	**99.78 $\pm$ 0.19**	99.55 $\pm$ 0.78	99.58 $\pm 9.02e-6$
49	**99.52 $\pm$ 0.15**	99.37 $\pm$ 0.30	99.27 $\pm 132e-5$
52	94.19 $\pm$ 3.55	99.61 $\pm$ 0.57	**99.99** $\pm 1.71e-7$
60	99.65 $\pm$ 0.17	99.80 $\pm$ 0.15	**99.99** $\pm 7.18e-6$
Avg. AUC	**94.09**	93.36	92.82

Table 4.5 shows that the overall average AUC for LSHAD was slightly better than for ADMNC and Autoencoder. However, the 3 algorithms outperformed each other in specific scenarios. Results were similar, at around 99% AUC, for subsets 3,

7, 35, 36, 48, 49, 52, and 60, while differences occurred with the remaining subsets: for subsets 1, 33, and 43: LSHAD and ADMNC outperformed Autoencoder, for subset 9 and 39, LSHAD and Autoencoder outperformed ADMNC; and for subset 17, ADMNC and Autoencoder outperformed LSHAD. LSHAD thus produced similar or better results than ADMNC or Autoencoder for all subsets except subset 17.

While LSHAD achieved the best average AUC, slightly better (1%) than its competitors, overall the three methods did an excellent AD job for this dataset. Autoencoder had the lowest average AUC, but only performed poorly with 3 datasets (1, 33, 43); its higher standard deviation on those datasets indicates difficulty in adjusting the parameters. While the reasons are difficult to ascertain, due to the lack of transparency and interpretation of this method (it operates like a black box), we can deduce the possible cause. First, dataset 33 is unbalanced, as only 2.54% represents benign data. This quantity of normal activity may not be sufficiently representative, causing the Autoencoder to generate noise when reconstructing its input. Moreover, for dataset 33 (Kenjiro attack type capture), data distribution may be noisy, as the performance of both LHSAD and ADMNC with this dataset was also poorer relative to their results for the other datasets. Second, while datasets 1 and 43 have balanced classes, the problem may lie in a loss of important information in the compression phase, as autoencoders are lossy [47] in the degradation that occurs in compression. The density-based methods using the hashing (LSHAD) and Gaussian mixture model (ADMNC) techniques function better for the specific distributions in these datasets. Comparing LSHAD with ADMNC, ADMNC slightly outperformed LSHAD in several datasets. Nonetheless, the weakest performance of LHSAD was an impressive AUC of 71%.

Note that the optimal values defined for LHSAD hyperparameters tested on medium datasets (Section 3.4) also hold for large datasets, as indicated by the high-performance results. This would suggest that LSHAD is suitable for processing large-dimension datasets, with acceptable accuracy rates, as it is among the best-performing algorithms and also is among the most scalable methods.

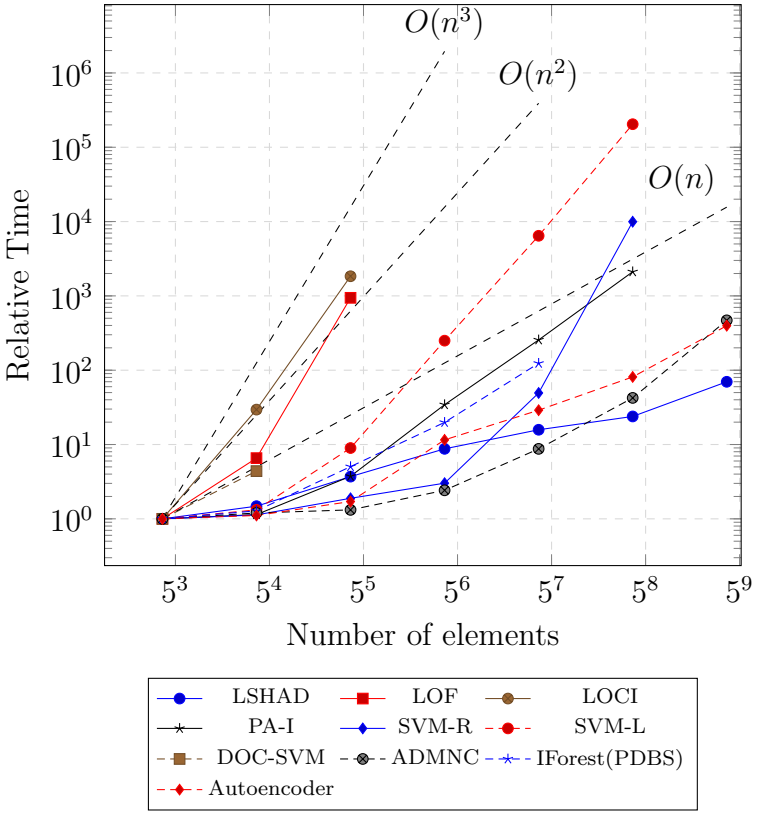


Figure 3.12: Execution time of each algorithm increasing the size samples of the Synthetic dataset. Axis are represented using logarithmic scale

3.5.4. Scalability versus AD performance

We used the Pareto optimization method [187] to evaluate the tradeoff between scalability and AD for the algorithms. In multi-objective optimization, the Pareto front is defined as the border between the region of feasible points (not strictly dominated by any other) for which all constraints are satisfied and the region of unfeasible points (dominated by others).

Figure 4.8 plots all the algorithms used in our study, maximizing the average AUC (X axis) and minimizing processing speed (Y axis). To compute t time complexity we used the number of samples of the largest dataset n that each algorithm was capable of handling and the processing time t required, that is, $\frac{log(t)}{log(n)}$. Figure 4.8 shows that LSHAD, LOCI, and SVM-R are on the Pareto front, although note

that LOCI and SVM-R were unable to process the largest datasets.

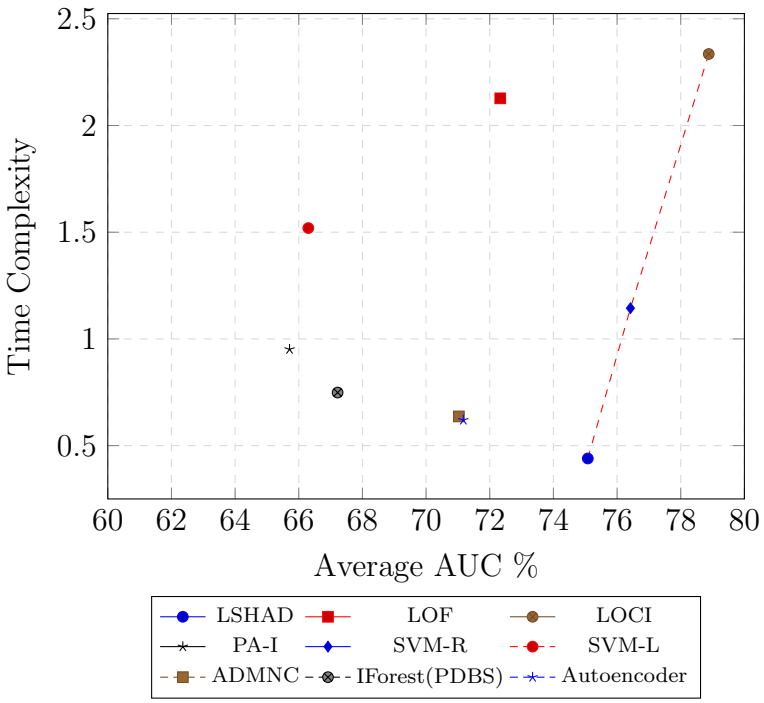


Figure 3.13: Pareto front of a multi-objective optimization problem based on mean AD performance for all datasets (higher is better) versus time complexity (smaller is better)

In summary, in our experiments for accuracy and scalability, LSHAD is demonstrated to be among the best state-of-the-art methods, and has the additional advantage of hyperparameter autotuning.

3.6. Conclusions

LSHAD is a novel algorithm based on the LSH technique, developed in order to obtain an AD model that could handle large-scale datasets. We leverage LSH, which enables groups of similar data points to be detected, to estimate the density of the input space regions, which is used, in turn, to estimate the probability of a data point being an anomaly.

Our algorithm, implemented in the Apache Spark framework, is tailored for distributed environments and so is capable of processing large datasets due to its scalability properties. An important advantage of our method is its AutoML feature, which implements automatic hyperparameter tuning and thereby reduces computational resource needs and the time required for manual hyperparameter tuning.

The LSHAD algorithm was compared for AD and scalability performances with state-of-art methods in a variety of datasets. Our empirical study demonstrates that LSHAD is comparable to the best available methods in achieving satisfactory AD results for both synthetic and real datasets, and performs better than other methods in terms of scalability, especially with very large datasets. In summary, our contributions are as follows:

1. We propose a novel AD method based on LSH that obtains accuracy results on a par with SotA methods and scalability results that outperform those of any of its competitors.

2. The model manages distributed scenarios, as it was developed using the Apache Spark framework and so can distribute data processing across multiple clusters.

3. The model automates the time-consuming and error-prone hyperparameter tuning process, which not only improves efficiency but also makes the algorithm available to non-expert users in the ML field, currently not a feature of most AD models.

Additionally, the scalability and efficiency of the LSHAD algorithm make it a sustainable solution for large-scale AD tasks. The ability to process large datasets and to reduce computational resource needs through automatic hyperparameter tuning reduces the environmental impact of the algorithm, making it a sustainable choice for AD applications in the long term.

Chapter 4

Novel unsupervised methods applied in IoT intrusion Detection

With the development of technology and information systems, the IoT concept is increasingly present in our daily lives. It is now possible to interact with different smart objects connecting the physical and the digital worlds through the Internet. Thus, it will also be possible to record data related to our actions more effectively and use this information to our advantage to integrate services and applications. However, the advancement of such technology comes along together with significant risks and vulnerabilities.

IoT has brought major benefits to society and industries, however, the security of such technology has not yet matured. The increased number of connected IoT devices will provide more opportunities for attackers to obtain access and utilize them in large-scale attacks. Securing IoT devices is becoming increasingly difficult for both consumers and manufacturers [2].

The great dependence on this technology both at a personal and business level highlights the importance of information security these days. Smart-connected objects will become even more common than smartphones are nowadays and, thus, access to personal and sensitive data should be restricted and monitored. Ensuring confidentiality, integrity, and availability is a great challenge since these systems are subject to various types of attacks, some to stop services, others to steal information [69].

The lack of security measures and dedicated AD systems for these heterogeneous networks make them vulnerable to a range of attacks, such as Denial of Service (DoS/DDoS), spoofing, data leakage, causing damage to hardware and system blackouts, disrupting the system availability and even physically harming individuals [12, 13].

As mentioned and described in Chapter 2, IDS have been developed over the years as a solution to face attacks performed in computational systems connected to the Internet. Such systems are usually classified regarding the detection methods used [206]. These systems are differentiated into four categories: textbfSignature-based IDS using signatures stored in the internal database to identify network behaviours that match such signatures; **Anomaly-based IDS** which usually apply statistical or Machine Learning techniques to construct the normal behaviour profile. This approach is effective to detect new attacks; **Specification-based IDS** similar to the Anomaly-based IDS, these systems identify deviation from normal behaviours but using rules and thresholds (specified by a human expert) that define the expected behaviour for network components such as protocols, nodes and routing table; **Hybrid IDS** where this category uses concepts from signature, anomaly and specification-based IDS to maximise their advantages and minimise their limitations.

Signature-based IDS systems such as Bro [155] and SNORT [166] are ineffective within IoT ecosystems since they only work with traditional IP-only networks [203]. According to Zarpelão et al. [206], research on IDS schemes appropriate for IoT is still incipient. Thus, solutions available in the literature do not cover a wide range of intrusions and IoT technologies. Thereby, we aim at analysing various unsupervised learning AD techniques as a solution to be used in an Anomaly-based or Hybrid-based IDS for IoT devices. As was seen in previous chapters, large unlabeled datasets are a challenge for AD models, especially in critical domains such as Intrusion Detection. Explainability is crucial for understanding the behaviour of models, especially when it comes to identifying and handling anomalies. Additionally, tuning the hyperparameters of models is a problem that needs to be addressed when working with large datasets. These challenges are further exacerbated in critical domains where the consequences of misdetection can be severe. Therefore, in this chapter, we evaluate a range of traditional unsupervised AD methods, as well as two novel approaches, the LSHAD presented in the previous chapter and EADMNC, on

an IoT intrusion detection large dataset. Our evaluation will focus on both AD rates and time processing performance, as well as the usefulness of the explanation trees obtained from the EADMNC method.The novel LSHAD and EADMNC methods not only demonstrate strong performance on both AD, but they are also designed with sustainability in mind. The efficient use of computational resources and the automatic hyperparameter tuning feature of the LSHAD algorithm make it a highly sustainable solution for anomaly detection in real-world scenarios. The scalability properties of both methods also ensure that they can handle large datasets, reducing the need for additional computational resources to be used. Overall, this chapter will provide a thorough analysis of the strengths and limitations of traditional and novel AD methods, highlighting the importance of both accuracy and efficiency in the selection of an appropriate approach. The main contributions provided in this chapter are:

- The analysis and data pre-processing description of the new IoT-23 dataset recently developed by A. Parmisano et al. [153].
- The evaluation and comparison, in terms of detection performance and scalability, of a suite of 8 different unsupervised algorithms over the IoT-23 dataset.
- One of the methods in the comparison suite, named EADMNC [24], generates explanatory trees. Such trees provide knowledge and understanding regarding the IoT attacks performed in the IoT-23 dataset. In this work, a human expert in the cybersecurity field analyses the results obtained.
- Two of the eight methods are scalable (LSHAD [135], and the previously mentioned EADMNC [24]), and in this work, we include a comparison of these two detection methods in terms of their performance detection using all the data from each subset.

The rest of this chapter is organised as follows: Section 4.1 reviews SotA IDS designing solutions as well as machine learning methods applied in IoT for intrusion and AD; Section 4.2 analyses the new IoT-23 dataset; Section 4.3 presents all the unsupervised learning algorithms used for performance evaluation; Section 4.4 describes the evaluation and comparison of the methods in terms of AD and execution time performance and also presents an analysis of the attacks in IoT-23

dataset using the explainability property of EADMNC, highlighting its importance for understanding the execution process behind IoT attacks. Finally, Section 4.5 summarises the main conclusions of our work.

4.1. Related Work

AD in the IoT infrastructure is a growing interest. Thus, even though some work has been carried out in this area, it is still a very active field of research.

In [86], Hasan et al. compared several ML models in their performance to predict attacks and anomalies in IoT systems. They concluded that Random Forest models had the best overall performance for their particular study using a dataset from the Kaggle repository provided by Pahl et al. [149].

Brun et al. present a methodology using a deep learning approach with dense random neural networks for the online detection of network attacks against IoT gateways [28]. Their approach can predict the probability that a network attack is ongoing from a set of metrics extracted from packet captures.

Vu et al. [196] proposed a novel representation learning method to better predictively "describe" unknown attacks, facilitating supervised learning-based AD methods. The authors developed three regularized versions of AutoEncoders to learn a latent representation from the input data. To evaluate the performance of the proposed models, the authors did experiments on nine recent IoT datasets[132]. The experimental results showed that new latent representation could significantly enhance the performance of supervised learning methods in identifying unknown IoT attacks.

Al-Hawawreh et al. [140] used deep learning models to develop an AD technique for Internet industrial control systems that can learn using information collected from TCP/IP packets. Regarding deep learning models, the authors used an AutoEncoder algorithm for producing the optimal parameters when learning normal network behaviours. Autoencoder optimal parameters are then used, as an effective tuning mechanism, in a standard supervised deep neural network model. Their model was tested on two benchmark datasets, the UNSW-NB15 [139] and the NSL-

KDD [185]. Their method achieved a high detection rate and low false alarms compared with other techniques developed in recent studies.

Bostani and Sheikjan [23] proposed a real-time hybrid intrusion detection framework consisting of two IDS-based type modules: specification-based and anomaly-based to detect sink attacks and selective forwarding. The authors used agents in their specification-based module to analyze the behaviour of host nodes and send results to the route node through normal data packets. The agents using incoming data packages in the anomaly-based module employ the unsupervised optimum-path forest algorithm for projecting clustering. Their model can work in a distributed platform due to its MapReduce-based architecture [205]. It can detect anomalies in parallel as a global detection approach. Their proposed method has also been extended to detect wormhole attacks.

Eskandari et al. [64] presented the anomaly-based IDS named PassBan to secure IoT devices. The solution can take full advantage of edge computing to detect cyber threats as it can be deployed on cheap IoT gateways. Their IDS detected common cyber-attacks, namely Port Scanning, HTTP Login Brute Force, SSH Login Brute Force, and SYN Flood attacks. The authors showed that PassBan could detect almost all malicious traffic with very low False Positive rates and relatively high accuracy.

Deploying an IDS on IoT devices can be a solution for protecting them from intrusions [183, 14, 146]. However, most IDS are not prepared for problems such as rare malicious or even benign activities which are not present in the training data, thus possibly causing an increase of false positives and negatives. Also, many models do not scale well when dealing with large amounts of data. Thus, in this work, we present an empirical evaluation with several unsupervised learning techniques as fittable solutions to integrate with an IDS for IoT capable of dealing with the mentioned problems. Since these techniques aim to detect anomalous patterns that are not present in the training phase, by implementing them together with supervised techniques, it should be possible to reduce the false positive/negative rate [145, 112, 106]. Besides, some of these methods can handle large amounts of data due to their distributed characteristics. A summary of the related work discussed in this section is represented in Table 4.1.

Table 4.1: Related Work summary

Works	Dataset	Methods	Evaluation Metrics
[86]	DS2OS [149]	RF, LR, SVM, DT, ANN	Accuracy, Precision, Recall, ROC curve
[28]	Simulated dataset – not available	Dense RNN	Nº: UDP, ICMP, packages (Long/short time), broadcast messages, diff. btw. established connections.
[196]	Nine IoT datasets [132]	Autoencoder	False Alarm Rate, Miss Detection Rate, AUC
[140]	UNSW-B15 [139], NSL-KDD [185]	Autoencoder + DNN	Accuracy, Detection Rate, False Positive Rate
[23]	Simulated dataset – not available	Optimum Path Forest	True Positive Rate, False Positive Rate, Accuracy
[64]	Simulated dataset – not available	Isolation Forest, Local Outlier Factor	Confusion Matrix, Precision, Recall, F1 Score

4.2. IOT-23 Dataset preparation and Analysis

IoT-23 is a recent dataset of network traffic captured from IoT devices from A. Parmisano et al. [153], published in January 2020, with captures ranging from 2018 to 2019. This is the large-scale dataset over which the different AD methods were evaluated and compared. The dataset is described in detail in Appendix A Section A.1.3.

The IoT-23 dataset went through a preparation process before applying the AD algorithms: We first discarded the features **ts** and **uid** since they generate a unique value for each record and therefore do not offer relevant or generalisable information to assist the algorithms in pattern detection. As the unsupervised methods used in our work cannot deal with strings, we converted all string values from **proto, service, conn_state, local_orig, local_resp** and **history** features into numerical values. In cases where **"-"** appeared in data, referring to empty values generated by the Zeek analyser tool, we replaced it with the value **-1**. Concerning IP address features (**id.orig_h , id.resp_h**) we applied Unigram, Bigram, and Trigram feature extraction techniques. The idea is to extract three features from the IP address to assist the AD algorithms in identifying patterns through subnets. Consequently, requests from unexpected IP addresses or unexpected requests from known IP addresses could be indicators of potentially malicious activities. We first separated all IP address elements by the dots. Each feature is a representation of one or several IP address elements. The first feature denominated by IP Unigram represents the first IP address element. The second feature, named IP Bigram, has the concatenation of the first and second element of the IP address. The third and last feature, IP

Trigram, contains the three elements of the IP address, all concatenated. It is important to note that the second and third elements of the IP address are composed of three elements, even if their value is less than 100. In that case, zeros are added to the left (see Table 4.2 as an example).

Table 4.2: Unigram, Bigram, Trigram, feature extraction example technique

IP Unigram	**IP Bigram**	**IP Trigram**		
192	192168	192168001	=>	192.168.1.199

We also ignored the different types of attacks, as our goal is to identify anomalies, and we are using unsupervised learning methods. Thus our idea is to simulate a real environment that can potentially be targeted by unknown attacks. We only considered two classes for the label feature. The "Benig" class is represented by value 0, and the "Malicious" class is represented by value 1, corresponding to any attack or suspicious activity in the IoT devices.

As can be seen in Table A.2, the IoT-23 dataset contains large subsets of data. Because most of the algorithms chosen cannot handle large amounts of data, we selected a random sample from each subset, containing approximately 10 000 records. When extracting each sample from the subset, we take care of balancing the classes. There were cases in which balancing the classes was not possible due to the small size of the subset or the small percentage of records of a particular class (e.g., ID subsets 8, 20, 21, 34, 36, 42, 44, 48, 49, 52 and 60). For these cases, we tested the algorithms, keeping the original proportion of classes of each subset.

We generated several histograms to understand the distribution of each feature in each subset. As an example, in Figure 4.1 we can see the histograms from the CTU-IoT-Malware-Capture-17-1 subset. We verified that each feature distribution was different in each scenario except for the 'local_orig' and 'local_resp' features. For these two features, the values were always 0 in all scenarios, meaning in this particular case, that the originated and response connections were given as undefined by the Zeek analyzer tool.

Regarding the correlation between features (see example plot from Figure 4.2), it can be observed that in most scenarios, the features 'duration', 'orig_bytes', 'resp_bytes', 'orig_pkts', and 'resp_pkts' are strongly correlated. The longer the

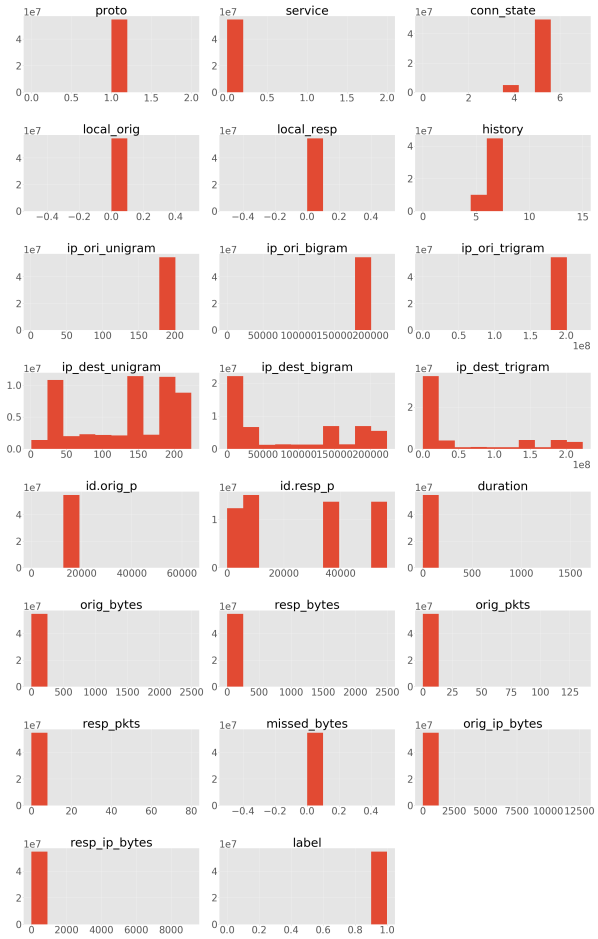


Figure 4.1: Histograms of each feature for the CTU-IoT-Malware-Capture-17-1 subset.

connection lasts, the more information (i.e. packets, bytes) is received/sent between users/devices, being this the reason for correlated features. This correlation also happens with 'ip_ori_unigram', 'ip_ori_bigram', 'ip_ori_trigram' since these features arose from the same original feature. Before applying the AD algorithms, we normalized each subset to prevent some methods from giving more importance to features with large numeric values.

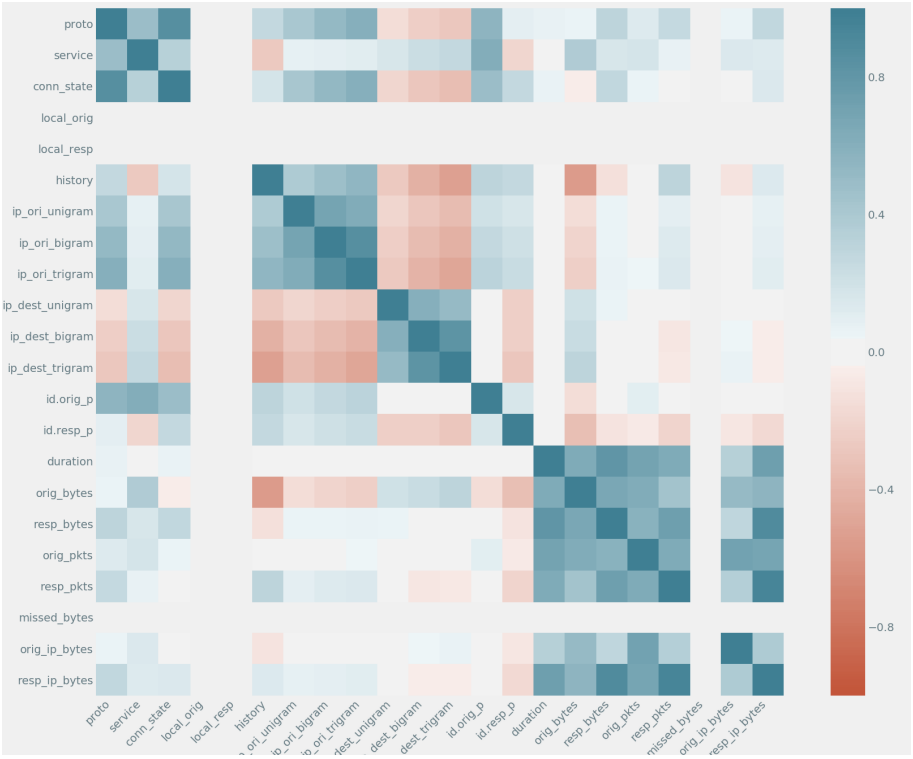


Figure 4.2: Correlation plot of CTU-IoT-Malware-Capture-17-1 subset.

4.3. Methods used

We selected a variety of SotA unsupervised learning methods used in AD problems to measure and compare their performance, including two recently developed methods from Eiras-Franco et al. [62] and Meira et al. [135] (the last one described in Chapter 3), both with distributed properties that can handle large amounts of data.

We carried out the evaluation using eight different algorithms: OC-NN, OC-SVM, LOF, IForest, PA-I, Autoencoder, EADMNC and LSHAD. All of them are described in Appendix A Section A.2.

Table 4.3 represents a summary of the selected methods identifying for each one certain characteristics: Their *Scalability*, in order words, the property of handling a large amount of data; their *Explainability*, meaning the capability of justifying their

[1]It has several hyperparameters, with only one being automatic tuned

Table 4.3: Algorithms characteristics

Methods	Auto-Hyperparameter	Scalable	Explainable
OC-NN	Yes[1]	No	No
OC-SVM	Yes[1]	No	No
LOF	Yes[1]	No	No
IForest	No	No	No
PA-I	No	No	No
Autoencoder	No	yes	No
EADMNC	No	Yes	Yes
LSHAD	Yes	Yes	No

output; the property of automatic tuning their hyperparameters, in order to free the users from the task of manually adjusting hyperparameters, an operation that will need the assistance of an expert in machine learning.

4.4. Experimentation

This section presents the AD results of the selected unsupervised algorithms. We provide an analysis of the results obtained for the IoT-23 dataset and a comparative evaluation between processing time and AD performance methods.

4.4.1. AD Performance

As mentioned in Section 4.3, a variety of unsupervised learning state-of-the-art methods was selected to evaluate and provide a comparative analysis of their results for IoT AD problems using the IoT-23 dataset as a benchmark.

We employed the Apache-Spark version with the automatic hyperparameter tuning mechanism[2] for the **LSHAD** method, developed by Meira et al. [135]. The following methods included in the scikit-learn Python machine learning library[3] were used:

[2]https://github.com/eirasf/lsh-anomaly-detection
[3]https://scikit-learn.org/stable/index.html

- The **OC-SVM** with the radial basis function kernel;
- The **OC-NN** using distances from the first neighbour;
- **LOF** using the *auto* hyperparameter which will attempt to decide the most appropriate algorithm to compute the nearest neighbors and the hyperparameter *novelty* enabled;
- **IForest** using 500 estimators in the ensemble.

Also, we used the **Autoencoder** implementation in Python from Elephas framework[4] an extension of Keras, which allows to run distributed deep learning models at scale with Spark, employing a bottleneck architecture of 3 layers with 50 neurons in the first and last layer, five neurons in the middle layer and a hyperbolic tangent activation function. For **PA-I** method it was used the built-in MatLab implementation from Martínez-Rego et al. [126]. Finally, it was used the **EADMNC** method from Botana et al. [24] implemented in Scala Apache-Spark[5].

Cross-validation with five folds was performed in our experiments, using a MacBook-Pro laptop with 8GB of RAM with a 2.9GHz Intel Dual-Core i5 processor. We used the AUC metric to measure the performance of each model. We opted for this metric since some subsets are unbalanced, and AUC abstracts such problems by giving equal importance to sensitivity and specificity. Also, this metric is mainly used in the literature when it comes to evaluating or comparing AD methods. It measures the quality of the model's predictions regardless of the classification threshold chosen.

Table 4.4 shows the AUC results of each algorithm in each subset of the IoT23 dataset. We can observe that the **OC-NN** has the best detection performance with an average AUC of 99% in all IoT-23 subsets. This high score allows us to conclude that the Minkowski distance (distance metric used by default in the algorithm) between data points in the feature space can distinguish normal and malware activities in each scenario. Although this is a very high detection score, we can observe that other methods have also obtained relevant results. Thus, **LSHAD** and **EADMNC**, rank as the second and third methods in performance (above 97%),

[4]http://maxpumperla.com/elephas/
[5]https://bit.ly/2YzMGrd

Table 4.4: AUC% Results of selected algorithms for IoT-23 Dataset

ID Dataset	LSHAD	OC-SVM	IForest	Autoencoder	OC-NN	PA-I	EADMNC	LOF
1	**97.07**	88.00	88.89	65.46	95.00	99.13	74.60	87.10
3	99.13	68.34	85.26	82.35	97.49	**99.96**	95.07	92.73
7	99.98	99.58	91.90	73.00	**100**	**100**	99.82	99.14
8	99.97	99.95	99.31	99.99	99.90	**100**	99.98	99.99
9	81.69	54.76	62.18	86.98	98.32	**100**	97.72	98.28
17	97.47	62.50	70.52	91.69	99.55	**99.98**	98.18	82.90
20	**99.96**	99.92	91.42	99.82	**99.96**	81.67	99.95	95.57
21	**99.97**	99.95	85.55	99.94	99.96	80.00	99.94	96.61
33	99.45	93.67	76.62	69.64	**99.79**	58.61	99.58	97.24
34	99.23	99.05	73.69	81.73	98.91	**99.92**	99.73	80.14
35	99.88	**99.96**	73.41	52.82	99.95	99.84	99.87	99.75
36	99.50	99.94	79.71	99.75	**100**	**100**	99.91	94.81
39	99.38	62.99	64.86	**100**	99.41	**100**	85.79	98.88
42	99.88	99.97	**99.98**	99.94	99.97	75.00	99.78	91.82
43	99.94	99.83	63.66	61.30	**99.99**	99.66	99.98	99.64
44	99.56	99.53	97.75	98.85	**100**	92.00	98.60	94.29
48	**99.94**	99.71	99.55	95.36	99.66	99.67	99.84	94.77
49	99.36	93.32	94.31	95.35	99.38	**99.98**	98.85	76.09
52	99.77	99.94	91.69	99.80	99.94	**99.99**	99.84	94.75
60	**99.86**	99.22	81.22	98.94	99.20	**99.84**	99.67	94.50
Avg. AUC	98.55	91.01	83.57	87.63	**99.32**	94.26	97.73	93.45

and it is worth remarking on this, as they can automatically adjust its parameters (the first) and provide an explanation to the results (the second). At the same time, both can manage large-scale datasets. **PA-I** and **LOF** rank as fourth and fifth, with average AUCs above 94% and 93%, respectively. In addition, **PA-I** method has a high number of first ranks (it ranks first in 11 of the 20 subsets). Finally, **OC-SVM**, **IForest** and **Autoencoder** are the three last methods in performance with Average AUCs between 91% and 83% .

Analyzing each scenario, Table 4.4 shows that most methods were able to correctly detect the Mirai attack, being this a simple attack technique applying brute force in telnet with a dictionary combination (scenarios 34, 35, 43, 44, 48, 49, 52, 60). The methods **OC-SVM** and **IForest** presented poor performance detection AUC results for scenario 9 corresponding to Linux, Hajime attack, being this a more sophisticated attack technique, while **PA-I** was able to separate adequately both classes with a 100% AUC, and **OC-NN**, and **EADMNC** methods obtained AUCs above 98%. The **OC-SVM** could not detect the Mushtik and IRC bot attack from scenarios 3 and 39, showing a low AUC score. Overall the algorithms proved to be good solutions to be used for AD problems in IoT devices.

To verify if the overall methods have a significant statistical difference between them, it was performed the Nemenyi post-hoc statistical test [56] with $\alpha = 0.05$ (Figure 4.3). Nemenyi statistical test is described in Appendix A Section A.4.

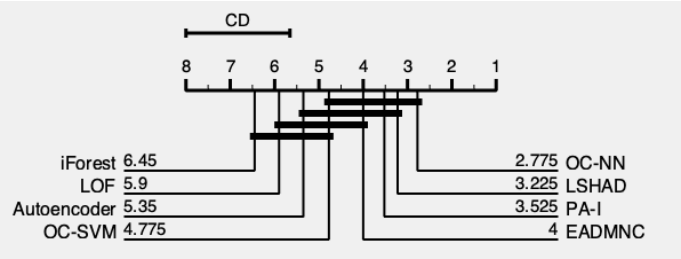


Figure 4.3: Nemenyi statistical test for evaluating AD AUC scores methods

Analysing Figure 4.3, we can see that, regarding their performance, the Nemenyi test separated the algorithms into four groups represented as the thick horizontal lines, being the best group of methods ordered from top to bottom. The methods connected to the same horizontal thick lines do not present significant statistical differences (e.g. the first and best group of methods are: **OC-NN**, **LSHAD**, **PA-I**, **EADMNC** and **OC-SVM**). The methods that appear in the last group and are not present in the first group (**IForest**, **Autoencoder LOF**), show statistical differences with **OC-NN**. On the other hand, **Autoencoder** is in the fourth group, and it is not statistically different compared to the **LSHAD** method in the first group, since both are connected in the second group. However, **LSHAD** and **PA-I** are statistically different compared to **LOF** and **IForest**. The same happens with **EADMNC** that shows the statistical difference with **IForest**. However, it is not statistically different from the other methods since it connects them in the first, second and third groups. We can consider the first group of methods (**OC-NN**, **LSHAD**, **PA-I** and **EADMNC**) as the best solutions for AD problems in IoT devices due to their high-performance results. However, there are other properties of the methods to take into accounts, such as their complexity, scalable characteristics and the ability to handle large datasets, which is the case of **EADMNC** and **LSHAD** since they were developed using the MapReduce approach from Apache Spark.

4.4.2. AD Performance with Distributed Methods

We compared the AD performance of **EADMNC**, **LSHAD** and **Autoencoder** as all of them are scalable and capable of dealing with large data scenarios. Thus, we used all the data from each subset of the IoT-23 dataset and applied five-fold cross-validation. The resources of the Centre of Supercomputing of Galicia (CESGA) were used [70], employing 22 machines with 35GB of RAM and 22 cores each.

Table 4.5: AUC Results of LSHAD, EADMNC and Autoencoder for IoT-23 Datasets

ID DATASET	LSHAD	ADMNC	Autoencoder
1	89.60 ±0.87	**91.95 ± 1.87**	62.58 ± 0.0038
3	**99.53 ± 0.12**	95.45 ± 0.61	96.80 ± 0.0011
7	99.94 ± 0.02	99.68 ± 0.43	**99.71 ± 0.00023**
8	99.97 ±0.05	82.41 ± 0.06	**99.99 ± 6.78e-5**
9	**84.11 ± 2.42**	64.99 ± 15.72	99.89 $\pm 8.96e-9$
17	71.76 ± 21.05	97.22 ± 1.06	**99.99 ± 1.32e-6**
20	**99.96 ± 0.042**	99.94 ± 0.45	99.82 ±0.002
21	**99.97 ± 0.04**	99.94 ± 0.277	99.94 ±0.0004
33	76.42 ± 5.08	**83.31 ± 18.26**	51.81 ± 0.017
34	**99.10 ± 0.37**	99.08 ± 0.52	92.10 ± 0.004
35	98.48 ± 1.38	**99.84 ± 0.06**	95.21 ± 0.017
36	99.77 ± 0.29	99.36 ± 1.21	**99.99 ± 8.99e-8**
39	97.42 ± 0.21	76.98 ± 2.80	**99.99 ± 4.54e-5**
42	99.94 ± 0.04	99.78 ± 0.03	**99.94 ± 0.0003**
43	91.29 ± 3.23	**99.99 ± 0.0005**	59.72 ± 0.038
44	**99.56 ± 0.59**	98.60 ± 0.67	98.85 ± 0.002
48	**99.78 ± 0.19**	99.55 ± 0.78	99.58 $\pm 9.02e-6$
49	**99.52 ± 0.15**	99.37 ± 0.30	99.27 ± 132e-5
52	94.19 ± 3.55	99.61 ± 0.57	**99.99 ± 1.71e-7**
60	99.65 ± 0.17	99.80 ± 0.15	**99.99 ± 7.18e-6**
Avg. AUC	**95.00**	94.34	92.82

The detection performance of all methods, represented in Table 4.5, decreases by about 3% - 4% compared to evaluation subsets samples in Table 4.4 for **LSHAD** and **EADMNC**. Regarding **Autoencoder** obtained much better results compared to the subsets samples, except for the subsets 1, 33 and 43, which showed a small decrease. This decrease occurs because the samples of small subsets used previously do not contain all the information regarding the activity in the network, leading

to a degradation in the performance of methods when dealing with large subsets containing noisy data. Despite this, for all methods, there is no significant decrease in performance, showing very similar performance to the Average AUC in Table 4.5, with **LSHAD** algorithm performing slightly better. All methods can properly detect all different types of attacks, except:

- Kenjiro attack in subsets 17 and 33 where **LSHAD** presented relatively low AUC results, 71 and 76, respectively. Despite **EADMNC** and **Autoencoder** performed well in subset 17 (97,22 and 99.99). The results for subset 33, although better than those of **LSHAD** regarding **EADMNC**, were only 83 AUC, while **Autoencoder** could not detect the attack showing 51.81 AUC;
- **EADMNC** had a drastic decrease of performance when detecting Linux, Hajime attack type in subset 9, compared to the performance achieved when detecting the same attack type in a small sample (Table 4.4);
- Hakai and IRCbot attack types (Subsets 8 and 39) were not properly detected by **EADMNC**, showing AUC results of 82 and 76, respectively;
- **Autoencoder** showed poor performance in detecting Hide and Seek attack from scenario 1, such as Mirai attack from scenario 43.

Hajime, Hakai and Kenjiro attacks are all more sophisticated variants of the more simple Mirai botnet. Thus, these three variants are not detected as easily by the methods tested as in the case of Mirai. It is also interesting to note that **EADMNC** shows the difficulty in distinguishing normal from abnormal behaviour in IRC protocol where IRC botnet was performed.

4.4.3. Scalability Evaluation

To test the scalability of the methods, in Figure 4.4 we have used the subset 35 from the IoT-23 dataset, in which size was varied. We started with a sample of 100 records and increased it by five times in each iteration. We measured the execution time by the ratio of the process time duration from the first sample with 100 records and the duration process time of a specific sample size for each algorithm, allowing

us to empirically approximate the time complexity of each method since the methods are implemented in different platforms.

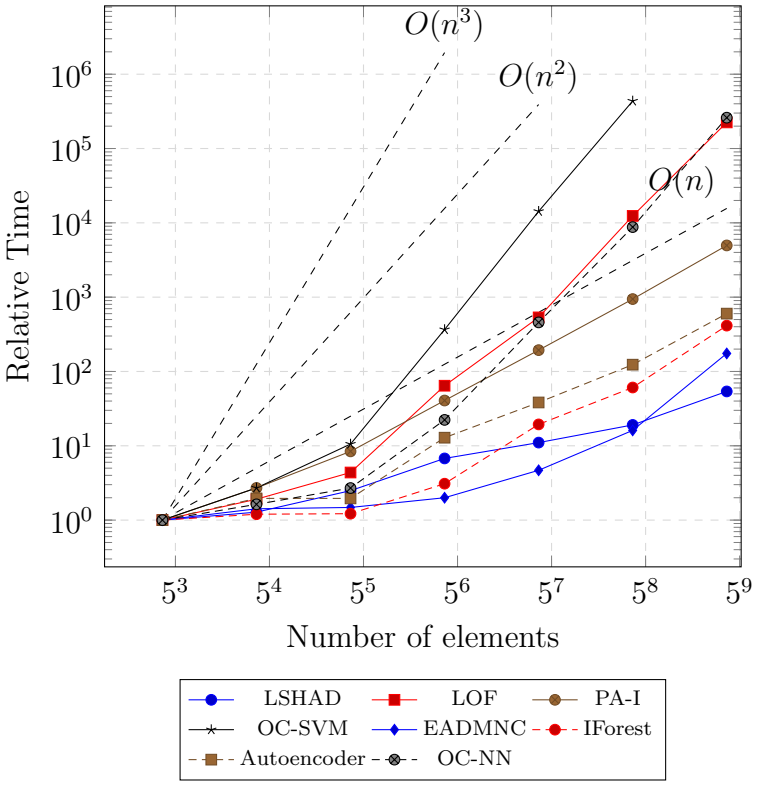


Figure 4.4: Execution time of each algorithm increasing the size samples of the IoT-23 dataset (sub-set 35). Axis are represented using logarithmic scale

Looking at Figure 4.4 we can observe that all the methods can process the data very fast for small sizes of samples (100 to 2.500 records). With more than 2.500 samples, we can verify that **LOF**, **OC-SVM** and **OC-NN** tend to quadratic complexity with **OC-SVM** not being able to handle the largest sample size (1.562.500 records). The **Autoencoder** shows linear-complexity while the remaining methods started with sub-linear complexity and then tended to linear complexity, except for **LSHAD** which maintained a sub-linear complexity. It is also interesting to observe the behaviour of **EADMNC**. It was the most scalable method, slowly increasing the processing time until **LSHAD** overtakes it, the method with the lowest complexity of all the suite of studied methods when handling the largest amount of data with a sample of 1.562.500 records, thus making it the best method for dealing with

huge datasets.

At this point, we identified **OC-NN** as the best algorithm to detect anomalies and **LSHAD** as the best scalable algorithm. However, there is another characteristic of the methods worth taking into account in cybersecurity datasets, and that is the ability to offer explanations of the underlying process of anomaly identification, one of the advantages of the **EADMNC** algorithm.

4.4.4. Explaining Anomalies

In order to carry out an in-depth analysis of the AD results, we have explored the explainability characteristic of the EADMNC algorithm. The use of regression trees allows this method to justify the flagging of elements as anomalies, thus identifying the features with the most significant impact when an attack (anomaly) is detected. In this subsection, we present and discuss some interesting examples within the trees generated for several subsets of the IoT-23 dataset.

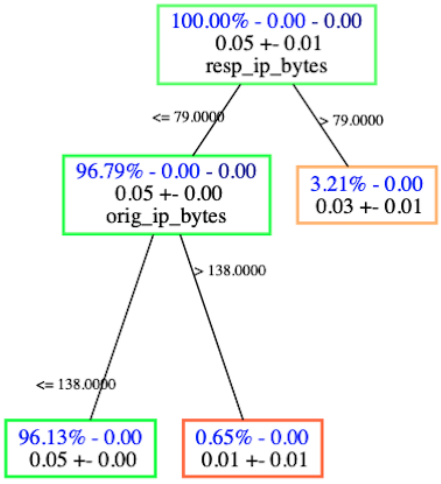


Figure 4.5: Explanatory tree after pruning using subset 20 from IoT-23 dataset.

Reading Figure 4.5 from left to right, each node shows: the proportion of elements that it represents regarding the full subset (shown in blue), overall variance of the anomaly estimators of elements in the node (shown in blue), the weighted estimator variance w.r.t children nodes (shown in dark blue)-first line- and mean and standard deviation for the subset of estimators-second line-. The coloured boxes

range from green to yellow and orange to red to indicate the likelihood level of an observation being considered as normal activity or as an anomaly, respectively. The redder the box, the more likely the observation is to be an anomaly, and the greener the more likely the observation is to be normal activity. Observations with no box are considered neutral.

Figure 4.5 offers a beautifully simple but effective pruned tree for the detection of attacks by the Torii botnet. This tree is not just another Mirai variant, being much more sophisticated than its predecessor, such as being able to target a much wider range of devices and being much more stealthy and persistent than other IoT botnets. It is, hence, somewhat surprising that we can get such high detection rates (AUC is 99,95%) with such a simple tree. At the root of the tree, we have the attribute *resp_ip_bytes* that represents the number of IP level bytes that the responder sent. This feature follows the rule of thumb that, generally, less traffic corresponds to more benign activities. In fact, if the number of bytes is less than or equal to 79, traffic is overwhelmingly considered benign with a rate of 1 to 148. An additional check, this time for traffic generated by the originator, further filters anomalies. Concretely, only if *orig_ip_bytes* is above 138 bytes traffic is labelled as malicious. Going back to the root node again, anything above 79 bytes is deemed malicious. This tree is, as well, the perfect exemplification of the principle of *more is worse* in network security, showing how malicious traffic is always represented by terminal nodes to the right of benign ones.

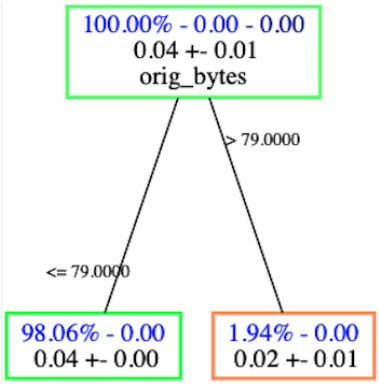


Figure 4.6: Explanatory tree after pruning using subset 42 from IoT-23 dataset.

Something similar happens with the pruned tree of subset 42, Figure 4.6, in which we see again that more data exchanges, in this case, orig_bytes representing the

number of payload bytes sent by the originator perfectly divide between malicious and benign traffic. Curiously enough, with a threshold of 79 bytes, like in the previous example, and similarly linking lesser values to benign and higher values to malicious traffic.

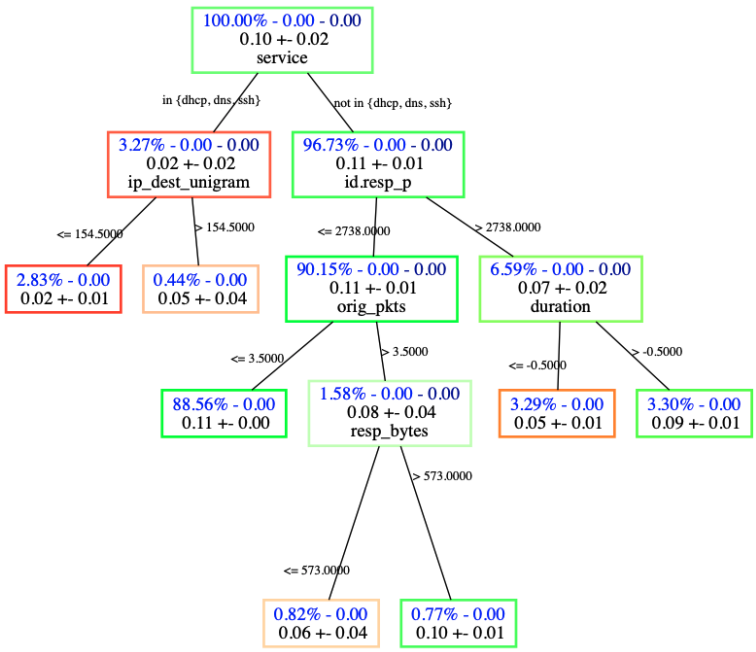


Figure 4.7: Explanatory tree after pruning using subset 3 from IoT-23 dataset.

A very insightful tree is the one generated for subset 3 of the IoT-23 dataset and shown in Figure 4.7. This attack corresponds to the Muhstick botnet. This is a purportedly Chinese[6] malware that has been evolving a great deal and targeting IoT and cloud servers since at least 2018, mining cryptocurrencies and perpetrating other nefarious activities by mainly exploiting web-based vulnerabilities and implementing command and control over IRC channels. The detection strategy exemplified by the tree is interesting on several levels. A very human-like approach is dividing the protocols in a set of safe and unsafe regarding the threat at hand, and that is precisely what the tree does at its root, deeming traffic over protocols such as DHCP, DNS, and SSH very likely malicious in this context. Of course, this only is reasonable over this concrete set of data and will not generalise well, as blank classifications

[6]Check https://www.bleepingcomputer.com/news/security/chinese-linked-muhstik-botnet-targets-oracle-weblogic-drupal/

such as these will likely lead to too many false positives. It is interesting to note the further use of ip_dest_unigram to refine the degree of normalcy. In this case, the use of a threshold of 154.4 indicates that IP ranges starting with values below 155 are considered more conducive to malicious traffic. A similar rule of *less is best* occurs in the subtree to the left, where we see that traffic associated with responder ports below 2739 is considered mostly benign, mainly if the number of packets sent by the originator is low. Alternatively, the other subtree to the right says that if the duration of the connection is known (and hence not -1, which codes unknown/missing values), then the associated traffic is likely benign. All in all, a very interesting way to characterise this particular kind of malicious traffic providing valuable information for system administrators in order to create prevention actions for any type of attack on IoT networks.

4.4.5. Scalability vs AD Performance

An important question is whether it is possible to find a good balance between AD performance and scalability. To evaluate this trade-off it was applied the Pareto Optimization method [187]. In multi-objective optimization, the Pareto front is the border between the region of feasible points (not strictly dominated by any other), for which all constraints are satisfied, and the region of unfeasible points(dominated by others). The Pareto-optimal set is the set of criteria for which no other criterion has both a higher performance detection and higher scalability, hence the members of the Pareto-optimal set are said to be non-dominated [73].

In Figure 4.8 all the algorithms used in our study are represented in order to maximize the average AUC metric of all IoT-23 subsets (X axis) and minimize the processing time (Y axis). To compute the time complexity, we used:

- The number of samples of the largest subset n that each algorithm could handle;
- The time t each algorithm took to process a particular subset.

Thus, it was applied $\frac{log(t)}{log(n)}$. Observing Figure 4.8 we can see that **OC-NN** and **LSHAD** are in the Pareto-Front. **LSHAD** dominate all methods regarding time

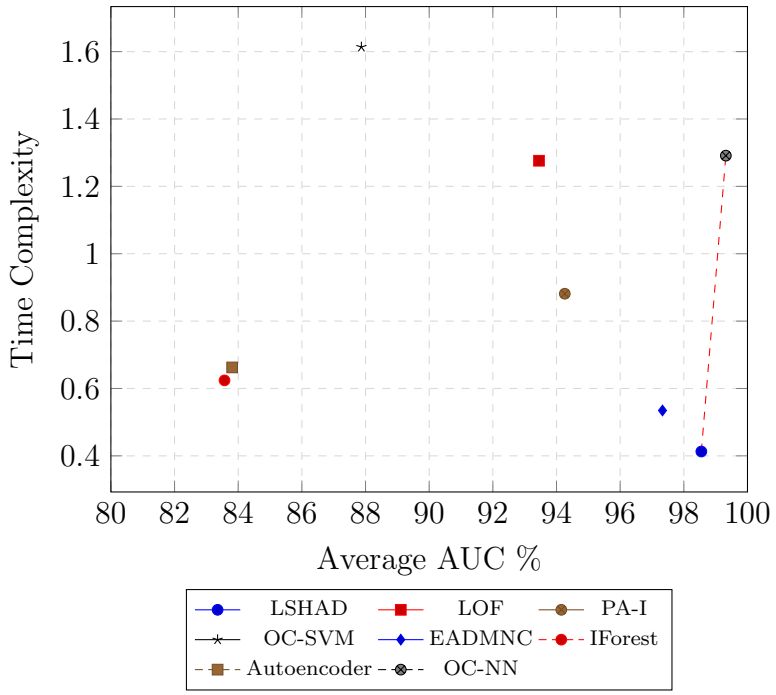


Figure 4.8: Pareto front of a multi-objective optimization problem based on the mean performance AD of all IoT-23 subsets (higher is better) vs time complexity (smaller is better)

complexity. On the other hand, **OC-NN** dominate all methods regarding performance detection. **EADMNC** is not in the Pareto-front since it is slightly dominated by **LSHAD** concerning performance detection. However, it is the only method capable of explaining its results, a crucial characteristic to understand how the IoT attacks are performed. These algorithms do not show a significant difference in AD performance. Therefore, by sacrificing slight detection performance, we have two methods (**LSHAD** and **EADMNC**) with valuable functionalities to tackle big data challenges, such as computing large amounts of data and achieving explainability/transparency in data results.

4.5. Conclusion

In this chapter has evaluated a set of state-of-the-art unsupervised learning methods mainly used for AD problems, identifying their solid points and suitability and providing recommendations when dealing with IoT datasets. The evaluation process measures each algorithm's AD and time performance using the recent IoT-23 dataset. We have also explored some functionalities of the methods that can be relevant in the application area, such as the ability to handle large datasets, the ability to autotune hyperparameters and the ability to provide explanations about the detected anomalies.

Detection performance results showed that **OC-NN** is the best method, with a very high 99 average AUC score for a random sample of the selected IoT-23 subsets. However, a Nemenyi statistical test did not show a significant statistical difference between other methods such as **EADMNC**, **LSHAD**, **OC-SVM** and **PA-I**. Although **OC-NN** was also able to detect anomalies when computational resources are scarce, this method will likely not be the best option to use in an IDS setting since it presents quadratic complexity in the processing time performance evaluation. On the other hand, **LSHAD**, **Autoencoder** and **EADMNC** are promising solutions in an Anomaly or Hybrid-based IDS setting, since all showed to be fast and scalable, presenting sub-linear complexity, and maintaining an adequate detection performance, not significantly different statistically from that of **OC-NN**. Other aspects to take into account are that **LSHAD** has an automatic hyperparameter tuning mechanism that relieves users from any costly tuning steps and that **EADMNC** automatically delivers explanations for the anomalies detected. Furthermore, both methods are scalable and therefore sustainable when compared to their competitors. All methods could handle the full subsets of the IoT-23 dataset since they were developed using the MapReduce approach from Apache Spark. Results showed that detection performance slightly decreased when using all data, and both obtained similar performance.

Chapter 5

Data-Driven PdM Framework for Railway Systems

In this chapter, we confront another of the open challenges in anomaly detection models: working with streaming data. In this respect, we present here a data-driven approach for detecting anomalies in a railway system using a combination of two unsupervised methods. This framework is designed as a data stream model, allowing it to handle streaming data in real-time and also promoting sustainability. In contrast to the static data-based methods presented in previous chapters, this model has limited memory and incorporates a forgetting mechanism and incremental learning, resulting in lower processing times and lower computational resource usage, thus contributing to a more green approach to the anomaly detection process. The work presented in this chapter has been accepted for publication in the Intelligent Data Analysis journal [136]. This approach is well-suited for detecting anomalies in complex, dynamic systems such as in the case of train systems, where data streams are constantly changing and traditional methods may be inadequate. By leveraging the strengths of multiple unsupervised techniques, this model is able to accurately identify unusual events and behaviours, providing valuable insights for the system's maintenance and optimization. Overall, this chapter will provide a detailed overview of the design and performance of this data-driven approach, highlighting its potential for use in a variety of streaming data applications.

PdM) is a method that uses real-time analytic tools to assess collected data from

various parts of one industrial machine [202]. The goal is to detect malfunctions as quickly as possible and fix them before they lead to catastrophic failure. Anomaly detection lies at the core of PdM, with the primary focus on finding anomalies in the working components of machines at early stages and alerting supervisors to carry out maintenance activities [100].

This work describes a data-driven predictive maintenance system to detect anomalies on an APU installed on trains of Metro of Porto. The goal is to identify as early as possible potential failures and notify the maintenance team of an anomaly (undetectable with traditional maintenance criteria), avoiding the inconvenience of removing a train from the operation and saving time and money for the company.

The data is collected from the APU using a set of analogic sensors and reading directly from the APU control system some digital signals that control the state of the APU. We receive the data in regular time intervals, and the learning process extracts information in near real-time to build a predictive model. The model can send an alarm to the maintenance teams, allowing timely intervention on the train.

In this work, we propose an online predictive model capable of dealing with incoming stream data with adaptive learning properties. Since the data incoming from the sensors is endless and received as a continuous flow, we choose to deepen the data stream mining topic, where the methods' computational resources are limited (memory, computational power, processing time). These methods are based on incremental learning as data is induced incrementally and contemplate a forgetting mechanism to deal with limited memory. They differ from batch learning models such as Deep Neural Networks, which are static, computational power is usually a must to get the best fitting in data, and the learning process is performed offline.

Furthermore, we followed a semi-supervised learning approach since we did not know when train failures occurred at the beginning of the project.Therefore, we have combined two methods, the *HS-Trees* algorithm for one-class AD in evolving streams [182] and an adaptation of the *K-Nearest Neighbour* [71, 186] capable of doing one-class classification in streaming data.

The main idea of our proposal is to use *HS-Trees* as the primary anomaly detector method to filter the incoming data. *HS-Trees* sends the observations detected as anomalies to the *One-Class K-Nearest Neighbour* method to reduce false positives.

Our model presented high-performance results, detecting most of the catastrophic failures and producing fewer false positives compared to the *HS-Trees* method.

The chapter is organized as follows: we provide an overview of the related work in the context of AD in Section 5.1. Section 5.2 describes the data used, the problem definition and the detailed description of our proposal. Section 5.3 presents the AD results of our model. Finally, Section 5.4 points out the main conclusions.

5.1. Related Work

Using sensors to monitor industrial equipment combined with the emergence of high-speed networks like 5G and computational systems allowed the development and adaptation of machine learning techniques to AD and predictive maintenance. In this section, we will present some studies regarding these two topics.

Maintenance in industrial equipment and repair procedures are typically responsive to a not-predicted issue. Since malfunctions in equipment affect the safety, availability, and environment, the authors in [109] proposed a real-time monitor to schedule monitoring tasks. These tasks obtain sensor information, measure the state and condition of several components, and determine when the most appropriate moment is to apply a PdM action on the equipment. The PdM topic has been attracting growing interest over the last years with several proposals exploring different machine learning methods for predictive maintenance or AD [109, 162, 114, 157, 165, 123, 113, 101, 45, 29, 17, 169, 99]. More recently, a survey proposed by [53] analyses all the related work regarding the usage of machine learning techniques for predictive maintenance on the railway industry.

Industrial equipment often lacks sufficient and diverse anomalous data to build a binary classification system. Thus many of the predictive maintenance models rely on unsupervised AD algorithms, which are responsible for determining whether an observation of the sensor deviates from the normal state of the equipment [68, 127]. Detecting the presence of anomalies in real-time provides valuable insights and knowledge about the equipment to make a rigorous assessment of possible maintenance interventions. There are several works in the literature related to the topic of predictive maintenance in railway systems, and they can be organized into super-

vised or unsupervised learning approaches:

Supervised Learning

Rabatel et al. [162] explored the application of sequential patterns to correctly identify normal and abnormal data generated by a set of sensors installed in three key train components.

Li et al. [114] proposed a five-step predictive maintenance framework. The first step is the feature extraction of the dataset containing information about bearings on the train. The second step is reducing dimensional space using the Principal Component Analysis. The model adopted was the Support Vector Machine. Finally, a confidence level for alarm prediction was defined, and a rule simplification divides the feature space into non-overlapping small grids.

In terms of predicting failures on door trains, Manco et al. [123] developed an application to predict and explain door failures using an outlier detection method. Pereira et al. [157] developed a failure detection system for classifying irregular open/close cycles within trains based on the difference between the inlet and outlet pressure in specific intervals of the cycle. More recently, Ribeiro et al. [165] explored data-driven PdM based on anomaly and novelty detection implemented to predict failure in the automatic door system. The results showed that a low-pass filter could significantly reduce the number of false alarms.

Fumeo et al. [76] described a condition-based maintenance algorithm that explores the online support vector regression algorithm to predict the remaining useful life of the railway vehicle. In particular, the authors aim to detect failures on the axle bearings as soon as possible.

Wan-Jui Lee [113] used the Linear Regression model to describe two different compressor operations (idle and running time). The authors used logistic functions to define the boundaries of the two classes or compressor operations modes. The system is used for air leakage detection by AD in a train's braking pipes. They used a density-based clustering method with a dynamic threshold to distinguish anomalies.

Bukhsh et al. [29] explored the usage of tree-based models like Random Forest, Decision trees, or XGBoost to predict the status of railway switches. Additionally,

the authors explored the Local Interpretable Model-Agnostic Explanations (LIME) to explain the possible reasons for the malfunction. Kalathas and Papoutsidakis [99] applied two well-known classification algorithms, the J48 and M5P, to monitor the health state of traction and braking subsystems of the Greek Railway. Adopting tree algorithms helps the maintenance teams understand the reason for the malfunction.

Kang et al. [101] described a system that uses a Bayesian statistical learning model to represent the expected behaviour of the train in terms of speed. The study's main objective was to capture changes and anomalies in the trains' speed to detect some malfunctions as early as possible.

Barros et al. [17] proposed adopting a rule-based system to detect anomalies on a train compressor unit. This system monitored several analogical and digital variables and then used a low pass filter to smooth the analogical signals and count the number of peaks in a time window. The rules were designed based on the maintenance teams' expertise to define the compressor units' normal state.

Unsupervised Learning

Salierno et al. [169] proposed architecture for predictive maintenance on the railway domain. The proposed architecture is to predict failures in the interlocking railway system of the Italian Railway. The authors adopted a Long Short Term Memory model to capture abnormal patterns of the interlocking system.

Davari et al. [54] describe a sparse autoencoder network for PdM on a metro railway domain. The proposed autoencoder is designed to predict failures on the air compressor subsystem to remove the train from circulation safely.

Chen et al. [45] presented a predictive system for the compressor air unit. The authors used a recurrent neural network using Long Short-Term Memory architecture for failure prediction. The authors compared their method with the random forest method, and the results showed that the neural network proposal was more stable when compared with the Random Forest.

All the described related works (summarized in table 5.1) rely on identifying the normal state of the system/component, considering as possible anomalies the observations that do not have the same familiar patterns. Different machine learning models or techniques were applied depending on the context and characteristics of

the equipment.

Our approach differs from the state of the art because it relies on machine learning techniques to identify abnormal patterns correctly. The supervised approaches presented in this section do not work in real-time because we do not know the ground truth. When we compare with unsupervised learning approaches, where some authors look to the autoencoders' higher values of reconstruction error to signal an anomaly, we suffer from a false positive alarms problem. Our method relies on a semi-supervised learning algorithm, HS-Tree, which learns a single class and classifies all the other classes as an anomaly.

If the output of one observation is positive for an anomaly, we use a kNN algorithm to see if the observation is distant from known normal observations of the air compressor unit. The ablation study in this manuscript shows a significant improvement in the evaluation metrics.

Table 5.1: Related Work Comparison

Ref.	Target System	Model	Explainable Model	Evaluation Metric
[162]	Train	Sequence Patterns	No	Recall & Precision
[123]	Doors	Outlier	No	AUC
[157]	Doors	LPF	No	False Alarm Rate & Impostor Pass Rate
[165]	Doors	LPF	No	Reduced False Alarm Rate & Reduced Impostor Pass Rate & Detection Error
[113]	Air Compressor	Linear Regression	No	RMSE & Confusion Matrix
[101]	Train	Bayesian Model	No	Error
[45]	Air Compressor	RNN / LSTM	No	F-Measure & AUC & Accuracy & Recall & Precision
[17]	Air Compressor	Rule-based	No	F-measure
[29]	Railway switches	XGB / RF / DT	Yes	Accuracy & F-Measure & Kappa
[99]	Railway switches	J48 / M5P	Yes	Recall & Precision & Matthews Correlation Coefficient
[114]	Bearings	SVM	No	Accuracy & Recall & Precision
[76]	Axle bearings	SVR	No	Mean Absolute Percentage Error
[169]	Interlocking	LSTM	No	Error
[54]	Air Compressor	SAE	No	F-measure

5.2. Methodology

5.2.1. Problem Definition

The APU is part of a compressed air system, which produces pressurizing air from an electric motor. The electrical current consumed by the motor is converted into kinetic energy. The compressed air system is a crucial component of the train and delivers essential pressurized air to several clients like pneumatic suspension, oil injection on the rail to reduce the friction and noise on the curves, and injection of sand to gain traction rails, and finally, connect other trains. Applying predictive maintenance here is essential to predict the equipment failure before it happens, decreasing costs and optimizing the service.

5.2.2. Trains Data

The data acquisition system collects information from several analogical sensors and digital signals generated by the APU control system. Based on the failure history of the train fleet, it is possible to identify the critical components of the system that generate the majority of the failures. These critical components are: (i) electrical valve; (ii) pressure valve; (iii) oil leaks; (iv) electrical motor; (v) pressure switches; and (vi) drying towers. The sensors and places to install them were strategically defined, considering the output of the failure history study. Figure 5.1 shows an overview of the train system.

The data acquisition system communicates with a cloud server that receives the data from the sensors with 1 Hz of sampling frequency. The system stores the data collected from the sensors and respective timestamps to a data logger file, and every five minutes, the file is sent to the server using the TCP/IP protocol application.

The considered analogical sensors were the following.

- TP2 - Measures the pressure on the compressor.
- TP3 - Measures the pressure generated at the pneumatic panel.
- H1 - This valve is activated when the pressure read by the pressure switch of

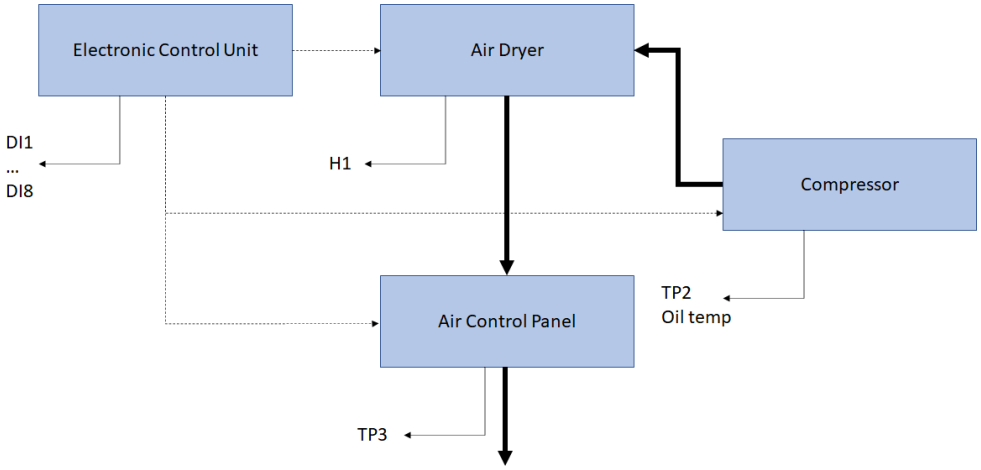


Figure 5.1: Train System: dark arrows represent the pneumatic system, dashed arrows the control system and the thin black arrows the sensors

the command is above the operating pressure of 10.2 bar.

- DV pressure - Measures the pressure exerted due to pressure drop generated air dryers towers, and when it is equal to zero, the compressor is working under load.
- Motor Current - Measures the current of one phase of the three-phase motor, which should present values close to 0 A when the compressor turns off, close to 4 A when the compressor is working offloaded and close to 7 A when the compressor is working under load. When the compressor starts to work, the motor current presents values close to 9 A.
- Oil Temperature - Measures the temperature of the oil present on the compressor
- Flowmeter - Measures the airflow that leaves the APU for Reservoirs

The considered digital sensors were the following.

- COMP - The electrical signal of the air intake valve on the compressor. It is active when there is no admission of air on the compressor, meaning that the compressor turns off or working offloaded.

- DV electric - the electrical signal that commands the compressor outlet valve. When it is active, it means that the compressor is working under load; when it is not active, it means that the compressor is off or offloaded.

- TOWERS - Defines which tower is drying the air and which tower is draining the humidity removed from the air. When it is not active, it means that tower one is working; when it is active, it means that tower two is working.

- MPG - Is responsible for activating the intake valve to start the compressor under load when the pressure in the APU is below 8.2 bar. Consequently, it will activate the sensor COMP, which assumes the same behaviour as MPG sensor.

- LPS - Is activated when the pressure is lower than 7 bars.

- Oil Level - Detects the oil level on the compressor and is active (equal to one) when the oil is below the expected values.

5.2.3. Proposed model

For our proposal, we only considered the analogical sensors data arriving in the stream recorded at each second. Figure 5.2 illustrates our anomaly detection model for predicting catastrophic failures. The algorithms employed to build the proposed model and detect the train system's catastrophic failures, namely HS-Trees and the OCKNN are described in Appendix A Section A.2.

Before feeding *HS-Trees* algorithm, we aggregated the data in minutes through the timestamp feature. This operation extracted each sensor's mean, median, standard deviation, and variance. Our experiences found that the information extracted by each minute was sufficient to prevent the *HS-Trees* algorithm from losing performance, thus optimizing the data processing time as it computes fewer records.

After running several experiments with *HS-Trees*, we noticed that this method was generating a large number of false positives since only 4 % of data was reported as a failure, while *HS-Trees* was detecting around 25 % of failures. To tackle this problem, we adopted the *OCKNN* algorithm to deal with data arriving continuously.

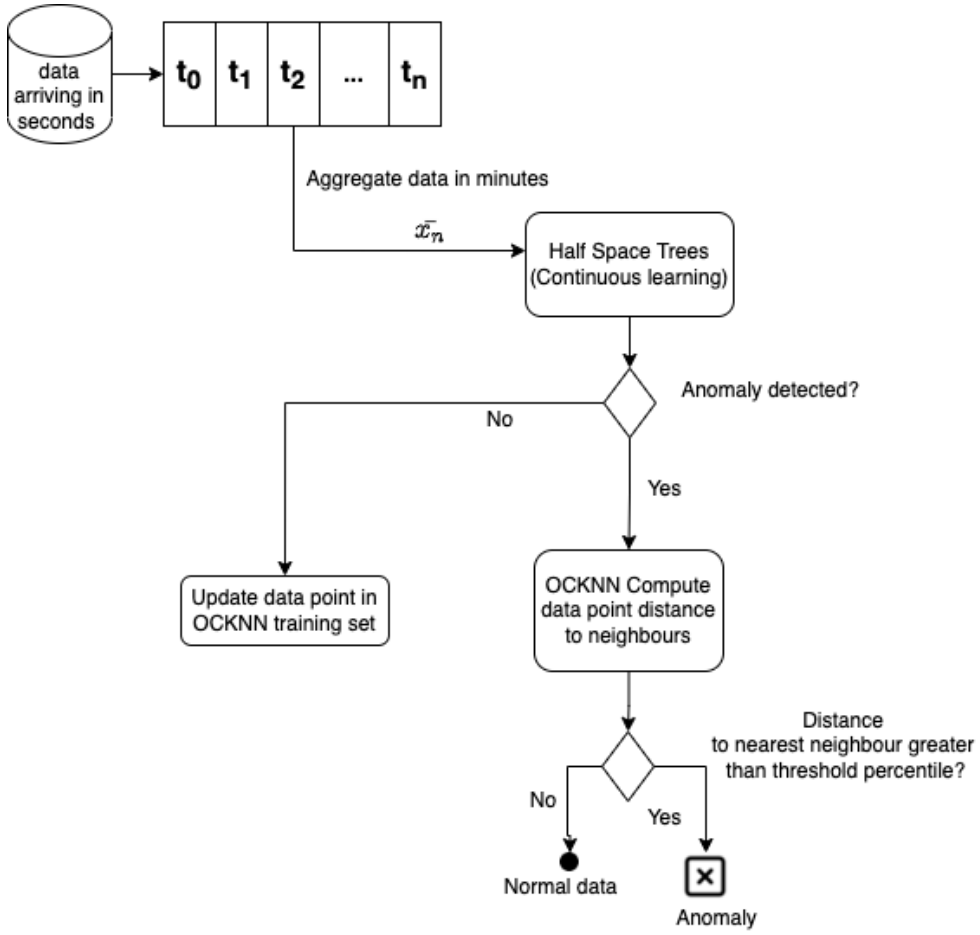


Figure 5.2: Proposed methodology

The idea is that the *OCKNN* evaluates each anomaly detected observation from *HS-Trees* to check if it was detected correctly.

Data points are updated in the *OCKNN* training set if *HS-Trees* inferred these points as normal data. The update process considers the maximum and minimum distances to neighbour's values captured during the stream. Distant normal points to its neighbours are added to the training set while neighbour points with the lowest distance are removed. This update mechanism showed high-performance results as stacking points with high distances present high sensitivity when detecting anomalous data.

In the case of *HS-Trees* inferred points as anomalous, the *OCKNN* method cal-

culates the distance from each arriving data point to its closest neighbour to verify whether they are at an abnormal distance. To better understand our model, it is presented in Algorithm 4 the pseudo-code implementation.

Before starting the data stream, initial parameters and data structures were defined: A set of initial training data with 1400 records for the *OCKNN* method, which represents a whole day stack (24h) from a period that we know there was no anomaly in the train system; a K number of neighbours to compute distances to the arriving data points from which we only used 1 neighbour; a $dist_{Max}$ variable to record the maximum distance to its neighbour set as 0 (lowest value to be replaced in the first iteration) and a $dist_{Min}$ variable to record the minimum distance to its neighbour set as 9999 (a high value to be replaced in the first iteration). The output of our model returns a list of TimeStamp values that indicate when an anomaly has occurred in the APU train system.

The data stream cycle starts in Line 2, where variable x is assigned to each arrival data point. The algorithm starts by computing the distance to the nearest neighbour, in line 8. Then it computes the distance percentile, which is used as a threshold to identify anomalies (line 5), employing a Zscore table value of 2.326 representing percentile 99%, which means detecting 1% of observations with high distance to neighbours. Then, the *HS-Trees* method starts by inferring the arrival data point (line 6), checking if it corresponds to an anomaly (line 7). If the data point is considered anomalous, it validates if the distance to its neighbour is greater than the threshold percentile (line 8). If confirmed, the TimeStamp value is stored in the list S (line 9). Also, the algorithm records the $dist_{Max}$ assigning its value to the nearest neighbour distance of the current data point if that distance is greater than the previous $dist_{Max}$ value.

In case *HS-Trees* infers the arrival data point as normal behaviour (line 12) the algorithm assigns $dist_{Min}$ and $dist_{MinIndex}$ as the current data point distance to its neighbour and the neighbour id respectively only if that distance is less than previous $dist_{Min}$ value. The next validation is performed to update the *OCKNN* training set with points identified as a normal activity of the train system by the *HS-Trees* if it matches a certain condition. The data is incremented in the *OCKNN* training set if the current normal data point distance to its neighbour is greater than the third quartile of $dist_{Max}$ value. The algorithm also discards from *OCKNN* training set the

Algorithm 4: Pseudo-code for HS-trees with OCKNN approach.

Input : $D_i \leftarrow$ OCKNN set of initial training points
$D_s \leftarrow$ Data Stream
$dist_{Max} \leftarrow$ Maximum distance to neighbours
$dist_{Min} \leftarrow$ Minimum distance to neighbours
$K \leftarrow$ Number of OCKNN neighbours

Output: $S \leftarrow$ Anomalies TimeStamp

```
1  S ← 0, D_i ← 1400, dist_Max ← 0, dist_Min ← 9999, K ← 1,
     OCKNN.fit(D_i);
2  while D_s continues do
3      Receiving the next streaming point x;
4      dist ← OCKNN.ComputeDistance(x);
5      distPercentile ← distMean + zScore * distSTD;
6      predict ← HSTrees.predict(x);
7      if predict == anomaly then
8          if dist > distPercentile then
9              S.append(x.Timestamp);
           end
10         if dist > dist_Max then
11             dist_Max = dist;
           end
       end
12     if predict == normal then
13         if dist < dist_Min then
14             dist_Min = dist;
15             dist_MinIndex = closestNeighbour.index;
           end
16         if dist > dist_Max * 0.75 then
17             D_i.drop(x[dist_MinIndex]);
18             D_i.append(x);
19             OCKNN.fit(D_i);
20             dist_Max = 0;
21             dist_Min = 9999;
           end
       end
22     HSTrees.partialFit(x);
   end
```

Result: S

nearest neighbour data point with the lowest distance. Then, $dist_{Max}$ and $dist_{Min}$ values are reset. These operations are listed from line 16 to 21. Finally, *HS-Trees* is incrementally fitted for each data point in order to build the mass profile used to estimate anomalies (line 22).

As anomalous events are rare, we define a threshold value representing 1 % of the arriving data with the highest distance values to its nearest neighbour. This threshold parameter value allowed our method to detect most anomalous periods generating few type I and II errors. Figure 5.3 shows the anomalous data points detected by our method in one of the performed experiments. Distance values equal to zero represent data points classified as normal behaviour, while distance values greater than zero are the anomalies detected by our method. The colours represent the real meaning of the data. In red are the data points that correspond to the real anomalies, and in blue, the data points that correspond to the real normal behaviour of the train system.

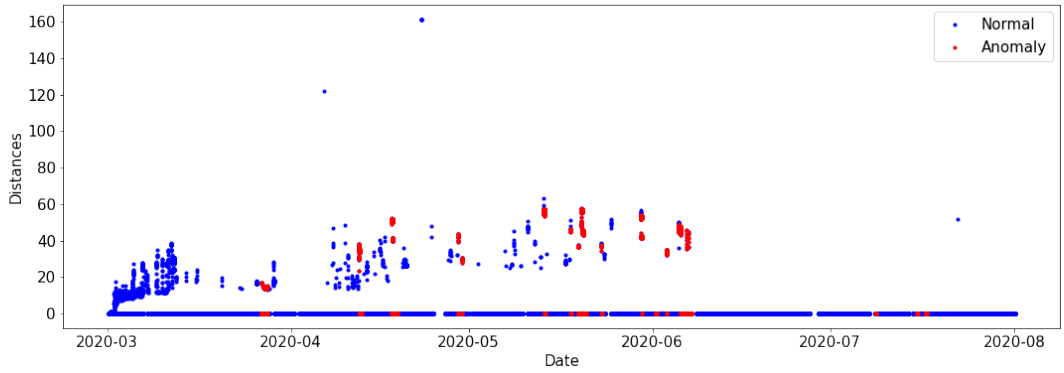


Figure 5.3: Anomalies detected by our method

It can be seen in Figure 5.3 a set of normal values detected as anomalies probably due to the initial fit of the *HS-Trees* model to the data distribution. The model classifies fewer observations as anomalous from mid-March, identifying practically part of all anomalous periods with only a few examples represented by normal activity (false negatives). This model was developed with python, using the implementation of the *HS-Trees* algorithm from the scikit-multiflow[1] library [138] and the imple-

[1]https://scikit-multiflow.github.io/

mentation of the *K Nearest-Neighbour* algorithm from the scikit-learn[2] library [156] adapted to work as one class classification with online data.

5.3. Model Evaluation

In this section, we evaluate our model and report the result of our experiments. We evaluated the model's effectiveness using data from a train in operation in 5 months of 2020, with some catastrophic failures reported during that period. The data contains 21 periods reported as anomalous. Some last a few minutes, others a couple of hours.

5.3.1. Evaluation Procedure

In order to evaluate the performance of our approach, five experiments were carried out with some state-of-the-art anomaly detection algorithms in the context of data streams, using the data from the analogical sensors present in the APU system. Therefore, the mean, median, standard deviation and variance from the DV_pressure, TP2, TP3, H1, Oil_temperature, Motor_current and mode were used. The last feature concerns the status of the train. This feature has three states: in progress, stopped, and under maintenance. Maintenance status data has been discarded as tests are performed on the trains, causing the APU system to generate anomalous values, misleading the model's predictions. It is also important to mention that all data were normalized using the standard window scaling technique, which standardizes features by removing the mean and scaling to unit variance. The mean and standard deviation are computed on a given window frame.

Regarding the algorithms, we tested our approach (**HSTreeOCKNN**) against AD methods for data streams such as: **Half-Space-Trees (HSTrees)** [182], **XStream** [125], **Isolation Forest (IForestASD)** [58] and **ExactStorm** [11].

To assess the models, we verified that the detected anomalies were within the reported anomalous period, as shown in Figure 5.4. If for a given model, there is an overlap in its output to the ground truth (in that anomalous period is detected more

[2]https://scikit-learn.org/stable/

than one anomaly), then all observations from that period are counted as anomalous (True Positive) in the model's output. Note that the results of our methodology were validated by experts at *Metro do Porto*.

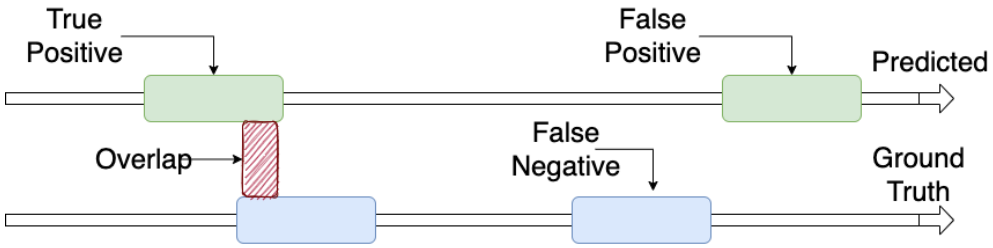


Figure 5.4: Models Validation approach.

We performed several experiments to adjust the hyperparameters reaching the settings in Table 5.2.

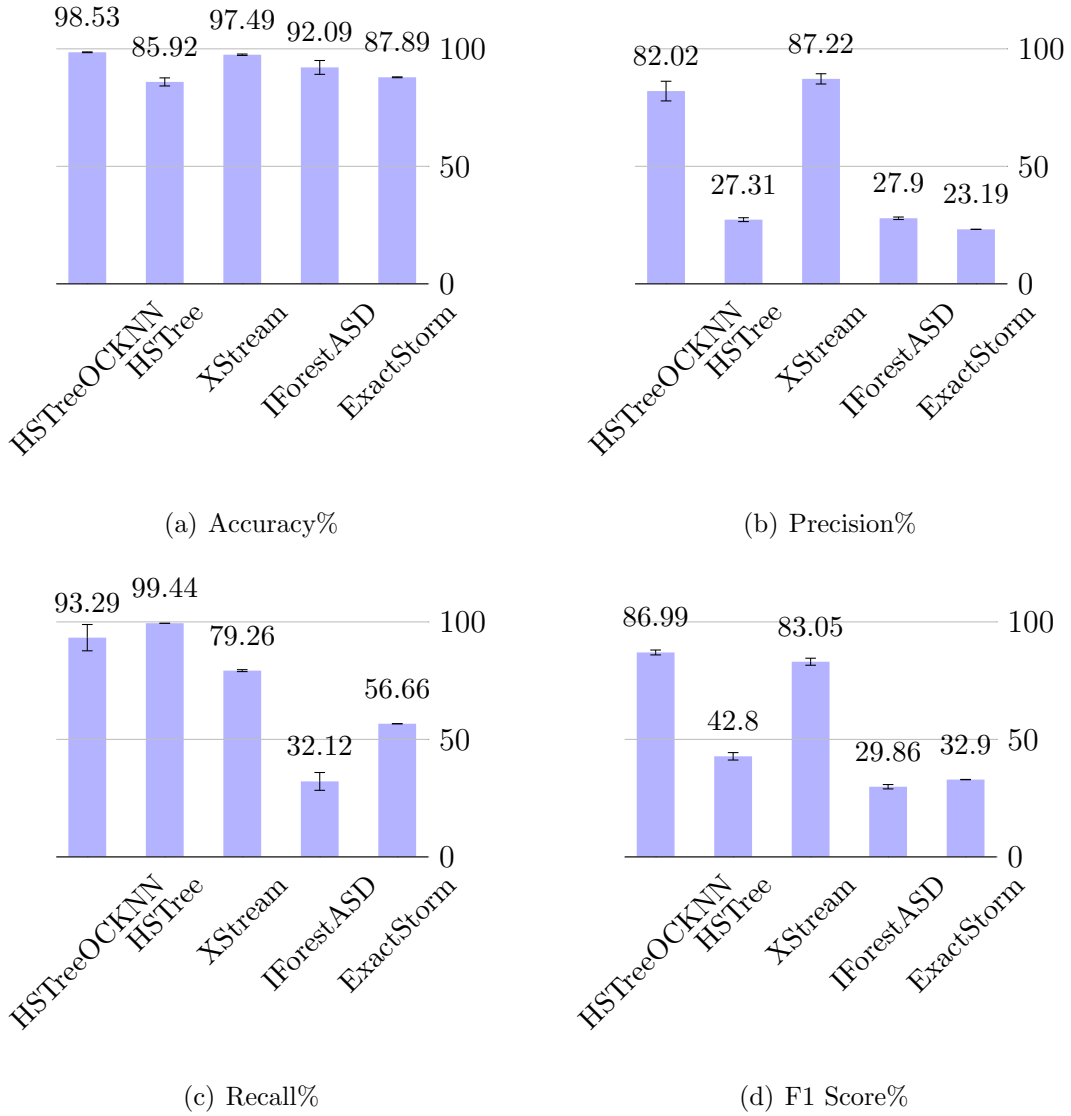
We used the accuracy, Precision, Recall, and F1 metrics for model evaluation, giving the necessary information to analyze the type I and type II errors.

5.3.2. Discussion

First, we start by analyzing the results of the models in Figure 5.5(a) where metric accuracy was used. We can observe that our model was the best, reaching an accuracy of around 98 %, followed by the **XStream** algorithm that achieved a 1% lower accuracy when compared to our approach. **IForestASD** was ranking third with an accuracy of 92 %, while **ExactStorm** and **HSTree** performed worst with an 87 % and 85 % accuracy respectively. A high accuracy value was expected since

Table 5.2: Selected hyperparameters

	HSTreeOCKNN	HSTree	IForestASD	XStream	ExactStorm
N Estimators	15	15	25	15	-
Window Size	400	400	400	400	400
Depth	10	10	-	7	-
size limit	75	75	-	-	-
Max Radious	-	-	-	-	0.15
N Components	-	-	-	20	-



(a) Accuracy%

(b) Precision%

(c) Recall%

(d) F1 Score%

Figure 5.5: Performance results of the methods using the metrics Accuracy (a), Precision (b), Recall (c) and F1 Score (d)

failures were rare, representing only 4 % of all data. However, this information is vague and insufficient to analyze the types of errors generated by the models.

Analyzing the remaining metrics, it can be seen in Figure 5.5(b) the percentage of type I errors generated by the models. In this case, the **XStream** was the best model with 87 % precision, followed by **HSTreeOCKNN** achieving 82 %. As **OCKNN** receives the output of **HS-Trees** to train and validate anomalies, we know

a priori that there would be an increase in performance due to the elimination of false positives by the **OCKNN**. Therefore was expected an increase in the precision metric by our method, which is visible in Figure 5.5(b), representing a 55 % higher value if we only used **HS-Trees** model to tackle this problem. The other models presented a poor performance, with values below 30 % precision, which means that these models misclassified most observations as anomalous.

Analyzing Figure 5.5(c) concerning recall metric, the results presented by the models are much better (except for the **IForestASD**), meaning all methods generated low type II errors. The models should generate less FN than FP since these errors indicate that an anomaly was mistaken for normal activity in the APU train systems. Failure to classify anomalous activity into normal activity will cause the train to run into a catastrophic failure, leading to high repair costs and the sporadic closure of the railway. As it can be seen, **HSTree** could detect almost every anomaly with a high cost of FP (Figure 5.5(b)). **HStreeOCKNN** was placed in second with 93 % of recall, a small cost of generating FN compared to **HStree** in order to get a higher precision value. **XStream** did a decent job with a 79 % recall value. At last, **ExactStorm** with 56 % recall followed by **IForestASD** detecting only a few anomalies.

To analyze the balance between precision and recall metrics, we can observe Figure 5.5(d), which presents the model's performance evaluated by the F1 score metric. Therefore, our approach was the best, achieving 87 % F1 score, followed by **XStream** as it was observed in Figures 5.5(b) and 5.5(c), it had a slightly lower type I error rate than the **HSTreeOCKNN** model, but with a significantly higher type II error rate. **HSTree** shows a poor F1 score due to the high type I error rate, while **IForestASD** and **ExactStorm** presented the worst performance in both precision and recall metrics.

5.4. Conclusions

Predictive Maintenance enables more efficient, longer-term planning for maintenance operations and makes it easier to allocate maintenance resources and define operational maintenance goals. One of the most promising aspects of the railway

industry's transformation is Predictive Maintenance through data collected on the equipment during operation to identify failures in real-time. Therefore, repairs can be adequately planned without unexpectedly taking trains out of service for emergencies or unnecessary routine Maintenance.

This work presents a data-driven predictive maintenance framework for the APU train system of *Metro of Porto*. We used the *HS-Trees* method combined with *OCKNN* to build a predictive model capable of detecting catastrophic anomalies and dealing with streaming data.

Our empirical study shows that the use of *HS-Trees* provided significant performance improvements when used in conjunction with *OCKNN*. The proposed predictive model obtained high AD performance while maintaining fewer false positives and negatives compared to SotA methods. Distances from neighbours are a viable solution to reduce false positives for this problem.

Chapter 6

Anomaly Detection on Natural Language Processing to Improve Predictions on Tourist Preferences

In this chapter, we address a different application field, specifically tourism, and a different type of data, text reviews written by tourists. We explore the use of argumentation-based dialogue models to facilitate the decision-making process in the context of predicting tourist preferences for points of interest. To do this, we study strategies for automatically predicting the ratings that tourists assign to these points of interest based on their reviews. To achieve this goal, we use natural language processing techniques to predict whether a review is positive or negative, and the rating assigned by the user on a scale of 1 to 5. We then apply a range of supervised machine learning methods, including logistic regression, random forests, decision trees, and k-nearest neighbours, to determine whether a tourist likes or dislikes a particular point of interest. However, our main focus in this chapter is on the use of unsupervised techniques to improve the performance of our supervised model in identifying only those tourists who truly like or dislike a particular point of interest. To do this, we utilize a distinctive approach in this field by applying unsupervised techniques for AD problems. The goal is to identify and classify only those tourists who have a strong preference for a particular point of interest, rather than trying to classify all tourists. Overall, this chapter will provide

a detailed analysis of the various methods and approaches we used, with a particular focus on the role of unsupervised techniques in predicting tourist preferences using argumentation-based dialogue models. The presented work was published in the Electronics Journal [134].

Argumentation-based dialogue models are extremely useful in contexts where a group of agents is intended to find solutions for complex decision problems using negotiation and deliberation mechanisms [34, 35, 31]. In addition, they allow human decision-makers to understand the reasons that led to a given decision (enhancing the acceptance of decisions) and to define mechanisms for intelligent explanations [189, 130]. These models receive the decision-maker's preferences as input (for instance, regarding criteria and alternatives), which are typically used to model the agents that represent them [33]. However, obtaining these preferences is not a simple process: first, in the contemporary and highly dynamic world in which we live, it is less and less comfortable for decision-makers to answer questionnaires and, second, it is sometimes difficult to express preferences through questionnaires [32, 36]. To facilitate this task, strategies that aim to automatically identify the users' preferences have been proposed. One of these strategies consists in using ML algorithms and NLP to automatically extract from a text corpus the users' opinions through different strategies such as text wrangling and pre-processing, named entity recognition and sentiment analysis [181, 48]. However, there are many algorithms and strategies that can be applied. Therefore, it is mandatory to develop specific procedures according to the application topic, to achieve the best results.

In this chapter, we studied the problem previously described under the topic of group recommendation systems, more specifically in the context of tourism, in which there has been an increased interest in the development of technologies capable of making recommendations according to the interests of each group member. We assumed as habitual that users/tourists express their opinions regarding POI on social networks (such as TripAdvisor, Facebook, or Booking.com) and we sought to take advantage of this to automatically predict their preferences non-intrusively. For this, we used a public dataset (available in Kaggle) and applied the development lifecycle for intelligent systems using concepts of NLP defined in [188]. More specifically, we developed forecast models using five supervised ML algorithms (Logistic Regression [201], Random Forest [26], Decision Trees [161], K-Nearest Neighbors [72], and

Long/Short-Term Memory [88]), using them both as classification and regression methods. We also applied three unsupervised ML algorithms (One-Class Nearest Neighbor [186], Isolation Forest [117], and Local Outlier Factor [27]) used for AD to improve the supervised ML methods' results. In addition, we used NLP to extract more knowledge from the users' reviews and various libraries of Sentiment Analysis (Vader [93], TextBlob [122] and Flair [4]) to find those that best fit this context.

The rest of the Chapter is organized in the following order: Section 6.1 reviews SotA works in the field of recommendation systems. Section 6.2 describes our methodology. Section 6.3 presents the obtained results. In the last section, some conclusions are put forward.

6.1. Related Work

Several works have been conducted and proposed for the development of recommended systems in the tourism context. Nilashi et al. [143] applied multi-criteria ratings in developing a new method for hotel recommendations in e-tourism platforms. The authors used supervised and unsupervised ML techniques to analyze the customers' online reviews. Cenni and Goethals [41] examined 100 reviews for languages written in English, Dutch, and Italian and analyzed three features, namely the types of speech acts that users used, the specific topics that they evaluated, and the extent to which they up-scaled or down-scaled their evaluative statements. The authors found a general trend towards similarity between the three language user groups under examination.

Valvida et al. [192] propose TripAdvisor as a source of data for sentiment analysis tasks. The authors develop an analysis for studying the matching between users' sentiments and automatic sentiment-detection algorithms. They provide some of the challenges regarding sentiment analysis on TripAdvisor. In [5], the authors present a review focused on the multi-criteria review-based RS, where they explain the user reviews' elements in detail and how these can be integrated into the RS to help develop their criteria to enhance its performance. The authors presented four future trends to support researchers who wish to pursue studies in this field based on the survey.

The work of Kbaier et al. [103] focused on building personalized RS in the tourism field. They proposed a hybrid RS that combines the three best-known recommender methods: CF, CB, and DF. In order to implement these recommender methods, the authors applied different ML algorithms, which were the K-Nearest Neighbors for both CB and CF and the Decision Tree for the DF. They conducted an extensive experimental study based on different evaluation metrics using extracted data from TripAdvisor.

In the work of Logesh et al. [121], they proposed an Activity and Behavior-Induced Personalized RS (ABiPRS) as a hybrid approach to predict persuasive POI recommendations. Their RS is designed to support travelling users by providing a compelling list of POIs as recommendations. As an extension, the authors designed a new group recommendation model to meet the requirements of the group of users by exploiting relationships between them. They also have developed a novel hybridization approach for aggregating recommendations from multiple RSs to improve the effectiveness of recommendations. The authors evaluated their approach on real-time large-scale datasets of Yelp and TripAdvisor.

In [177], the authors provided a fascinating study of users' evaluations of serendipity in urban recommender systems through a survey among 1641 citizens. They studied which characteristics of recommended items contribute to serendipitous experiences and to what extent this increases user satisfaction and conversion. Their results are aligned with findings in other application domains in the sense that there is a strong relation between the relevance and novelty of recommendations and the corresponding experienced serendipity. They found that serendipitous recommendations increase the chance of users following up on these recommendations.

6.2. Methodology

In this section, we describe the methodology in detail. We start by describing the problem that we intend to address. Next, we justify the choice of the dataset, and carry out its analysis, covering preprocessing and feature engineering. Finally, we approach the used computational techniques and describe the tests and results obtained.

6.2.1. Understanding the Problem Statement

The problem we want to overcome is to predict, non-intrusively and with a high level of accuracy, how much a tourist likes/dislikes a given POI. Subsequently, we intend to use the predicted preferences to model intelligent agents that represent tourists in a group recommendation system, who seek to jointly decide (using an argumentation-based dialogue model) and recommend to the group of tourists the set of POIs to visit. For this, we chose to use the reviews that tourists wrote on social media (TripAdvisor) to predict their preferences.

6.2.2. Collecting Dataset

The chosen dataset was selected based on 2 criteria: it needed to be a public dataset and should best represent the context in which this work intends to be applied. Therefore, a dataset available at Kaggle [7] and which is composed of more than 20 thousand hotel reviews extracted from TripAdvisor was selected. The fact that there are already many works on Kaggle's repository that use this dataset allowed us to know beforehand that it would be very difficult to obtain good results, since, for example, for predicting 5 classes, the presented accuracy of the vast majority varies between 30% and 60%.

6.2.3. Analyzing Dataset, Preprocessing, and Feature Engineering

The dataset is composed of the attributes "Review" and "Rating". Table 6.1 shows some examples of the type of records that make up the dataset. The "Rating" is between 1 and 5, where 1 is the worst and 5 is the best possible evaluation.

The dataset consisted of 20,491 records and 2 attributes, and it did not have any missing data. Figure 6.1 shows the distribution by "Rating". As can be seen, the dataset is quite unbalanced, with many more records with a positive evaluation (Rating 5:9054; Rating 4:6039) than with a negative evaluation (Rating 2:1793; Rating 1:1421). Furthermore, the number of records with an intermediate evaluation is also much lower than the number of records with a positive evaluation (Rating

Table 6.1: Small example of the used dataset.

Review	**Rating**
nice hotel expensive parking got good deal sta...	4
ok nothing special charge diamond member hilto...	2
nice rooms not 4 experience hotel monaco seat...	3
unique, great stay, wonderful time hotel monac...	5
great stay great stay, went seahawk game aweso...	5

3:2184).

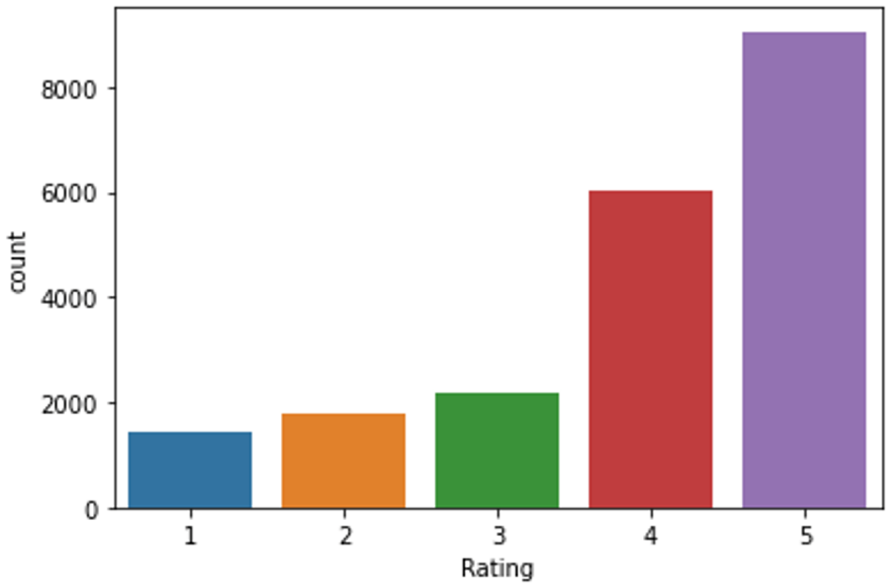


Figure 6.1: Distribution by "Rating".

To study possible correlations between the "Review" and the assigned "Rating", we created 3 new attributes: "Word_Count", "Char_Count", and "Average_Word_Length". The "Word_Count" stands for the number of words used in the "Review", the "Char_Count" stands for the number of characters used in the "Review", and the "Average_Word_Length" stands for the average size of the words used in the "Review". The "Average_Word_Length" did not show statistical relevance, but we found that the most negative reviews tended to be composed of more words than the most positive reviews (Figure 6.2), which made us believe that the attribute "Word_Count" would be very relevant for the creation of the model.

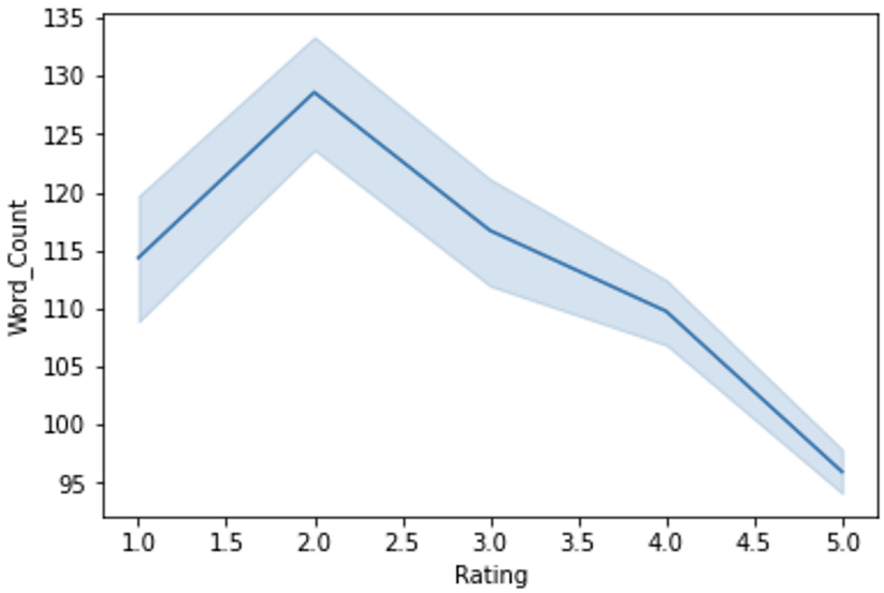


Figure 6.2: Correlation between the average number of words in the "Review" with the assigned "Rating".

In the next step, we analyzed which words were most used in the reviews. In addition, we analyzed which words were most used in negative reviews (Rating 1 and 2) and in positive reviews (Rating 3, 4, and 5). We found that many of the most used words were the same, both in positive and in negative reviews. In Table 6.2 are presented the most used words considering all the reviews. The fact that many of the most used words are the same, in both positive and negative reviews, made us wonder if eliminating these words would be a good strategy in creating the model.

Table 6.2: List of the most used words in reviews.

Word	#	**Word**	#	**Word**	#	**Word**	#	**Word**	#
hotel	42,079	not	30,750	room	30,532	great	18,732	n't	18,436
staff	14,950	good	14,791	did	13,433	just	12,458	stay	11,376
no	11,360	rooms	10,935	nice	10,918	stayed	10,022	location	9515
service	8549	breakfast	8407	beach	8218	food	8026	like	7677
clean	7658	time	7615	really	7612	night	7596	...	...

Then, we used some libraries to perform sentiment analysis. Sentiment analysis techniques allow the identification of people's opinions, feelings, or attitudes through their comments. These techniques make it possible to determine a sentiment in a

given sentence being classified as positive, negative, or neutral, using scalar values, and also through polarity (quantifying the sentiment as positive or negative through a value). These techniques are widely used in domains such as social networks, and their application is an excellent exercise to aid in interpreting and analyzing data from this particular field. Therefore, we applied 3 different libraries: Textblob, Vader, and Flair. Textblob and Vader presented similar results, while Flair did not obtain results that correlated with the "Rating". With Textblob, we obtained 2 new attributes (Polarity and Subjectivity), and with Vader, we obtained 3 new attributes Positive_Sentiment, Negative_Sentiment, and Neutral_Sentiment. Figure 6.3 presents the density of the "Polarity" attribute obtained with Textblob. We found that the "Polarity" is mostly positive, which makes sense since, as we saw earlier, most reviews are also positive.

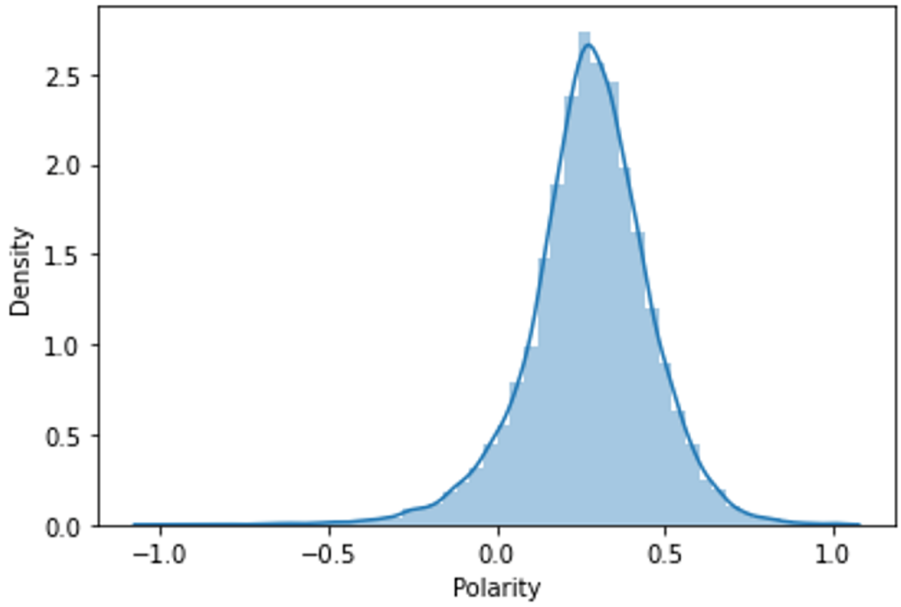


Figure 6.3: Density of the "Polarity" attribute obtained with Textblob.

Figure 6.4 presents the correlation between "Polarity" and "Rating". We can see that the polarity rises as the rating increases, which clearly demonstrates the existence of a correlation. However, we also found that the boxplots of each rating level are superimposed, which is a strong indicator of the difficulty in achieving success in creating classification models. In addition, we verified the existence of many outliers, which may not actually be accurate, as is the case for "Rating"

equal to 1, in which we verified the existence of many records with the polarity between −1 and −0.65. Figure 6.5 presents the correlation between "Subjectivity" and "Rating". As we can see, there does not seem to exist any kind of correlation between subjectivity and rating.

To create a more simplified version of the assessment made by tourists, we generated a new attribute called "Sentiment", with a value equal to 1 for records where the "Rating" was equal to or greater than 3 and with a value equal to 0 for records where the "Rating" was less than 3. This attribute will allow us to distinguish positive ratings from negative ratings.

We also carried out important preprocessing activities that allowed us to prepare the dataset and discover some important aspects. First, we put all the corpus in lowercase. Then, we tokenized the corpus and performed lemmatization and removed all the punctuation. In addition, we used other techniques, such as removing stopwords, stemming, and considering only the characters of the alphabet; however, these did not allow us to obtain better results. Finally, we used the MinMaxScaler to normalize the data.

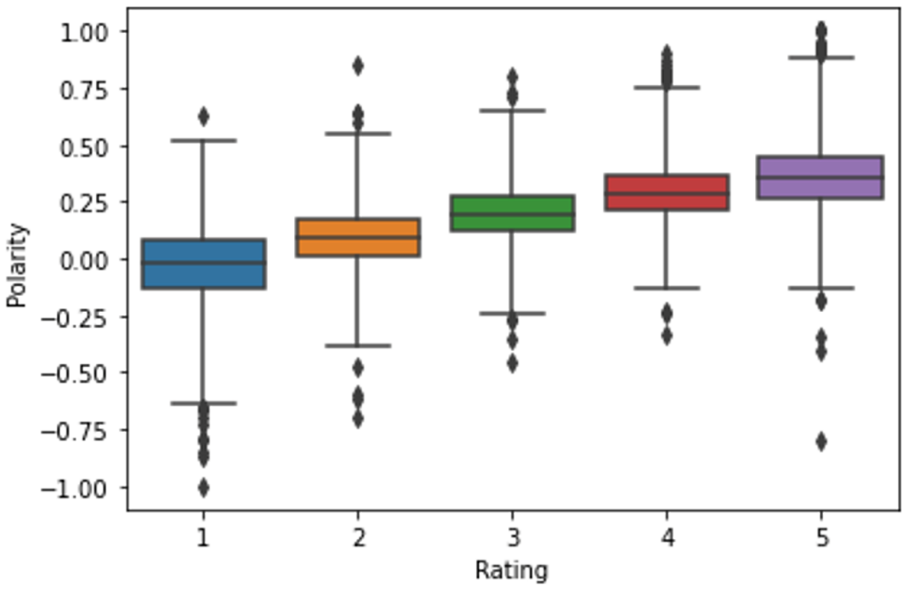


Figure 6.4: Correlation between "Polarity" and "Rating".

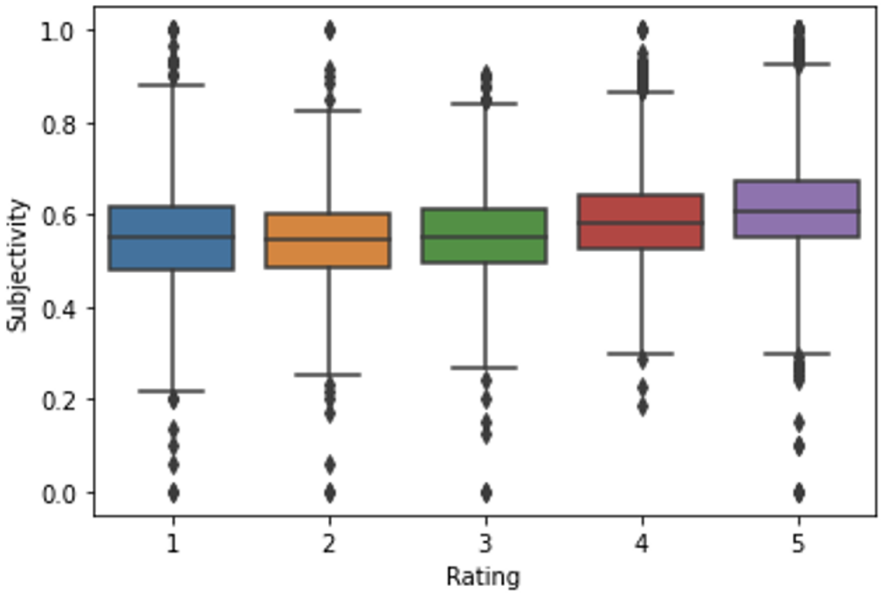


Figure 6.5: Correlation between "Subjectivity" and "Rating".

6.2.4. Computational Techniques

Considering the objective of this work, we believed that it would be important to test the results that would be possible to obtain with different algorithms, both as classification methods and as regression methods for supervised learning. We anticipated that if algorithms as classification methods failed due to previously identified limitations, algorithms as regression methods could be an acceptable alternative in the context of the objective of this work. Due to the vast number of existing methods, we decided to choose the classic and the most widely used in the literature. Our main criterion was the diversity of the mechanics with which these methods are structured. Hence, we chose methods from different categories based on decision trees, distances, neural networks, and decision boundaries. The algorithms used were: Logistic Regression [201], Random Forest [26], Decision Tree [161], K-Nearest Neighbors [72], and BiLSTM [88]. The first 4 used the Scikit-learn library and the last one used the Keras library.

We also considered applying unsupervised techniques used in AD problems. These methods are present in numerous domains and research fields. These can be found in industrial machinery failure [15, 179, 168], credit card fraud [30, 97, 167],

image processing [83, 50], medical and public health [98, 141, 204], network intrusion [176, 195, 193, 10], and others [46, 159, 170, 65, 133]. We focused on One-Class Classification (OCC) [186] methods to understand whether we could improve the results of the best classification algorithm. To do so, we selected three unsupervised methods from the Scikit-learn library: Isolation Forest, OCKNN, and LOF (described in Appendix A).

6.3. Tests and Evaluation

Several experiments were carried out with the selected algorithms to tune parameters for optimization. However, as no significant differences were found, the default configuration provided by the used libraries was employed for all algorithms. For estimating the performance of the ML models, we performed cross-validation with five repetitions.

We defined six different scenarios to create models. In the first three scenarios (#1, #2 and #3), the set of most used words that did not express feelings were removed (hotel, room, staff, did, stay, rooms, stayed, location, service, breakfast, beach, food, night, day, hotel, pool, place, people, area, restaurant, bar, went, water, bathroom, bed, restaurants, trip, desk, make, floor, room, booked, nights, hotels, say, reviews, street, lobby, took, city, think, days, husband, arrived, check, and told), and in the other 3 (#4, #5 and #6), all words were kept.

For all scenarios, we used the TfidfVectorizer class from the Scikit-learn library to transform the "Review_new" feature to feature vectors, and we defined max_features equal to 5000. In addition, in scenarios #1 and #4, the features considered were: "Review_new", "Polarity", "Word_Count", "Char_Count", "Average_Word_Length", "Positive_Vader_Sentiment", and "Negative_Vader_Sentiment"; in scenarios #2 and #5, the features considered were "Review_new" and "Polarity"; and in scenarios #3 and #6, only the feature "Review_new" was considered. We applied each supervised learning algorithm to each scenario with both the classification and regression methods. Thus, all combinations were used for a 5-class problem (Y = "Rating") and a 2-class problem (Y = "Sentiment"). Finally, we applied three AD methods to the output of the best classification model (2-class problem).

6.3.1. Classification and Regression Results with Supervised Methods

Figure 6.6 presents the results obtained with the five algorithms for each of the scenarios defined with the classification method for the 5-class problem (Y = "Rating"). Note that the Logistic Regression method is limited to two-class classification problems by default. However, with the Scikit-learn library, Logistic Regression can handle multi-class classification problems using the approach one-vs-rest [21]. Analyzing Figure 6.6, the Logistic Regression algorithm obtained the best results for all scenarios, with an accuracy always higher than 0.6, followed by the Random Forest algorithm. The other three algorithms obtained considerably lower results, and in the case of the BiLSTM algorithm, the results were very poor, as it classified all cases with a "Rating" of 4.

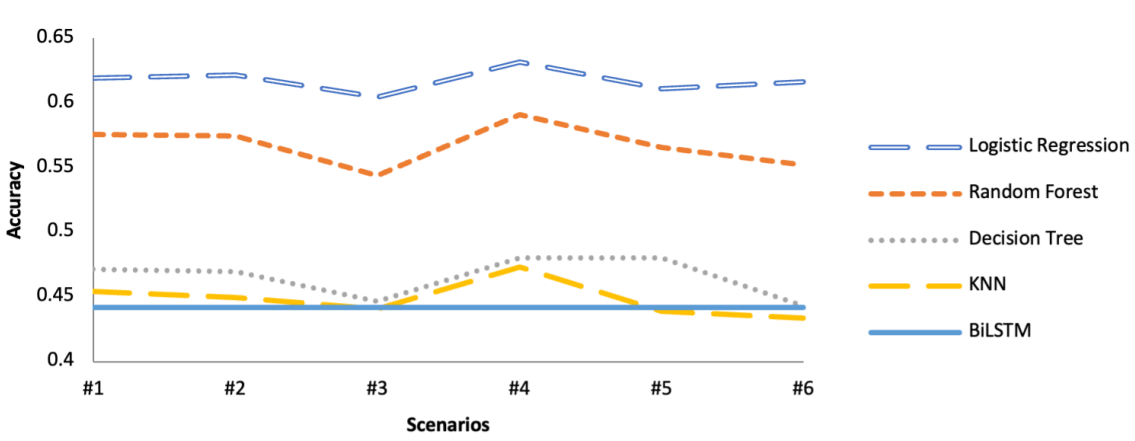


Figure 6.6: Algorithms' accuracy for the classification method (Y = "Rating").

Since scenario 4 was the one that allowed us to achieve the best results, in terms of accuracy, Table 6.3 presents precision and recall for each of the algorithms in scenario 4 with the classification method for the 5-class problem. We verified that the Logistic Regression and Random Forest algorithms presented interesting results. It is possible to verify that relatively high values were obtained for the extreme cases ("Rating" = 1 and "Rating" = 5), but the quality was quite low in the classification of intermediate values.

Table 6.3: Precision and recall for scenario 4 with the classification method (Y = "Rating").

	Precision					**Recall**				
	L 1	L 2	L 3	L 4	L 5	L 1	L 2	L 3	L 4	L 5
Logistic Regression	0.66	0.47	0.46	0.53	0.72	0.65	0.40	0.27	0.52	0.82
Random Forest	0.63	0.48	0.42	0.47	0.64	0.70	0.27	0.04	0.39	0.90
Decision Tree	0.49	0.33	0.23	0.39	0.62	0.50	0.32	0.23	0.39	0.62
KNN	0.37	0.20	0.19	0.40	0.64	0.60	0.22	0.18	0.31	0.67
BiLSTM	0	0	0	0.29	0	0	0	0	1	0

Figure 6.7 presents the results obtained with the 5 algorithms for each of the scenarios defined with the classification method for the 2-class problem (Y = "Sentiment"). As can be seen, the results were quite good. Once again, the Logistic Regression and Random Forest algorithms obtained the best results, with the Logistic Regression algorithm showing an accuracy very close to 0.95. The Decision Tree and K-Nearest Neighbors algorithms obtained reasonable results, mainly in scenarios where more features were considered. The BiLSTM algorithm returned the worst results.

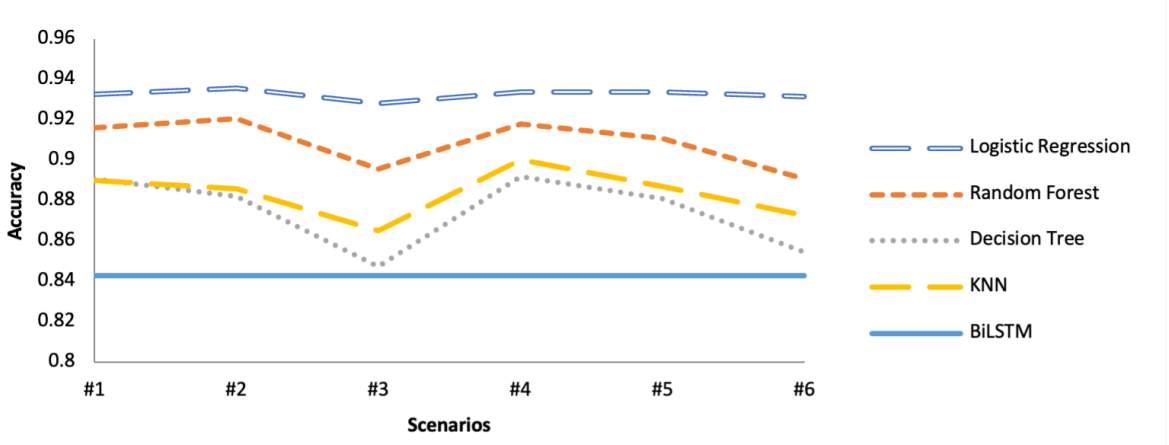


Figure 6.7: Algorithms' accuracy for the classification method (Y = "Sentiment").

Table 6.4 presents precision and recall for each of the algorithms in scenario 4 with the classification method for the 2-class problem. The results presented by the Logistic Regression algorithm are quite solid. It is verified that the recall for

L 1 (Sentiment = 0) is lower than desirable, but this is probably explained by the dataset being unbalanced.

Table 6.4: Precision and recall for scenario 4 with the classification method (Y = "Sentiment").

	Precision		**Recall**	
	L 1	**L 2**	**L 1**	**L 2**
Logistic Regression	0.849624	0.946389	0.702736	0.976846
Random Forest	0.873541	0.922977	0.558458	0.98495
Decision Tree	0.657431	0.934858	0.649254	0.937022
KNN	0.735152	0.923111	0.569652	0.96179671
BiLSTM	0	0.843061	0	1

The next experiences concern the application of the algorithms to the previously presented scenarios with the regression method. Figure 6.8 presents the Mean Absolute Error obtained with the 5 algorithms for each of the scenarios defined with the regression method for the 5-class problem (Y = "Rating"). We found that most algorithms obtained poor results. However, the Random Forest algorithm presented very interesting results, obtaining a Mean Absolute Error of 0.69 in scenario 4 (which is quite good considering the problem in question).

Table 6.5 presents the Mean Squared Error, Root Mean Square Error, and Mean Absolute Error for each of the algorithms in scenario 4 with the regression method for the 5-class problem. Once again, it is possible to verify that the Random Forest algorithm obtained very good results, unlike the other algorithms. Although the BiLSTM algorithm seems to give reasonable results, this only happens due to the fact that it always generates the same output and most reviews are positive.

Figure 6.9 presents the Mean Absolute Error obtained with the 5 algorithms for each of the scenarios defined with the regression method for the 2-class problem (Y = "Sentiment"). We verified that, in this case, all algorithms, with the exception of the BiLSTM algorithm, obtained very good results.

Table 6.6 presents the Mean Squared Error, Root Mean Square Error, and Mean Absolute Error for each of the algorithms in scenario 4 with the regression method for the 2-class problem. The Logistic Regression algorithm again presented very good

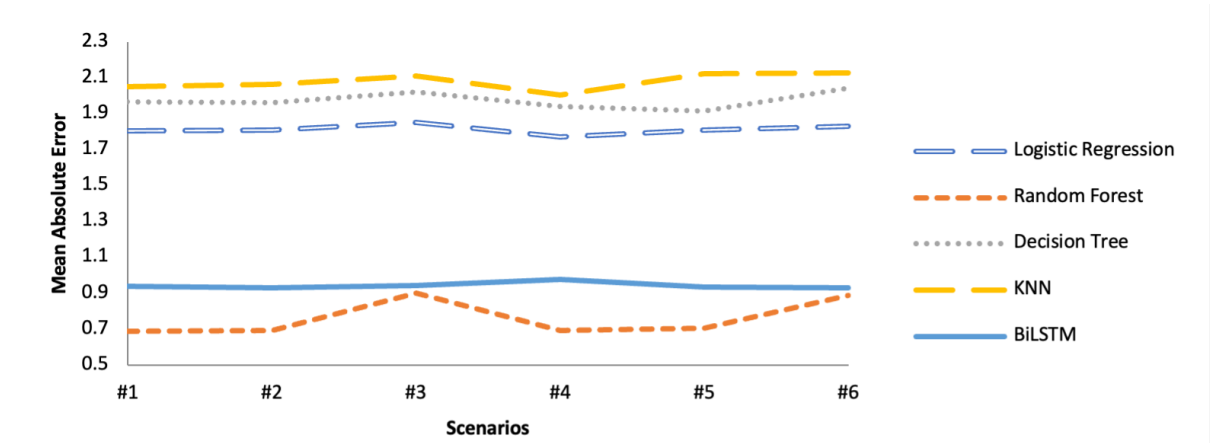


Figure 6.8: Algorithms' Mean Absolute Error for the regression method (Y = "Rating").

Table 6.5: Mean Square Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) for scenario 4 with the regression method (Y = "Rating").

	MSE	**RMSE**	**MAE**
Logistic Regression	4.140872	2.034913	1.77198
Random Forest	0.733771	0.856604	0.694623
Decision Tree	8.018544	2.831703	1.942417
KNN	5.818965	2.412253	2.007092
BiLSTM	1.522414	1.233862	0.978359

results that were consistent across all experiments. In this scenario, the K-Nearest Neighbors algorithm also presented interesting results.

6.3.2. Anomaly Detection Results

Through our next experiments, we selected OCC methods used in AD problems. We applied them to score the predicted output of the best classification algorithm—in this case, the Logistic Regression. These anomaly detectors are trained with normal data, identifying patterns that deviate from normality, which are considered

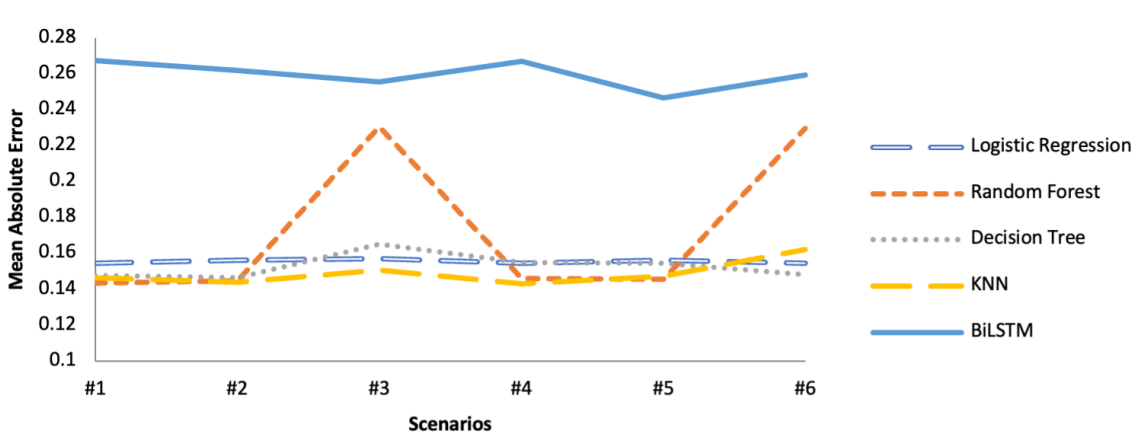


Figure 6.9: Algorithms' Mean Absolute Error for the regression method (Y = "Sentiment").

Table 6.6: MSE, RMSE, and MAE for scenario 4 with the regression method (Y = "Sentiment").

	MSE	**RMSE**	**MAE**
Logistic Regression	0.097812	0.312749	0.154837
Random Forest	0.07346	0.271034	0.146088
Decision Tree	0.154987	0.393684	0.154987
KNN	0.095474	0.308988	0.143406
BiLSTM	0.132327	0.363768	0.267391

anomalies. The main goal is to analyze whether these techniques can help the recommendation system that we intend to develop to correctly classify as many users as possible—that is, to detect whether they like a POI, improving the Logistic Regression performance.Therefore, as we can observe in Figure 6.11, the class 0 (showed as red dots), which we have considered as the anomalous one in this scenario, is dispersed through the graph in the Isolation Forest and OCKNN methods. We can also visualize that users with negative sentiments are at the top for the LOF method, with the highest scores. However, some of them are overlapped with users with positive sentiments, which means that although improvements in reducing false positives are possible, they come with the cost of increasing false negatives. We

identified LOF as the best method to apply for this purpose as it was shown to better separate the Y = "Sentiment" classes through its score compared to the other methods.

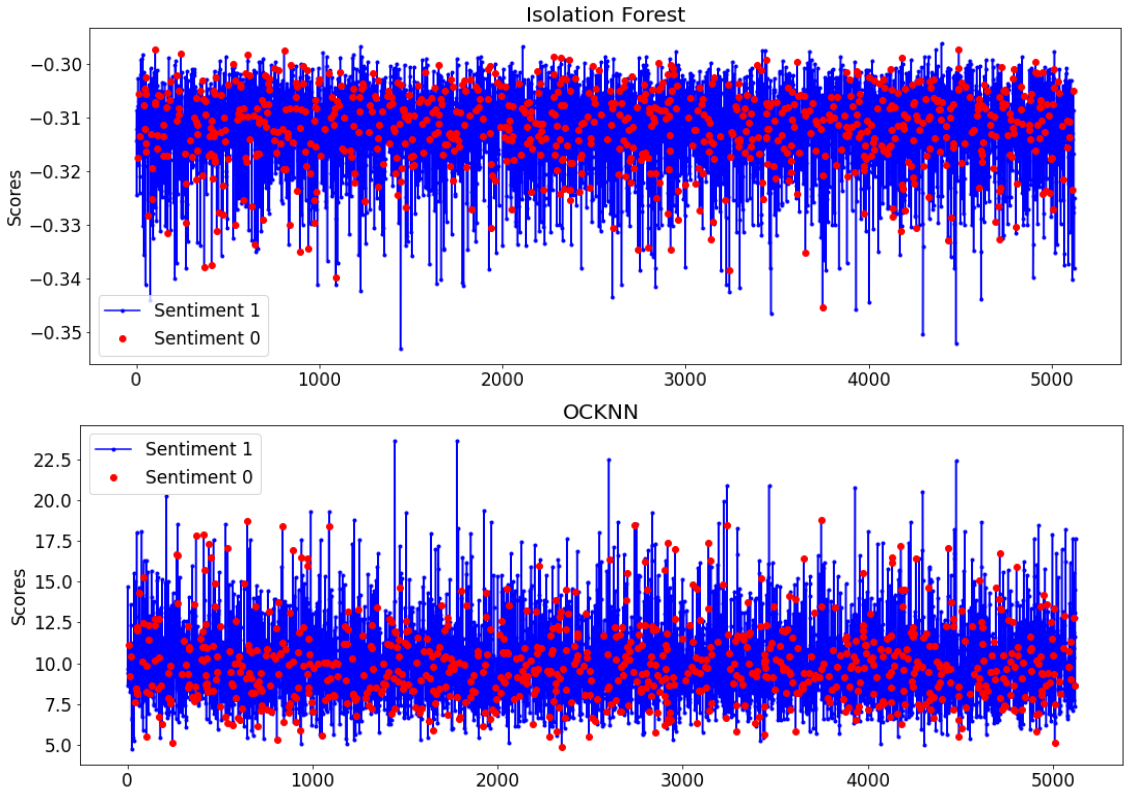


Figure 6.10: *Cont.*

We then performed four different experiments with this technique, analyzing the precision and recall metrics, as we intended to reduce false positives (increase precision), taking the increase in false negatives (decrease recall) into account. Thus, in the first two experiments, we applied LOF to separate Y = "Sentiment" classes by training with users with positive sentiments to isolate users with negative sentiments in the first experiment, while in the second experiment, we did the same, switching the classes (training with users with negative sentiment to isolate users with positive sentiment). We repeated the process for experiments three and four, this time using Y = "Ranking" to isolate the extreme ranking values, meaning that, in experiment three, we used users who rated 5 to train in order to isolate users who rated 1 and

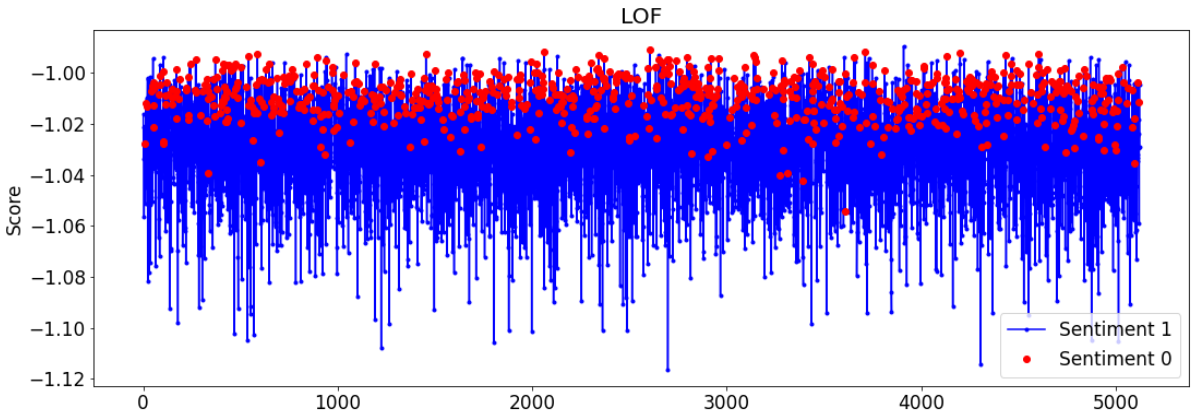


Figure 6.11: Anomaly scores distinguishing sentiment 1 from sentiment 0. The first graphic represents the Isolation Forest results, the second shows OCKNN results, and the third shows LOF results. The y-axis represents the scores and the x-axis represents the sample indices.

vice versa for the fourth experiment.

To visualize the experiments, we built different graphics (Figures 6.12–6.15). The y-axis represents precision and recall percentage values, and in the x-axis, the percentile thresholds from LOF are given. That is, the percentage instances with the highest score from LOF output are considered the isolated class. For example, threshold percentile 95% means that instances that have a score value greater than 95% of the highest score output are considered as the isolated class.

We can observe in all experiments that recall presents a linear increase when threshold values also increase, while precision shows a slight decrease for high threshold values. It is essential to mention that threshold percentile 100% represents the output of Logistic Regression without cuts, which is why recall is always 100%, which means the absence of false negatives since we are using the values predicted by the Logistic Regression method of only a specific class.

In the first experiment (Figure 6.12), we aimed to discard class 0 from the Logistic Regression output, reducing the population from class 1 in order to obtain the maximum users who liked a POI. We can see that if using a threshold percentile of 50%, we obtain approximately 99% precision, but with a high cost for the recall value (52%). In this experiment, precision has a slight increase when reducing the

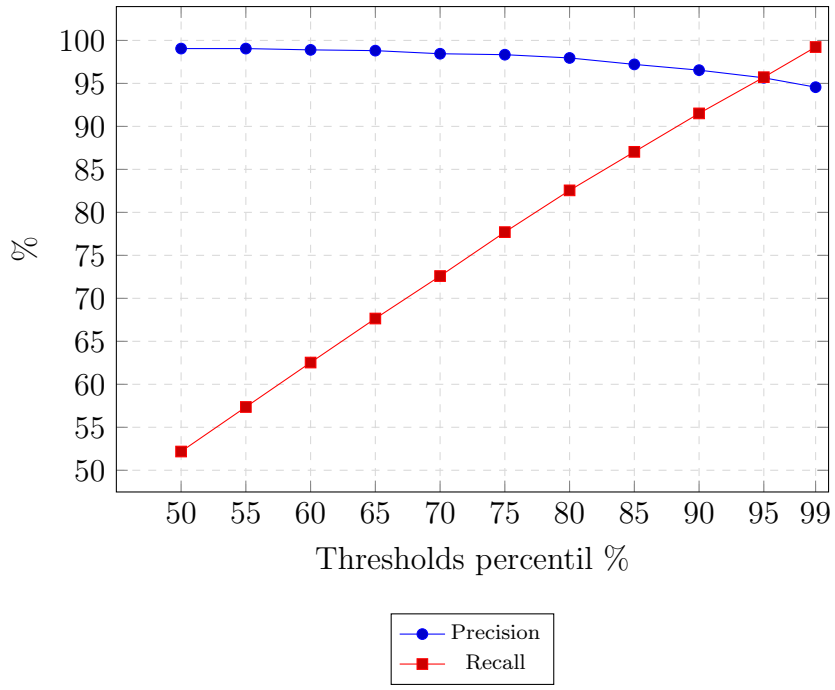


Figure 6.12: First experiment—isolating class 0 from Y = "Sentiment" in Logistic Regression output using LOF.

class 1 population in 20% (threshold percentile 80%), achieving a precision of 98%, while recall decreases at 82%. It obtains an acceptable recall value while precision converges to its highest value.

In Figure 6.13, it is possible to observe the experiment in which we intended to hit the highest number of users who did not like a POI. In this scenario, LOF shows poor performance since it could not separate adequately class 1 from class 0. In order to be able to increase precision in only 2% (from 84% to 86%), recall drops from 99% to 51%.

Regarding the third experiment, shown in Figure 6.14, our goal was to discard users who rated a POI as 1, while reaching the maximum number of users who rated a specific POI as 5. Regarding the third experiment, shown in Figure 6.14, we wanted to discard users who rated a POI as 1 while reaching the maximum number of users who rated a specific POI as 5; it can be seen that precision can increase from 73% to 78% when reducing the population from users who rate 5 in

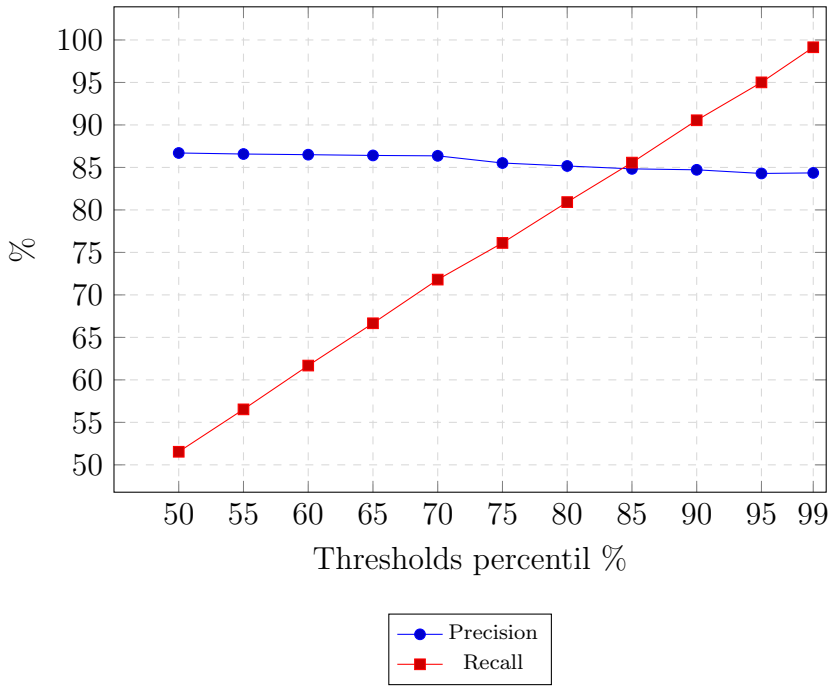


Figure 6.13: Second experiment—isolating class 1 from Y = "Sentiment" feature in Logistic Regression output using LOF.

50%. This increase of 5% is the same, visible in the first (Figure 6.12) and last experiment (Figure 6.15); however, the highest precision value is much higher in the first scenario.

6.3.3. Discussion

In this chapter, we carried out several experiments to understand the capability of machine learning models to predict user reviews on the TripAdvisor platform. We started with the classification and regression of two problems, multi-class (Y = "Rating") and binary (Y = "Sentiment"), to observe the models' behaviour. The results in the multi-class problems were not very high, especially in identifying the intermediate classes (Rating 2, 3, 4) due to the composition of the dataset. In the dataset analysis, we verified that, in addition to the classes being unbalanced (Figure 6.1), there is an overlap in the user evaluations (Figure 6.4). On the one

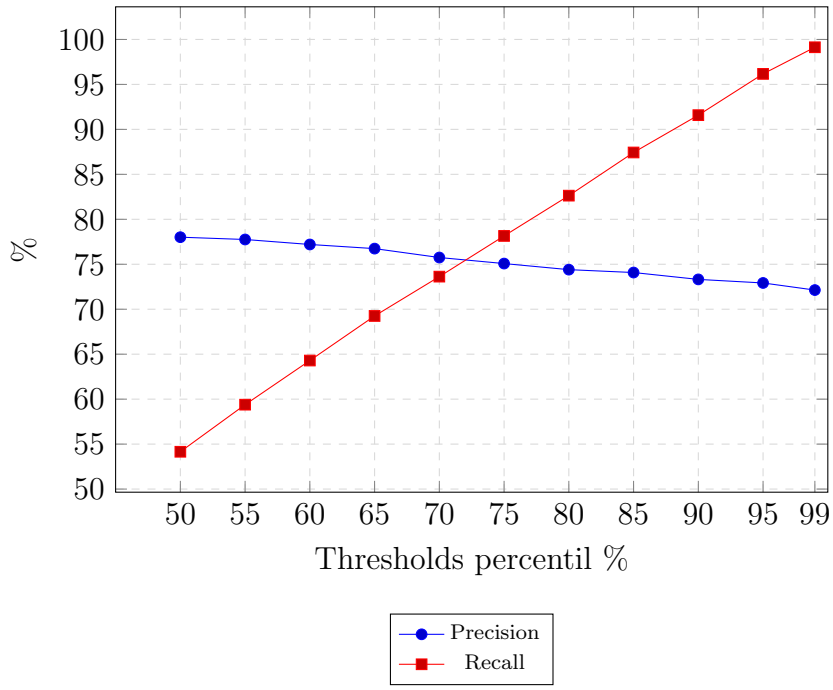


Figure 6.14: Third experiment—isolating class 1 from Y = "Ranking" in Logistic Regression output using LOF.

hand, the dataset may not be sufficiently representative—for example, in comments with a level 3 rating—and, on the other hand, the fact that users are different can also have a large impact on a scale from 1 to 5, i.e., the same words have different meanings/weights for different people and people who evaluate a POI with the same rating may express it in a completely different way. As expected, the binary problem = Sentiment) results were higher since the data were aggregated by the extreme ratings (1, 5), where the overlapped observations were minor compared to intermediate ratings. Since our goal was to identify those ratings classified as positive, which actually obtained a positive rating from the user (and vice versa), we applied AD techniques to improve the Logistic Regression precision. We verified that the LOF was the best AD method to better differentiate classes from the Logistic Regression output compared to OCKNN and Isolation Forest. The LOF algorithm could reduce false positives but with an associated cost (with linear growth derived from the noise present in the dataset) of increasing false negatives, which is excellent since it is essential that the recommendation system we intend to develop

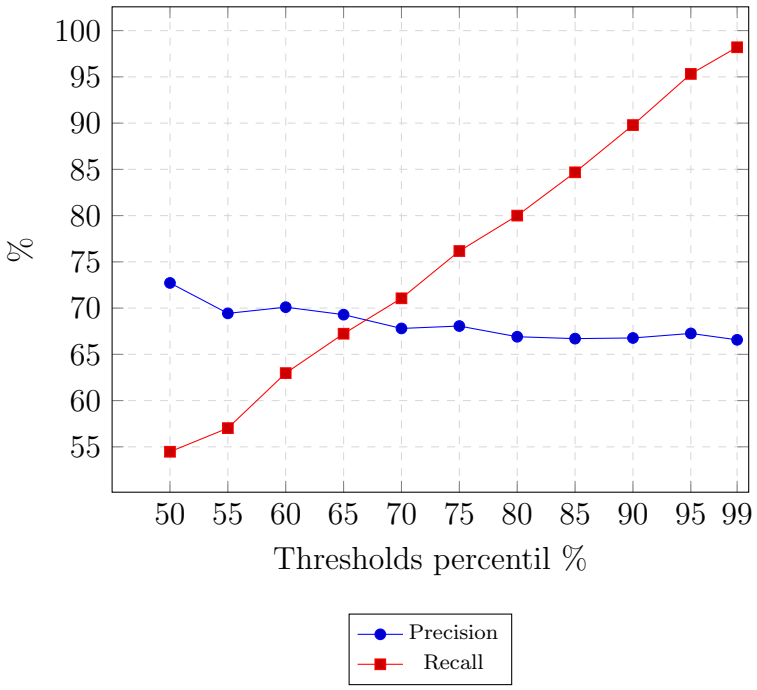


Figure 6.15: Fourth experiment—isolating class 5 from Y = "Ranking" in Logistic Regression output using LOF.

can identify POIs that users will like or not like with certainty.

6.4. Conclusions

This work aimed to study strategies to automatically predict tourists' preferences regarding tourism points of interest. The method consisted in using Machine Learning algorithms and Natural Language Processing techniques on reviews that tourists posted on TripAdvisor to predict their assigned ratings. The chosen dataset presented several issues, making it difficult to achieve better results (the top three were being unbalanced, having comments that were not about the POI, and having comments with very poor writing quality). Since this was a public dataset, we already knew it would be extremely challenging because most existing works present accuracy rates between 30% and 60%. However, we decided to use this dataset as it is a good example of the reality and type of problems that exist in the context of

the topic of this work.

The work carried out allowed us to reach important conclusions. First, the inclusion of sentiment analysis had a much smaller positive impact than expected. Furthermore, it was possible to notice that, for this dataset, the Vader and TextBlob models obtained a good correlation with the ratings associated with comments, while Flair did not. Second, although negative comments are usually longer, the inclusion of the "Word_Count" attribute did not prove to be relevant. Third, the Logistic Regression algorithm proved to be, for classification, the one that achieved greater accuracy, while the Random Forest algorithm, for regression, proved to be the one that obtained the smallest error. The Bidirectional LSTM algorithm obtained poor results for both classification and regression, most likely because the dataset was not large enough and contained several outliers, making it difficult for LSTM to extract patterns and generalize the data.

Finally, we verified that we can improve the precision of a model using AD techniques, albeit with a certain decrease in recall. The cost of increasing false negatives is defined by the anomalous threshold, which is a user-specified parameter. Therefore, the threshold can be adjustable so that there is a beneficial trade-off between precision and recall. We intended to create a model to identify only those tourists who truly like or dislike a particular point of interest, in which the main objective is not to identify everyone, but fundamentally not to fail those who are identified in those conditions.

Our experiments provide valuable information as they give an idea of the behaviour of ML models in a real scenario, helping to develop approaches for those who intend to create a recommendation system for decision support systems in the tourism field.

Chapter 7

Conclusions and Future Work

AD is a crucial task in many fields. The ability to identify and analyze unusual or unexpected patterns in data can help organizations prevent potential problems and improve their operations. Over the years, the technology used for AD has evolved significantly. Early approaches relied on simple statistical methods and manual inspection of data, but these were limited in their ability to handle large and complex data sets. Today, AD is a vast and widely studied field where new approaches are constantly emerging. With the advent of ML and big data technologies, the field of AD has seen significant advancements and will continue to be an important area of research and development in the future. Despite of such improvements, there are still limitations to be addressed such as scalable methods capable of dealing with large volumes of data, the lack of transparency in most SotA methods, the computational costs of hyperparameter tuning, or the ability to deal with stream data without the need for labels. In this thesis, we have tried to confront some of these problems, and several methods and comparative studies were presented and proposed to address the challenges described in Chapter 1. Below we briefly detail the main contributions of this work:

7.1. New algorithms and models

The new algorithms and models proposed to address the challenges of AD are the following:

- The first contribution is the LSHAD algorithm, which is a novel and sustainable AD method based on LSH capable of dealing with large-scale datasets. The resulting algorithm is highly parallelizable and its implementation in Apache Spark further increases its ability to handle very large datasets. Moreover, the algorithm incorporates an automatic hyperparameter tuning mechanism so that users do not have to implement costly manual tuning. This sustainable approach to AD has the added advantage of being an unsupervised method, that does not need labelled data for the learning process. The hyperparameter tuning mechanism is able to adjust its hyperparameters despite the input data domain. The LSHAD method is novel, and both of its characteristics, hyperparameter automation and distributed properties are not usual in AD techniques. The results for experiments with LSHAD across a variety of datasets point to SotA AD performance while handling much larger datasets than SotA alternatives. In addition, evaluation results for the tradeoff between AD performance and scalability show that our method offers significant advantages over competing methods.

- Another contribution is the data-driven approach for AD using a data stream model, which is designed also to be more sustainable than other available models. The proposed method assists in the early detection of failures and errors in machinery before they reach critical stages. We present an AD model following an unsupervised approach, combining the Half-Space-trees method with One Class K Nearest Neighbor, adapted to deal with data streams. Since the data incoming from the sensors is endless and received as a continuous flow, the framework is capable of dealing with data streams where the methods' computational resources are limited (memory, computational power, processing time) providing sustainability to the method. The model is based on incremental learning as data is induced incrementally and contemplate a forgetting mechanism to deal with limited memory. We evaluate our approach and compare it with the Half-Space-Trees method applied without the One Class K

Nearest Neighbor combination. Our model produced few type I errors, significantly increasing the value of precision when compared to the Half-Space-Trees model. Our proposal achieved high AD performance, predicting most of the catastrophic failures of the APU train system.

- Lastly, the thesis presents a study of strategies with NLP techniques for predicting tourist preferences based on their reviews. We explored different Machine Learning methods to predict users' ratings. We used NLP strategies to predict whether a review is positive or negative and the rating assigned by users on a scale of 1 to 5. We then applied supervised methods such as Logistic Regression, Random Forest, Decision Trees, K-Nearest Neighbors, and Recurrent Neural Networks to determine whether a tourist likes/dislikes a given point of interest. Additionally, we used a distinctive approach in this field through unsupervised techniques for AD problems. The goal was to improve the supervised model in identifying only those tourists who truly like or dislike a particular point of interest, in which the main objective is not to identify everyone, but fundamentally not to fail those who are identified in those conditions. The experiments carried out showed that the developed models could predict with high accuracy whether a review is positive or negative but have some difficulty in accurately predicting the rating assigned by users. Unsupervised method Local Outlier Factor improved the results, reducing Logistic Regression false positives with an associated cost of increasing false negatives.

7.2. Practical Applications

This thesis also explores the application of the proposed algorithms and models in various fields. The main application is intrusion detection, as cyber security is a critical area in computer systems, especially when dealing with sensitive data. At present, it is becoming increasingly important to assure that computer systems are secured from attacks due to modern society's dependence on those systems. To prevent these attacks, nowadays most organizations make use of anomaly-based IDS. Usually, IDS contain machine learning algorithms which aid in predicting or detecting anomalous patterns in computer systems. Most of these algorithms are supervised techniques, which contain gaps in the detection of unknown patterns

or Zero-day exploits since these are not present in the algorithm learning phase. To address this problem, we present in this thesis an empirical study of several unsupervised learning algorithms used in the detection of unknown attacks. In this study, we evaluated and compared the performance of different types of AD techniques in two publicly available datasets: the NSL-KDD and the ISCX. The aim of this evaluation allows us to understand the behaviour of these techniques and understand how they could be fitted in an IDS to fill the mentioned flaw. Also, the present evaluation could be used in the future, as a comparison of results with other unsupervised algorithms applied in the cybersecurity field. The results obtained show that the techniques used are capable of carrying out AD with acceptable performance and thus making them suitable candidates for future integration in Intrusion Detection tools.

In addition to intrusion detection, an evaluation study of novel unsupervised methods to deal with IoT attacks is presented in this thesis. The challenge of providing security to networks is becoming increasingly harder, particularly with the recent deluge of smart devices with average to poor security joining networks worldwide. With the evolution of the IoT and devices increasingly connected to the Internet, the challenge of ensuring the security and integrity of the network and all connected devices arises. The use of IDS seeks to help protect networks, for example, by preventing IoT devices from being used maliciously or raising awareness when they have been compromised. This dissertation explores the new Aposemat IoT-23 dataset of network traffic containing malware and benign scenarios executed in IoT devices. We initially performed an exploratory analysis of the dataset. We used it to evaluate several AD methods to evaluate their capability to distinguish between normal and abnormal network behaviour. Therefore, one of our contributions is to present a comparison of SotA solutions to detect intrusions in IoT. In our evaluation, we tested AD rates and time processing performances but also explored the usefulness of the explanation trees obtained from a novel AD algorithm denominated by **EADMNC**. Results showed that most of the methods under scrutiny have an excellent AD score, but the One-Class Nearest Neighbour is systematically the best performer. However, this method has the disadvantage of its lack of scalability and, thus, we show that novel methods such as **LSHAD** and **EADMNC** are much better suited for dealing with large datasets.

Another field widely studied in AD is the PdM. The emergence of the Industry 4.0 trend brings automation and data exchange to industrial manufacturing. Using computational systems and IoT devices allows businesses to collect and deal with vast volumes of sensorial and business process data. The growing and proliferation of big data and machine learning technologies enable strategic decisions to be made based on the analyzed data. It is presented in this thesis a data-driven PdM framework for the APU) system of a train of *Metro do Porto*. The proposed model is also applicable in PdM field, where it has been shown to be effective in identifying and predicting failures in equipment, reducing downtime and costs. This is achieved by using stream data, which enables it to process data in real-time and detect anomalies in a timely manner.

In addition, we addressed the Argumentation-based dialogue models field. These models have shown to be appropriate for decision contexts in which it is intended to overcome the lack of interaction between decision-makers, either because they are dispersed, they are too many, or they are simply not even known. However, to support decision processes with argumentation-based dialogue models, it is necessary to have knowledge of certain aspects that are specific to each decision-maker, such as preferences, interests, and limitations, among others. Failure to obtain this knowledge could ruin the model's success. We sought to facilitate the information acquisition process by studying strategies to automatically predict the tourists' preferences (ratings) in relation to points of interest based on their reviews. The study of tourist preferences prediction based on reviews, using Argumentation-based dialogue models, also provided insights into the use of NLP and unsupervised techniques in this field and showed how this approach can facilitate the information acquisition process and predict the tourists' preferences based on their reviews. The results of this study demonstrate the potential of this approach in decision-making contexts, and its ability to overcome the lack of interaction between decision-makers.

As described, this thesis covers a broad suite of problems arising from the advent of the AD field, while also taking into account sustainability considerations. The proposed approaches have demonstrated their capability to deal with large amounts of data unlabeled, providing hyperparameter tuning mechanisms and data interpretability. Thus, it is expected that the contribution of this thesis will open the door to the development of new sustainable approaches that can deal with AD

problems taking into account the referred limitations.

7.3. Future Work

As future work, we plan to develop new unsupervised AD approaches in the PdM domain. Most of the works in the literature related to data-driven approaches in PdM apply classical machine learning techniques or more complex methods in the deep learning field. However, unsupervised learning methods that deal with data streams are not usually applied even though performing as well as the previously referred methods with additional benefits such as needing less computational resources with limited and incremental learning characteristics.

We are currently working on a solution for detecting railway defects and classifying their severity through data stream methods. We intend to develop a model that is not only able to detect and classify faults but also is robust to adversarial attacks. These attacks involve the deliberate introduction of unusual or unexpected behaviour in a system in order to cause it to malfunction or behave in a way that is beneficial to the attacker, which in this case could lead to an undetected defect in the railway causing in the worst scenario a train derailment.

7.4. Publications from the thesis

The contents of the present research have been published in the following specialized journals and conferences:

Journal publications

- J Meira, R Andrade, I Praça, J Carneiro, V Bolón-Canedo, A Alonso-Betanzos and G Marreiros. Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. Journal of Ambient Intelligence and Humanized Computing 11 (11), 4477-4489. 2020. doi:10.1007/s12652-019-01417-9. JCR Q1.

- J Meira, C Eiras-Franco, V Bolón-Canedo, G Marreiros, and A Alonso-Betanzos. Fast anomaly detection with locality-sensitive hashing and hyperparameter autotuning. Information Sciences, 607, pp.1245-1264. 2022. doi:10.1016/j.ins.2022.06.035. JCR Q1.

- J Meira, B Veloso, V Bolón-Canedo, M Goreti, and A Alonso-Betanzos. Data-driven predictive maintenance framework for railway systems. Intelligent Data Analysis, In Press. 2023. JCR Q3.

- J Meira, J Carneiro, V Bolón-Canedo, A Alonso-Betanzos, P Novais, and G Marreiros. Anomaly Detection on Natural Language Processing to Improve Predictions on Tourist Preferences. Electronics, 11(5), p.779. 2022. doi:10.3390/electronics11050779. JCR Q2

Journal publications (Under review process)

- J Meira, C Eiras-Franco, V Bolón-Canedo, G Marreiros, A Alonso-Betanzos, and J Hernandez-Castro. Unsupervised learning methods for IoT IDS with Aposemat IoT-23 Dataset. 2022.

Conferences

- Meira, J., Andrade, R., Praça, I., Carneiro, J. and Marreiros, G. Comparative results with unsupervised techniques in cyber attack novelty detection. In International Symposium on Ambient Intelligence (pp. 103-112). 2018. Springer, Cham. doi:10.1007/978-3-030-01746-0_12

7.5. Other Publications

During the PhD program, other works have been developed and published in the following specialized journals and conferences:

Journal publications

- Martinho, D., Freitas, A., Sá-Sousa, A., Vieira, A., Meira, J., Martins, C. and Marreiros, G. A Hybrid Model to Classify Patients with Chronic Obstructive Respiratory Diseases. Journal of Medical Systems, 45(3), pp.1-11. 2021. doi:10.1007/s10916-020-01704-5. JCR Q1
- Conceição, L., Rodrigues, V., Meira, J., Marreiros, G. and Novais, P. Supporting Argumentation Dialogues in Group Decision Support Systems: An Approach Based on Dynamic Clustering. Applied Sciences, 12(21), p.10893. 2022. doi:10.3390/app122110893. JCR Q2

Conferences

- De Berardinis, J., Pizzuto, G., Lanza, F., Chella, A., Meira, J. and Cangelosi, A. At your service: Coffee beans recommendation from a robot assistant. In Proceedings of the 8th International Conference on Human-Agent Interaction (pp. 257-259). 2020. doi:10.1145/3406499.3418765.
- Crista, V., Martinho, D., Meira, J., Carneiro, J., Corchado, J. and Marreiros, G. A Hybrid Model to Classify Physical Activity Profiles. In International Conference on Practical Applications of Agents and Multi-Agent Systems (pp. 268-278). 2022. Springer, Cham. doi:10.1007/978-3-031-18697-4_22.
- Carneiro, J., Meira, J., Novais, P. and Marreiros, G. Using Machine Learning to Predict the Users Ratings on TripAdvisor Based on Their Reviews. In Practical Applications of Agents and Multi-Agent Systems (pp. 127-138). 2021. Springer, Cham. doi:10.1007/978-3-030-85710-3_11.

Bibliography

[1] M. A. Abdulhayoglu and B. Thijs. Use of locality sensitive hashing (lsh) algorithm to match web of science and scopus. *Scientometrics*, 116(2):1229–1245, 2018. pages 33

[2] T. A. Ahanger, A. Aljumah, and M. Atiquzzaman. State-of-the-art survey of artificial intelligent techniques for iot security. *Computer Networks*, page 108771, 2022. pages 63

[3] J. Aiken and S. Scott-Hayward. Investigating adversarial attacks against network intrusion detection systems in sdns. In *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–7. IEEE, 2019. pages 7

[4] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics (demonstrations)*, pages 54–59, 2019. pages 105

[5] S. M. Al-Ghuribi and S. A. M. Noah. Multi-criteria review-based recommender system–the state of the art. *IEEE Access*, 7:169446–169468, 2019. pages 105

[6] I. Al-Turaiki and N. Altwaijry. A convolutional neural network for improved anomaly-based network intrusion detection. *Big Data*, 9(3):233–252, 2021. pages 7

[7] M. H. Alam, W.-J. Ryu, and S. Lee. Joint multi-grain topic sentiment: modeling semantic aspects for online reviews. *Information Sciences*, 339:206–223, 2016. pages 107

[8] A. Aleroud and G. Karabatis. Toward zero-day attack identification using linear data transformation techniques. In *2013 IEEE 7th International Conference on Software Security and Reliability*, pages 159–168. IEEE, 2013. pages 18

[9] I. Alhussein and A. H. Ali. Application of dbscan to anomaly detection in airport terminals. In *2020 3rd International Conference on Engineering Technology and its Applications (IICETA)*, pages 112–116. IEEE, 2020. pages 6

[10] S. A. Althubiti, E. M. Jones, and K. Roy. Lstm for anomaly-based network intrusion detection. In *2018 28th International telecommunication networks and applications conference (ITNAC)*, pages 1–3. IEEE, 2018. pages 113

[11] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 811–820, 2007. pages 98

[12] E. Anthi, A. Javed, O. Rana, and G. Theodorakopoulos. Secure data sharing and analysis in cloud-based energy management systems. In *Cloud Infrastructures, Services, and IoT Systems for Smart Cities*, pages 228–242. Springer, 2017. pages 64

[13] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap. A supervised intrusion detection system for smart home iot devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, 2019. pages 64

[14] B. Arrington, L. Barnett, R. Rufus, and A. Esterline. Behavioral modeling intrusion detection system (bmids) using internet of things (iot) behavior-based anomaly detection via immunity-inspired algorithms. In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–6. IEEE, 2016. pages 67

[15] G. Aydemir and B. Acar. Anomaly monitoring improves remaining useful life estimation of industrial machinery. *Journal of Manufacturing Systems*, 56:463–469, 2020. pages 112

[16] M. Bahri, F. Salutari, A. Putina, and M. Sozio. Automl: state of the art with a focus on anomaly detection, challenges, and research directions. *International Journal of Data Science and Analytics*, pages 1–14, 2022. pages 31, 43

[17] M. Barros, B. Veloso, P. M. Pereira, R. P. Ribeiro, and J. Gama. Failure detection of an air production unit in operational context. In J. Gama, S. Pashami, A. Bifet, M. Sayed-Mouchawe, H. Fröning, F. Pernkopf, G. Schiele, and M. Blott, editors, *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*, pages 61–74, Cham, 2020. Springer International Publishing. pages 87, 89, 90

[18] L. Basora, X. Olive, and T. Dubot. Recent advances in anomaly detection methods applied to aviation. *Aerospace*, 6(11):117, 2019. pages 6

[19] P. Bhandaru. What is botnet, prevention and detection techniques, 2018. pages 163

[20] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. pages 166

[21] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. pages 114

[22] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021. pages 2, 4

[23] H. Bostani and M. Sheikhan. Hybrid of anomaly-based and specification-based ids for internet of things using unsupervised opf based on mapreduce approach. *Computer Communications*, 98:52–71, 2017. pages 67, 68

[24] I. Botana, C. Eiras-Franco, and A. Alonso-Betanzos. Regression tree based explanation for anomaly detection algorithm. *MDPI Proceedings*, 54:7, 2020. pages 8, 65, 73, 176

[25] A. Bouguettaya, H. Zarzour, A. M. Taberkit, and A. Kechida. A review on early wildfire detection from unmanned aerial vehicles using deep learning-based computer vision algorithms. *Signal Processing*, 190:108309, 2022. pages 34

[26] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. pages 104, 112

[27] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000. pages 30, 34, 105, 175

[28] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos. Deep learning with dense random neural networks for detecting attacks against iot-connected home environments. In *International ISCIS Security Workshop*, pages 79–89. Springer, Cham, 2018. pages 66, 68

[29] Z. A. Bukhsh, A. Saeed, I. Stipanovic, and A. G. Doree. Predictive maintenance using tree-based classification techniques: A case of railway switches. *Transportation Research Part C: Emerging Technologies*, 101:35–54, 2019. pages 87, 88, 90

[30] F. Carcillo, Y.-A. Le Borgne, O. Caelen, Y. Kessaci, F. Oblé, and G. Bontempi. Combining unsupervised and supervised learning in credit card fraud detection. *Information sciences*, 2019. pages 112

[31] J. Carneiro, P. Alves, G. Marreiros, and P. Novais. A multi-agent system framework for dialogue games in the group decision-making context. In *World Conference on Information Systems and Technologies*, pages 437–447. Springer, 2019. pages 104

[32] J. Carneiro, P. Alves, G. Marreiros, and P. Novais. Group decision support systems for current times: Overcoming the challenges of dispersed group decision-making. *Neurocomputing*, 423:735–746, 2021. pages 104

[33] J. Carneiro, R. Andrade, P. Alves, L. Conceição, P. Novais, and G. Marreiros. A consensus-based group decision support system using a multi-agent microservices approach. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2098–2100, 2020. pages 104

[34] J. Carneiro, D. Martinho, G. Marreiros, A. Jimenez, and P. Novais. Dynamic argumentation in ubigdss. *Knowledge and Information Systems*, 55(3):633–669, 2018. pages 104

[35] J. Carneiro, D. Martinho, G. Marreiros, and P. Novais. Arguing with behavior influence: a model for web-based group decision support systems. *International Journal of Information Technology & Decision Making*, 18(02):517–553, 2019. pages 104

[36] J. Carneiro, P. Saraiva, L. Conceição, R. Santos, G. Marreiros, and P. Novais. Predicting satisfaction: perceived decision quality by decision-makers in web-based group decision support systems. *Neurocomputing*, 338:399–417, 2019. pages 104

[37] P. Casale, O. Pujol, and P. Radeva. Approximate convex hulls family for one-class classification. In *International workshop on multiple classifier systems*, pages 106–115. Springer, 2011. pages 172

[38] P. Casas, J. Mazel, and P. Owezarski. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. *Computer Communications*, 35(7):772–783, 2012. pages 19

[39] E. Castillo, D. Peteiro-Barral, B. G. Berdiñas, and O. Fontenla-Romero. Distributed one-class support vector machine. *International journal of neural systems*, 25(07):1550029, 2015. pages 17, 53

[40] T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli. The autoencoding variational autoencoder. *Advances in Neural Information Processing Systems*, 33:15077–15087, 2020. pages 34

[41] I. Cenni and P. Goethals. Negative hotel reviews on tripadvisor: A cross-linguistic analysis. *Discourse, Context & Media*, 16:22–30, 2017. pages 105

[42] R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019. pages 34

[43] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009. pages xxiii, 1, 2, 4, 5, 8, 30

[44] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 90–98. SIAM, 2017. pages 18

[45] K. Chen, S. Pashami, Y. Fan, and S. Nowaczyk. Predicting air compressor failures using long short term memory networks. In *EPIA Conference on Artificial Intelligence*, pages 596–609. Springer, 2019. pages 87, 89, 90

[46] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai. Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *IEEE Access*, 8:47072–47081, 2020. pages 113

[47] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016. pages 59

[48] X. Chen, H. Xie, G. Cheng, L. K. Poon, M. Leng, and F. L. Wang. Trends and features of the applications of natural language processing techniques for clinical trials text analysis. *Applied Sciences*, 10(6):2157, 2020. pages 104

[49] X. Chi, C. Yan, H. Wang, W. Rafique, and L. Qi. Amplified locality-sensitive hashing-based recommender systems with privacy protection. *Concurrency and Computation: Practice and Experience*, page e5681, 2020. pages 33

[50] J. K. Chow, Z. Su, J. Wu, P. S. Tan, X. Mao, and Y.-H. Wang. Anomaly detection of defects on concrete structures with the convolutional autoencoder. *Advanced Engineering Informatics*, 45:101105, 2020. pages 113

[51] C. Cimpanu. New hakai iot botnet takes aim at d-link, huawei, and realtek routers. *ZDNet. url: https://www. zdnet. com/article/new-hakai-iot-botnettakes-aim-at-d-link-huawei-and-realtek-routers/(visited on 05/02/2022)*, 2018. pages 163

[52] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006. pages 176

[53] N. Davari, B. Veloso, G. d. A. Costa, P. M. Pereira, R. P. Ribeiro, and J. Gama. A survey on data-driven predictive maintenance for the railway industry. *Sensors*, 21(17):5739, 2021. pages 87

[54] N. Davari, B. Veloso, R. P. Ribeiro, P. M. Pereira, and J. Gama. Predictive maintenance based on anomaly detection using deep learning for air production unit in the railway industry. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2021. pages 89, 90

[55] M. C. de Souza, B. M. Nogueira, R. G. Rossi, R. M. Marcacini, B. N. Dos Santos, and S. O. Rezende. A network-based positive and unlabeled learning approach for fake news detection. *Machine Learning*, 111(10):3549–3592, 2022. pages 8

[56] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006. pages 25, 51, 55, 75, 180

[57] L. Deng, D. Yu, et al. Deep learning: methods and applications. *Foundations and trends® in signal processing*, 7(3–4):197–387, 2014. pages 165

[58] Z. Ding and M. Fei. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17, 2013. pages 98

[59] Docs.zeek.org. Zeek user manual v3.2.0. https://docs.zeek.org/en/current/scripts/base/protocols/conn/main.zeek.html#base-protocols-conn-main-zeek, 2019. pages 164, 165

[60] D. Dua and E. Karra Taniskidou. Uci machine learning repository [http://archive. ics. uci. edu/ml]. irvine, ca: University of california. *School of Information and Computer Science*, 2017. pages 43

[61] C. Eiras-Franco, B. Guijarro-Berdiñas, A. Alonso-Betanzos, and A. Bahamonde. A scalable decision-tree-based method to explain interactions in dyadic data. *Decision Support Systems*, 127:113141, 2019. pages 31, 176

[62] C. Eiras-Franco, D. Martínez-Rego, B. Guijarro-Berdiñas, A. Alonso-Betanzos, and A. Bahamonde. Large scale anomaly detection in mixed numerical and categorical input spaces. *Information Sciences*, 487:115–127, 2019. pages 31, 34, 57, 71, 176

[63] S. Eltanbouly, M. Bashendy, N. AlNaimi, Z. Chkirbene, and A. Erbad. Machine learning techniques for network anomaly detection: A survey. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 156–162. IEEE, 2020. pages 7

[64] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli. Passban ids: An intelligent anomaly based intrusion detection system for iot edge devices. *IEEE Internet of Things Journal*, 2020. pages 67, 68

[65] G. Fenza, M. Gallo, and V. Loia. Drift-aware methodology for anomaly detection in smart grid. *IEEE Access*, 7:9645–9657, 2019. pages 113

[66] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70(3):447–489, 2019. pages 5

[67] D. Fernández-Francos, Ó. Fontenla-Romero, and A. Alonso-Betanzos. One-class convex hull-based algorithm for classification in distributed environments. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(2):386–396, 2017. pages xxvi, 17, 172, 173

[68] D. Fernández-Francos, D. Martínez-Rego, O. Fontenla-Romero, and A. Alonso-Betanzos. Automatic bearing fault diagnosis based on one-class ν-svm. *Computers & Industrial Engineering*, 64(1):357–365, 2013. pages 87

[69] E. W. T. Ferreira, G. A. Carrijo, R. de Oliveira, and N. V. de Souza Araujo. Intrusion detection system with wavelet and neural artifical network approach for networks computers. *IEEE Latin America Transactions*, 9(5):832–837, 2011. pages 63

[70] C. Finisterrae. Centre of supercomputing of galicia (cesga). *Science and Technology Infrastructures (in spanish ICTS), Tech. Rep*, 2012. pages 76

[71] E. Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985. pages 86, 169

[72] E. Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989. pages 104, 112

[73] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *International Conference on Parallel Problem Solving from Nature*, pages 584–593. Springer, 1996. pages 82

[74] R. Foorthuis. On the nature and types of anomalies: A review of deviations in data. *International Journal of Data Science and Analytics*, 12(4):297–331, 2021. pages xxi, 1, 3, 4

[75] J. Fruhlinger. The mirai botnet explained: How iot devices almost brought down the internet, 2018. pages 162

[76] E. Fumeo, L. Oneto, and D. Anguita. Condition based maintenance in railway transportation systems based on big data streaming analysis. *Procedia Computer Science*, 53:437–446, 2015. pages 88, 90

[77] S. Gao, Z.-Y. Li, M.-H. Yang, M.-M. Cheng, J. Han, and P. Torr. Large-scale unsupervised semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. pages 31

[78] A. B. Gardner, A. M. Krieger, G. Vachtsevanos, B. Litt, and L. P. Kaelbing. One-class novelty detection for seizure analysis from intracranial eeg. *Journal of Machine Learning Research*, 7(6), 2006. pages 17

[79] Z. Geng, Y. Zhang, and Y. Han. Joint entity and relation extraction model based on rich semantics. *Neurocomputing*, 429:132–140, 2021. pages 34

[80] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019. pages 34

[81] G. Giacinto, R. Perdisci, M. Del Rio, and F. Roli. Intrusion detection in computer networks by a modular ensemble of one-class classifiers. *Information Fusion*, 9(1):69–82, 2008. pages 17

[82] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016. pages 18

[83] D. Gong, L. Liu, V. Le, B. Saha, M. R. Mansour, S. Venkatesh, and A. v. d. Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019. pages 113

[84] M. T. Guerreiro, E. M. A. Guerreiro, T. M. Barchi, J. Biluca, T. A. Alves, Y. de Souza Tadano, F. Trojan, and H. V. Siqueira. Anomaly detection in automotive industry using clustering methods—a case study. *Applied Sciences*, 11(21):9868, 2021. pages 6

[85] Y. Han, Y. Lang, M. Cheng, Z. Geng, G. Chen, and T. Xia. Dtaxa: An actor–critic for automatic taxonomy induction. *Engineering Applications of Artificial Intelligence*, 106:104501, 2021. pages 34

[86] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. *Internet of Things*, 7:100059, 2019. pages 66, 68

[87] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998. pages 174

[88] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. pages 105, 112

[89] W. Hong, E. J. Hwang, J. H. Lee, J. Park, J. M. Goo, and C. M. Park. Deep learning for detecting pneumothorax on chest radiographs after needle biopsy: Clinical implementation. *Radiology*, page 211706, 2022. pages 34

[90] C.-Y. Hsu and W.-C. Liu. Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing. *Journal of Intelligent Manufacturing*, 32(3):823–836, 2021. pages 6

[91] X. Hu, Y. Han, and Z. Geng. A novel matrix completion model based on the multi-layer perceptron integrating kernel regularization. *IEEE Access*, 9:67042–67050, 2021. pages 34

[92] K.-W. Huang, G.-W. Chen, Z.-H. Huang, and S.-H. Lee. Anomaly detection in airport based on generative adversarial network for intelligent transportation system. In *2022 IEEE International Conference on Consumer Electronics-Taiwan*, pages 311–312. IEEE, 2022. pages 6

[93] C. Hutto and E. Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014. pages 105

[94] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998. pages 32, 177

[95] N. Japkowicz. *Concept learning in the absence of counterexamples: An autoassociation-based approach to classification*. Rutgers The State University of New Jersey-New Brunswick, 1999. pages 166

[96] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, pages 293–298, New York, New York, USA, 2001. ACM Press. pages 30

[97] H. John and S. Naaz. Credit card fraud detection using local outlier factor and isolation forest. *International Journal of Computational Science and Engineering*, 7(4):1060–1064, 2019. pages 112

[98] T. Jombart, S. Ghozzi, D. Schumacher, T. J. Taylor, Q. J. Leclerc, M. Jit, S. Flasche, F. Greaves, T. Ward, R. M. Eggo, et al. Real-time monitoring

of covid-19 dynamics using automated trend fitting and anomaly detection. *Philosophical Transactions of the Royal Society B*, 376(1829):20200266, 2021. pages 113

[99] I. Kalathas and M. Papoutsidakis. Predictive maintenance using machine learning and data mining: A pioneer method implemented to greek railways. *Designs*, 5(1):5, 2021. pages 87, 89, 90

[100] P. Kamat and R. Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S Web of Conferences*, volume 170, page 02007. EDP Sciences, 2020. pages 86

[101] S. Kang, S. Sristi, J. Karachiwala, and Y. Hu. Detection of anomaly in train speed for intelligent railway systems. *Int. Conf. Control. Autom. Diagnosis, ICCAD*, 2018. pages 87, 89, 90

[102] Kaspersky. What is a trojan? - definition and explanation, 2021. pages 164

[103] M. E. B. H. Kbaier, H. Masri, and S. Krichen. A personalized hybrid tourism recommender system. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 244–250. IEEE, 2017. pages 106

[104] S. S. Khan and M. G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014. pages 17

[105] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019. pages 7

[106] G. Kim, S. Lee, and S. Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4):1690–1700, 2014. pages 67

[107] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982. pages 167

[108] S. A. Kokatnoor and B. Krishnan. Twitter hate speech detection using stacked weighted ensemble (swe) model. In *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 87–92. IEEE, 2020. pages 9

[109] A. Koons-Stapf. Condition based maintenance: Theory, methodology, & application. 01 2015. pages 87

[110] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Loop: local outlier probabilities. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1649–1652. ACM, 2009. pages 30

[111] H.-P. Kriegel, P. Kröger, and A. Zimek. Outlier detection techniques. *Tutorial at KDD*, 10, 2010. pages 30

[112] A. D. Landress. A hybrid approach to reducing the false positive rate in unsupervised machine learning intrusion detection. In *SoutheastCon 2016*, pages 1–6. IEEE, 2016. pages 67

[113] W.-j. Lee. Anomaly detection and severity prediction of air leakage in train braking pipes. *International Journal of Prognostics and Health Management*, 21, 2017. pages 87, 88, 90

[114] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur. Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C: Emerging Technologies*, 45:17–26, 2014. pages 87, 88, 90

[115] Z. Li, J. Tang, L. Zhang, and J. Yang. Weakly-supervised semantic guided hashing for social image retrieval. *International Journal of Computer Vision*, 128(8):2265–2278, 2020. pages 33

[116] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich. A survey on anomaly detection for technical systems using lstm networks. *Computers in Industry*, 131:103498, 2021. pages 2

[117] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008. pages 105

[118] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):3, 2012. pages 33, 171, 172

[119] H. Liu, F. Hussain, C. L. Tan, and M. Dash. Discretization: An enabling technique. *Data mining and knowledge discovery*, 6(4):393–423, 2002. pages 21

[120] Z. Liu, J. Liu, C. Pan, and G. Wang. A novel geometric approach to binary classification based on scaled convex hulls. *IEEE transactions on neural networks*, 20(7):1215–1220, 2009. pages 172

[121] R. Logesh, V. Subramaniyaswamy, V. Vijayakumar, and X. Li. Efficient user profiling based intelligent travel recommender system for individual and group of users. *Mobile Networks and Applications*, 24(3):1018–1033, 2019. pages 106

[122] S. Loria et al. textblob documentation. *Release 0.15*, 2(8), 2018. pages 105

[123] G. Manco, E. Ritacco, P. Rullo, L. Gallucci, W. Astill, D. Kimber, and M. Antonelli. Fault detection and explanation through big data analysis on sensor streams. *Expert Syst. Appl.*, 87:141–156, nov 2017. pages 87, 88, 90

[124] L. M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001. pages 17

[125] E. Manzoor, H. Lamba, and L. Akoglu. xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1963–1972, 2018. pages 98

[126] D. Martínez-Rego, D. Fernández-Francos, O. Fontenla-Romero, and A. Alonso-Betanzos. Stream change detection via passive-aggressive classification and Bernoulli CUSUM. *Information Sciences*, 305:130–145, 2015. pages 34, 53, 73, 176

[127] D. Martínez-Rego, O. Fontenla-Romero, A. Alonso-Betanzos, and J. C. Principe. Fault detection via recurrence time statistics and one-class classification. *Pattern Recognition Letters*, 84:8–14, 2016. pages 87

[128] D. Martinus and J. Tax. One-class classification: Concept-learning in the absence of counterexamples. *Delft University of Technology*, 2001. pages 166

[129] O. Mazhelis. One-class classifiers: a review and analysis of suitability in the context of mobile-masquerader detection. *South African Computer Journal*, 2006(36):29–48, 2006. pages 166

[130] P. McBurney and S. Parsons. Dialogue games for agent argumentation. In *Argumentation in artificial intelligence*, pages 261–280. Springer, 2009. pages 104

[131] M. K. McNutt, R. Camilli, T. J. Crone, G. D. Guthrie, P. A. Hsieh, T. B. Ryerson, O. Savas, and F. Shaffer. Review of flow rate estimates of the deepwater horizon oil spill. *Proceedings of the National Academy of Sciences*, 109(50):20260–20267, 2012. pages 7

[132] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018. pages 66, 68

[133] J. Meira, R. Andrade, I. Praça, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos, and G. Marreiros. Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. *Journal of Ambient Intelligence and Humanized Computing*, 11(11):4477–4489, 2020. pages 16, 17, 113

[134] J. Meira, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos, P. Novais, and G. Marreiros. Anomaly detection on natural language processing to improve predictions on tourist preferences. *Electronics*, 11(5):779, 2022. pages 9, 104

[135] J. Meira, C. Eiras-Franco, V. Bolón-Canedo, G. Marreiros, and A. Alonso-Betanzos. Fast anomaly detection with locality-sensitive hashing and hyperparameter autotuning. *Information Sciences*, 607:1245–1264, 2022. pages 29, 65, 71, 72, 177

[136] J. Meira, B. Veloso, V. Bólon-Canedo, M. Goreti, and A. Alonso-Betanzos. Data-driven predictive maintenance framework for railway systems. *Intelligent Data Analysis, In Press*, 2022. pages 7, 85

[137] T. Micro. "hide 'n seek" botnet uses peer-to-peer infrastructure to compromise iot devices, 2018. pages 162

[138] J. Montiel, J. Read, A. Bifet, and T. Abdessalem. Scikit-multiflow: A multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914, 2018. pages 97

[139] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 military communications and information systems conference (MilCIS)*, pages 1–6. IEEE, 2015. pages 66, 68

[140] A.-H. Muna, N. Moustafa, and E. Sitnikova. Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*, 41:1–11, 2018. pages 66, 68

[141] K. Naidoo and V. Marivate. Unsupervised anomaly detection of healthcare providers using generative adversarial networks. *Responsible Design, Implementation and Use of Information and Communication Technology*, 12066:419, 2020. pages 113

[142] U. Naseem, I. Razzak, and P. W. Eklund. A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter. *Multimedia Tools and Applications*, 80(28):35239–35266, 2021. pages 9

[143] M. Nilashi, O. Ibrahim, E. Yadegaridehkordi, S. Samad, E. Akbari, and A. Alizadeh. Travelers decision making using online review in social network sites: A case on tripadvisor. *Journal of computational science*, 28:168–179, 2018. pages 105

[144] K. Noto, C. Brodley, and D. Slonim. Frac: a feature-modeling approach for semi-supervised and unsupervised anomaly detection. *Data mining and knowledge discovery*, 25(1):109–133, 2012. pages 19

[145] H. Om and A. Kundu. A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. In *2012 1st International Conference on*

Recent Advances in Information Technology (RAIT), pages 131–136. IEEE, 2012. pages 67

[146] C. O'Reilly, A. Gluhak, M. A. Imran, and S. Rajasegarar. Anomaly detection in wireless sensor networks in a non-stationary environment. *IEEE Communications Surveys & Tutorials*, 16(3):1413–1432, 2014. pages 67

[147] C. Osborne. Meet torii, a new iot botnet far more sophisticated than mirai variants. *URL:" https://www. zdnet. com/article/meet-torii-a-new-iot-botnet-far-more-sophisticated-than-mirai*, 2018. pages 163

[148] P. Paganini. Mirai okiru botnet targets for first time ever in the history arc-based iot devices, 2018. pages 164

[149] M.-O. Pahl and F.-X. Aubet. All eyes on you: Distributed multi-dimensional iot microservice anomaly detection. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 72–80. IEEE, 2018. pages 66, 68

[150] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021. pages 5

[151] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)*, pages 315–326. IEEE, 2003. pages 30, 34

[152] T. J. Park, N. Kanda, D. Dimitriadis, K. J. Han, S. Watanabe, and S. Narayanan. A review of speaker diarization: Recent advances with deep learning. *Computer Speech & Language*, 72:101317, 2022. pages 34

[153] A. Parmisano, S. Garcia, and M. J. Erquiaga. Stratosphere laboratory. aposemat iot-23. a labeled dataset with malicious and benign iot network traffic., 2020. pages xxii, 58, 65, 68, 161, 162

[154] L. Parsons, E. Haque, and H. Liu. Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105, 2004. pages 19

[155] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23-24):2435–2463, 1999. pages 64

[156] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. pages 98

[157] P. Pereira, R. P. Ribeiro, and J. Gama. Failure prediction – an application in the railway industry. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 8777:264–275, 2014. pages 87, 88, 90

[158] M. R. Pillutla, N. Raval, P. Bansal, K. Srinathan, and C. Jawahar. Lsh based outlier detection and its application in distributed setting. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2289–2292. ACM, 2011. pages 35, 36, 53

[159] T. Pourhabibi, K.-L. Ong, B. H. Kam, and Y. L. Boo. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133:113303, 2020. pages 113

[160] K. Qin, Q. Wang, B. Lu, H. Sun, and P. Shu. Flight anomaly detection via a deep hybrid model. *Aerospace*, 9(6):329, 2022. pages 6

[161] J. R. Quinlan. Probabilistic decision trees. In *Machine Learning*, pages 140–152. Elsevier, 1990. pages 104, 112

[162] J. Rabatel, S. Bringay, and P. Poncelet. Anomaly detection in monitoring sensor data for preventive maintenance. *Expert Syst. Appl.*, 38(6):7003–7015, 2011. pages 87, 88, 90

[163] Radware. The rise of the botnets: Mirai & hajime, 2017. pages 163

[164] D. Rahmawati and R. Sarno. Anomaly detection using control flow pattern and fuzzy regression in port container handling. *Journal of King Saud University-Computer and Information Sciences*, 33(1):11–20, 2021. pages 6

[165] R. P. Ribeiro, P. Pereira, and J. Gama. Sequential anomalies: a study in the railway industry. *Machine Learning*, 105(1):127–153, 2016. pages 87, 88, 90

[166] M. Roesch et al. Snort: Lightweight intrusion detection for networks. In *Lisa*, volume 99, pages 229–238, 1999. pages 64

[167] N. Rtayli and N. Enneya. Enhanced credit card fraud detection based on svm-recursive feature elimination and hyper-parameters optimization. *Journal of Information Security and Applications*, 55:102596, 2020. pages 112

[168] J.-R. Ruiz-Sarmiento, J. Monroy, F.-A. Moreno, C. Galindo, J.-M. Bonelo, and J. Gonzalez-Jimenez. A predictive model for the maintenance of industrial machinery in the context of industry 4.0. *Engineering Applications of Artificial Intelligence*, 87:103289, 2020. pages 112

[169] G. Salierno, S. Morvillo, L. Leonardi, and G. Cabri. An architecture for predictive maintenance of railway points based on big data analytics. In *International Conference on Advanced Information Systems Engineering*, pages 29–40. Springer, 2020. pages 87, 89, 90

[170] K. K. Santhosh, D. P. Dogra, and P. P. Roy. Anomaly detection in road traffic using visual surveillance: A survey. *ACM Computing Surveys (CSUR)*, 53(6):1–26, 2020. pages 113

[171] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001. pages 33, 34

[172] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. Support vector method for novelty detection. *Advances in neural information processing systems*, 12, 1999. pages 174, 175

[173] T. Security. Hakai botnet shows signs of intense activity in latin america, 2018. pages 163

[174] H. J. Shin, D.-H. Eom, and S.-S. Kim. One-class support vector machines—an application in machine fault detection and classification. *Computers & Industrial Engineering*, 48(2):395–408, 2005. pages 17

[175] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374, 2012. pages 17, 19

[176] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi. A deep learning approach to network intrusion detection. *IEEE transactions on emerging topics in computational intelligence*, 2(1):41–50, 2018. pages 113

[177] A. Smets, J. Vannieuwenhuyze, and P. Ballon. Serendipity in the city: User evaluations of urban recommender systems. *Journal of the Association for Information Science and Technology*, 73(1):19–30, 2022. pages 106

[178] A. Smiti. A critical overview of outlier detection methods. *Computer Science Review*, 38:100306, 2020. pages 33

[179] R. M. Souza, E. G. Nascimento, U. A. Miranda, W. J. Silva, and H. A. Lepikson. Deep learning for diagnosis and classification of faults in industrial rotating machinery. *Computers & Industrial Engineering*, 153:107060, 2021. pages 112

[180] F. K. Sufi and M. Alsulami. Automated multidimensional analysis of global events with entity detection, sentiment analysis and anomaly detection. *IEEE Access*, 9:152449–152460, 2021. pages 9

[181] S. Sun, C. Luo, and J. Chen. A review of natural language processing techniques for opinion mining systems. *Information fusion*, 36:10–25, 2017. pages 104

[182] S. C. Tan, K. M. Ting, and T. F. Liu. Fast anomaly detection for streaming data. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011. pages 86, 98, 167, 168

[183] M. Taneja. An analytics framework to detect compromised iot devices using mobility behavior. In *2013 International Conference on ICT Convergence (ICTC)*, pages 38–43. IEEE, 2013. pages 67

[184] J. Tang, Z. Chen, A. W. Fu, and D. W. Cheung. Capabilities of outlier detection schemes in large datasets, framework and methodologies. *Knowledge and Information Systems*, 11(1):45–84, dec 2006. pages 30

[185] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. Ieee, 2009. pages 17, 19, 67, 68, 160

[186] D. M. J. Tax. One-class classification: Concept learning in the absence of counter-examples. 2002. pages 86, 105, 113, 169

[187] J. Teich. Pareto-front exploration with uncertain objectives. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 314–328. Springer, 2001. pages 60, 82

[188] J. Thanaki. *Python natural language processing*. Packt Publishing Ltd, 2017. pages 104

[189] M. Thimm. Strategic argumentation in multi-agent systems. *KI-Künstliche Intelligenz*, 28(3):159–168, 2014. pages 104

[190] S. Thudumu, P. Branch, J. Jin, and J. J. Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1):1–30, 2020. pages 5

[191] T. Tieleman, G. Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. pages 18

[192] A. Valdivia, M. V. Luzón, and F. Herrera. Sentiment analysis in tripadvisor. *IEEE Intelligent Systems*, 32(4):72–77, 2017. pages 105

[193] N. T. Van, T. N. Thinh, et al. An anomaly-based network intrusion detection system using deep learning. In *2017 international conference on system science and engineering (ICSSE)*, pages 210–214. IEEE, 2017. pages 113

[194] M. Vicente, B. Gelera, A. Remilliano, C. Toyama, and J. Urbanec. Bashlite updated with mining and backdoor commands, 2019. pages 164

[195] R. Vinayakumar, K. Soman, and P. Poornachandran. Applying convolutional neural network for network intrusion detection. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1222–1228. IEEE, 2017. pages 113

[196] L. Vu, L. Cao, Q. Nguyen, D. Nguyen, H. Dinh, and E. Dutkiewicz. Learning latent representation for iot anomaly detection. *IEEE Transactions on Cybernetics*, 2020. pages 66, 68

[197] R. S. Wahono, N. S. Herman, and S. Ahmad. A comparison framework of classification models for software defect prediction. *Advanced Science Letters*, 20(10-11):1945–1950, 2014. pages 180

[198] J. Wang, Y. Tang, S. He, C. Zhao, P. K. Sharma, O. Alfarraj, and A. Tolba. Logevent2vec: Logevent-to-vector based anomaly detection for large-scale logs in internet of things. *Sensors*, 20(9):2451, 2020. pages 8

[199] Y. Wang, S. Parthasarathy, and S. Tatikonda. Locality Sensitive Outlier Detection: A ranking driven approach. In *2011 IEEE 27th International Conference on Data Engineering*, pages 410–421. IEEE, apr 2011. pages 35, 36

[200] R. Wetzig, A. Gulenko, and F. Schmidt. Unsupervised anomaly alerting for iot-gateway monitoring using adaptive thresholds and half-space trees. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 161–168. IEEE, 2019. pages xxvi, 168

[201] R. E. Wright. Logistic regression. 1995. pages 104, 112

[202] J. Yan, Y. Meng, L. Lu, and L. Li. Industrial big data in an industry 4.0 environment: Challenges, schemes, and applications for predictive maintenance. *IEEE Access*, 5:23484–23491, 2017. pages 86

[203] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, pages 1–7, 2015. pages 64

[204] M. Yuan, N. Boston-Fisher, Y. Luo, A. Verma, and D. L. Buckeridge. A systematic review of aberration detection algorithms used in public health surveillance. *Journal of biomedical informatics*, 94:103181, 2019. pages 113

[205] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Presented as part of*

the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12), pages 15–28, 2012. pages 37, 67

[206] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga. A survey of intrusion detection in internet of things. *Journal of Network and Computer Applications*, 84:25–37, 2017. pages 64

[207] S. Zavrak and M. İskefiyeli. Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access*, 8:108346–108358, 2020. pages 7

[208] X. Zhang, M. Salehi, C. Leckie, Y. Luo, Q. He, R. Zhou, and R. Kotagiri. Density biased sampling with locality sensitive hashing for outlier detection. In *International Conference on Web Information Systems Engineering*, pages 269–284. Springer, 2018. pages 35, 36

[209] C. Zheng, J. Yang, and A. Davila. Muhstik botnet attacks tomato routers to harvest new iot devices, 2020. pages 163

Appendix A

Methods and Materials

This appendix describes the datasets, AD methods and evaluation metrics used in this thesis.

A.1. Datasets

A.1.1. NSL-KDD dataset

In 1999, in the third international competition in the conference KDD, the KDD 99 dataset (UCI Machine Learning Repository 2015)[1] was presented to the scientific community. This dataset is frequently used in the literature of IDS evaluation, and contains simulated network activity samples, corresponding to normal and abnormal activity divided in five categories:

- **DoS**: An intruder tries to make a service unavailable (contains 9 types of DoS attacks);
- **Remote to Local (R2L)**: An intruder tries to obtain remote access to the victim's machine (contains 15 types of R2L attacks);
- **User to Root (U2R)**: An intruder with physical access to the victim's machine tries to gain super-user privileges (contains 8 types of U2R attacks);

[1] https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

- **Probe**: An intruder tries to get information about a victim's machine (contains 6 types of Probe attacks);
- **Normal**: It constitutes the normal operations or activities in the network

Tavallaee et al. [185] made some improvements to KDD 99 dataset and the result was the NSL-KDD dataset. This dataset is already organized into two subsets: one to train the algorithms, and another one to test them. Each data sample contains 43 features where four of which are nominal type, six are binary and the rest of them are numerical type.

A.1.2. ISCX dataset

ISCX is a dataset developed by Shiravi et al. (Shiravi et al. 2012) at the Canadian cybersecurity institute. This dataset is based on the concept of profiles that contain detailed descriptions of abstract intrusions and distributions for applications, protocols, services, and low-level network entities. To create this dataset, real network communications were analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3 and FTP protocols. In this regard, a set of guidelines have been established to delineate a valid dataset that establishes the basis for profiling. These guidelines are vital to the effectiveness of the dataset in terms of realism, total capture, integrity, and malicious activity (Shiravi et al. 2012). Each data sample in the ISCX dataset contains 21 attributes. There is a total of 7 days of network traffic captured with 4 different attack types shown in Table A.1.2.

A.1.3. IOT-23 DATASET

The IoT-23 dataset consists of 23 IoT network traffic capture in the form of pcap files. From these, 20 are scenarios corresponding to infected IoT devices, and three represent benign ones. The benign come from an Amazon Echo home intelligent personal assistant, a Somfy smart doorlock and a Philips HUE smart led lamp. The authors used a Raspberry Pi to execute specific malware samples with several protocols, performing different actions in each malicious scenario. All 23 scenarios

Table A.1: ISCX captured activity. The attacks were captured along with normal network activity. To distinguish between a normal observation and an abnormal one it is presented in the ISCX dataset an attribute called "label" where value 1 represents an attack and value 0 represents normal activity

Capturing date	Network activity
11/06/2010	Normal
12/06/2010	Normal
13/06/2010	Normal + Internal infiltration into the network
14/06/2010	Normal + HTTP Denial of Service
15/06/2010	Normal + Distributed Denial of Service using a Botnet IRC
16/06/2010	Normal
17/06/2010	Normal + Brute Force SSH

were carried out in a controlled network environment with an unrestricted internet connection, simulating a standard environment for a real IoT device.

Regarding all the files available, we have used the **conn.log.labeled** files from each scenario. The Zeek network analyser generated these files through the original pcap file, which contains the flows of the capture network connection as a normal Zeek **conn.log** file. Nevertheless, it also contains two new columns for the labels. A python script added these columns that compared this data with rules to validate if the flow data fit the labelling criteria. We ignored the three benign scenarios in our work since these have only a few data flows (1983 flows in all three sample scenarios). Each of the other malicious scenarios already has an acceptable quantity of benign flows. In this way, we aim to understand whether each algorithm can differentiate normal from abnormal network behaviour in each scenario.

According to the authors of [153], they captured long periods of running malware by rotating pcap files every 24 hours, due to the large size of the traffic generated. However, in some cases, the authors had to stop the capture because of the rapid growth of the pcap file size. As can be seen in Table A.2, some of the captures differ significantly in the number of hours recorded.

In each scenario, the authors capture a certain type of intrusion:

- **HNS** - It compromises machines through a worm-like technique that generates a random list of IP addresses as potential victims. HNS can execute various

Table A.2: IoT-23 dataset malicious scenarios. See complete table information at [153]

ID SubSet	ZeekFlows	Benign %	#Malware %	Pcap Size	Malware
1	1,008,749	46.52	53.48	140 MB	Hide and Seek
3	156,104	2.91	97.09	56 MB	Muhstik
7	11,454,723	0.66	99.34	897 MB	Linux.Mirai
8	10,404	20.95	79.05	2.1 MB	Linux.Hajime
9	6,378,294	0.35	99.65	472 MB	Hakai
17	54,659,864	0.06	99.94	7.8 GB	Kenjiro
20	3,210	99.5	0.5	3.9 MB	Torii
21	3,287	99.57	0.43	3.9 MB	Torii
33	54,454,592	2.54	97.46	3.9 GB	Kenjiro
34	23,146	8.3	91.7	121 MB	Mirai
35	10,447,796	79.08	20.92	3.6G	Mirai
36	13,645,107	0.02	99.98	992 MB	Okiru
39	73,568,982	0.01	99.99	5.3GB	IRCBot
42	4,426	99.86	0.14	2.8 MB	Trojan
43	67,321,810	30.56	69.44	6 GB	Mirai
44	237	89.79	10.21	1.7 GB	Mirai
48	3,394,347	0.11	99.89	1.2G	Mirai
49	5,410,562	0.07	99.93	1.3 GB	Mirai
52	19,781,379	0.01	99.99	4.6 GB	Mirai
60	3,581,029	0.07	99.93	21 GB	Gagfyt

commands, similar to a P2P protocol interfering with the device's operation. It comes with several anti-tampering techniques to prevent others from kidnapping or to poison the victim's device. It can also target devices through web exploit capabilities. Unlike other IoT botnets, HNS does not have a DDoS function. However, it adds cyber espionage features to the botnet via a file theft component[137];

- **Mirai** - Contains a limited dictionary of 61 username and password combinations to be brute-forced in the Telnet protocol. It employs a simple, clear-text TCP-based protocol on port 23 for C&C communications, and omits domains or Domain Generation Algorithms (DGA) to protect its C&C from being discovered and quickly blocklisted. It is a simple botnet attack, yet lethal. On 21 October 2016, Mirai botnet launched the largest-ever DDoS attacks against DNS provider Dyn, causing websites like Twitter, Paypal and Amazon to be unreachable for several hours across Europe and the US [75];

- **Hajime** - It leverages the infection method and brute force exploits of Mirai however is capable of updating itself. Hajime can efficiently and quickly expand its member bots with richer functionality. The distributed bot network used for C&C and updating uses trackerless torrents through the well known public BitTorrent P2P network, employing several dynamic hashes that change daily to better conceal C&C activity [163];

- **IRC BOT** - It was the first built botnet. It uses an IRC-based chat channel to perform automated tasks. It is usually configured as a channel operator and enables the performance of several channel-specific tasks on behalf of the user. They are accountable for many cyber attacks, from DDoS and spam attacks to click fraud and Keylogging [19];

- **Muhstik** - It is a botnet variant with a self-propagating worm-like capability to infect Linux servers and IoT devices. It uses multiple exploits taking advantage of vulnerabilities in web servers and platforms such as Weblogic, WordPress and Drupal. Muhstik usually launches DDoS attacks and cryptocurrency mining in IoT bots, allowing the attacker to earn profits [209];

- **Hakai** - It is distributed to vulnerable devices through brute-force attacks or 0-day exploits as Mirai-variant. Hakai is used to perform HTTP, UDP, TCP and STD flood DDoS attacks. Hakai has risen tremendously, showing indication of extreme activity in Latin America [173]. Moreover, the Hakai codebase has made it into the hands of other groups, originating two different Hakai-based variants named **Kenjiro** and **Izuku** [51];

- **Torii** - It differs from other well-known botnets, especially in its advanced techniques. Torii tries to be more persistent and stealthy once the device is compromised. It comes with a set of features for extracting sensitive information. It has a modular architecture capable of executing and searching commands via multiple layers of encrypted transmission. In addition, Torii supports a wide range of target architectures, including PowerPC, SuperH, ARM, MIPS, x86, x64 and others. [147].

- **Gagfyt** - It is considered one of the progenitor IoT botnets, initially spread by exploiting Shellshock vulnerabilities in Busybox on various devices in 2015. After infection, Gagfyt uses Telnet to perform reconnaissance and propagate.

Gagfyt commands can target embedded systems and, in some circumstances, bypass DDoS mitigation services. It also has backdoor capabilities and cryptocurrency mining [194].

- **Trojan** - It is a malware disguised as legitimate software. It can infect devices with phishing or other social engineering attack techniques, for instance, by opening an infected email attachment or clicking on a link to a malicious website. It tricks users into installing the malware working behind the scenes to achieve, mostly, access and control over a particular device or machine. [102].
- **Okiru** - It is another Mirai variant reaching billions of devices due to its specific targeting of ARC processors. Okiru's telnet attack login information is slightly longer than Mirai, it supports up to 114 credentials and its configuration is encrypted in two parts with encrypted telnet bombardment passwords. [148].

Regarding the data characteristics, the **conn.log.labeled** files used in our experiments have 23 features, these being:

- **ts** - Time of the first packet
- **uid** - Unique identifier of the connection.
- **id.orig_h, id.orig_p, id.resp_h, id.resp_p** - The Connection's 4-tuple of endpoint addresses/ports.
- **proto** - The transport layer protocol of the connection;
- **duration** - The connection duration. For tear-downs of 3 or 4-way connections, not including the final ACK;
- **service** - An identification of an application protocol being sent over the connection;
- **orig_bytes** - The number of payload bytes the originator sent;
- **resp_bytes** - The number of payload bytes the responder sent;
- **conn_state** - Several connection status, see in [59];

- **local_orig** - **T** If the connection is originated locally; **F** If it was originated remotely. Empty if is undefined;
- **local_resp** - The same as local_orig but instead of representing originated connection it represents connection responded from;
- **missed_bytes** - Indicates the number of bytes missed in content gaps, which is representative of packet loss;
- **orig_pkts** - Number of packets that the originator sent;
- **orig_ip_bytes** - Number of IP level bytes that the originator sent;
- **resp_pkts** - Number of packets that the responder sent;
- **resp_ip_bytes** - Number of IP level bytes that the responder sent;
- **history** - Records the state history of connections as a string of letters. The meaning of those letters can be seen in [59];
- **tunnel_parents** - if this connection was through a tunnel, it indicates the uid values for any encapsulated parent connections used during the lifetime of this internal connection..
- **label** - The class of the record. **Benign** activity in the network or **malicious**. If it is malicious indicates the name of the malware;
- **detailed-label** - Detailed description of the malicious activity.

A.2. Methods

A.2.1. Autoencoders

An Autoencoder is a neural network that is part of a sub-area of machine learning called Deep learning (sets of algorithms with several layers of processing which are used to model high-level abstractions of data [57]). Neural Networks are interconnected processing units that are organized by one or more layers which can be used in the implementation of a complex functional mapping between input and output

variables [20]. They can perform linear or non-linear transformations through the processing of the units in the different layers [129].

The autoencoder, also known as autoassociator, is a kind of neural network that is trained to make the input features the same or very similar to the output features [95]. In the classification task, the autoencoder can reproduce accurately only the vectors whose structure is similar to the structure learned by the neural network.

As a neural network, the autoencoders are sensitive to outliers since they contribute to the minimization of the error function. The disadvantage of this method is the need of employing a number of parameters that have to be specified by the user [128]. These parameters consist of selecting a number of hidden layers, the number of hidden units in each layer, the type of transformation function, the learning rate and the stopping rule. In addition to these parameters, it is necessary to estimate the number of weights (usually equal to the number of hidden units and input units) for the training set. A large amount of data is essential for an accurate estimation of weights. The computational resources for this algorithm are considerably high, since the learning process is iterative, being repeated several times throughout the training set until the stop rule is satisfied. For the application of this algorithm, the H2O package[2] was used. This package is an open-source mathematical engine for big data processing machine learning algorithms, such as generalized linear models, gradient boosting, Random Forests and neural networks (deep learning) in several cluster environments.

After loading the datasets (NSL-KDD and ISCX) already pre-processed to this engine, the h2o.deeplearning function was used to train the autoencoder algorithm. Regarding the parameters to be used, we tested several, but we find that Glander's3 approach, applying a bottleneck architecture used to fraud detection in credit card transactions, presented better results. When tuning the autoencoder, we used a separated validation set, and we found that using a bottleneck architecture, where the number of neurons in the middle layer is lower than the first and last layers, presented better classification results.

[2]https://cran.r-project.org/web/packages/h2o/index.html

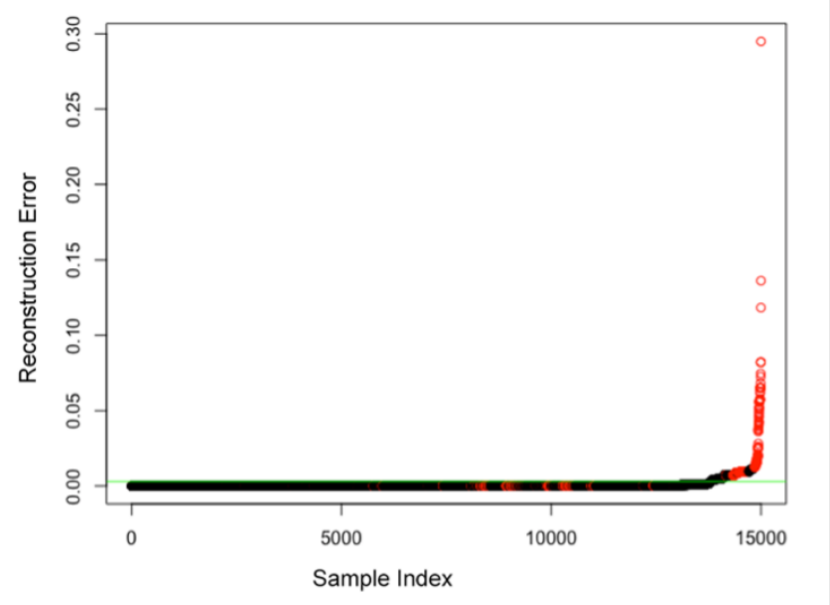


Figure A.1: Reconstruction of the mean square error.

A.2.2. Half Space Trees

HS-Trees is a one-class anomaly detector algorithm built to deal with data streaming environments [182]. *HS-Trees* is an ensemble that learns incrementally as the data arrives, capable of performing unsupervised learning in evolving data. The basic concept of this algorithm is to create binary trees by partitioning data into sub-spaces or regions. When generating a tree, the algorithm selects a random dimension and splits it into disjoint, equal-volume halves, thus creating the left and right side child nodes. This process is repeated until the tree reaches a maximum depth (user-specified parameter). Therefore, any data point in the domain travels a single path from the tree's root to the leaves going through the different sub-spaces. Although the data is splitted into density sub-spaces, this algorithm differs from clustering algorithms such as SOM [107] since *HS-Trees* output are scores of anomalous values estimated by the density regions, SOM computes clusters regions through distance based-techniques. The *HS-trees* captures the mass in each node, representing the number of data points, and uses it to profile each data point's anomaly estimation.

The data-stream is partitioned into windows of equal sizes, named *reference*

window and *latest window*. In *reference window* the algorithm learns the mass profile (r value stored in each node) to infer the anomaly scores of new data points arriving in the *latest window* [182]. According to the mass profile values, a data point is considered normal for high mass regions, while anomalous data points fall into low mass regions. Also, the *latest window* (l value stored in each node) records the mass profile, and once this window is full, that is, after a set of data points are recorded in the new mass profile, the *latest window*, overrides the old mass profile and becomes the *reference window*. Thus *reference window* always keeps the latest mass profile, used to evaluate the new data that arrives in the stream. This process repeats until the end of the stream.

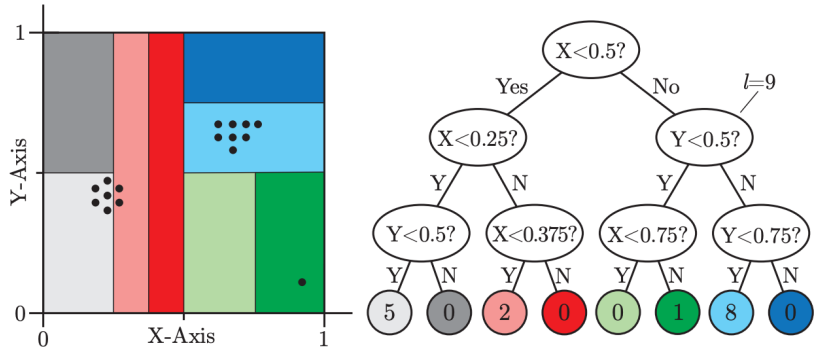


Figure A.2: HS-trees example by [200] and a recorded latest mass profile. The left image represents the data partitioned, and the right image the HS-tree generated.

Figure A.2 shows a representation of a *HS-Trees*. The HS-Trees partitions the domain dimensions with range values [0,1] on the left side, where dots represent data points. The right side of Figure A.2 shows an HS-tree with a three-depth level. The inner nodes contain information regarding the splitting values and which dimension was partitioned, leaves contain mass profile values of the latest window. Also, it is possible to observe the mass profile value stored in the root's right child ($l = 9$). This value is incremented by one when a newly arrived data point traverses it on its path to a leaf.

To make the algorithm more robust, $t \in \mathbb{N} > 0$ HS-trees are created and combined to form an ensemble algorithm, reducing the spread or dispersion of predictions, thus, improving performance. Also, a random perturbation is assigned to each feature space when split in the tree's root, providing diversity to the algorithm.

Each tree in the ensemble computes a score for each data point independently regarding the anomaly score. The computed score allows the new arrival data point to traverse through the tree's nodes until it reaches the leaves or a node with a mass profile equal to or less than a user-determined value s denoted by the size limit (minimum mass required in a node). The final anomaly computed score is the sum of all anomaly scores of the HS-trees.

A.2.3. One-Class K Nearest Neighbour

The OCKNN [186] is an adaptation of the original K-Nearest Neighbour algorithm [71] for supervised learning using distances between neighbours to classify the data. The K in OCKNN is the number of nearest neighbours, the core deciding factor. This algorithm is a lazy learning algorithm since it does not learn in the training phase, where all data points are used at the time of prediction. This algorithm only stores or memorises the data points in the training phase and waits until classification is performed. It is named One-Class since it requires only one class in the training phase. After memorising all data points, in the prediction phase, the algorithm uses a user-specified distance metric such as euclidean, Manhattan, Chebyshev, Minkowski, Minkowski or Mahalanobis to compute distances between points in the domain space. The mean distance between the new data point and its k nearest neighbours from the training set is calculated.

The mean distance value of each data point to its k neighbours represents the anomaly score. An anomalous classification is made if a distance value for a specific data point is more significant than a $d \in \mathbb{R} > 0$ distance threshold (user-specified parameter).

It can be observed in Figure A.3 how the OCKNN algorithm works for one neighbour ($K = 1$). The black dots within the drawn area represent the normal data points. The stars (A and C) are data points classified as anomalous since distance $d1$, and $d3$ are more significant than the distance threshold dx, while data point B is classified as normal as distance $d2$ is minor than dx.

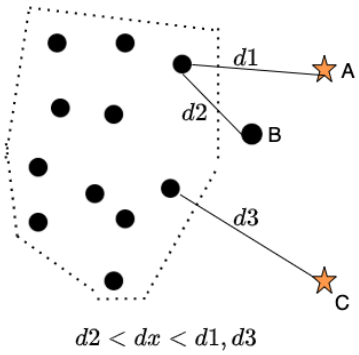


Figure A.3: OCKNN illustration example in two-dimensional space, where $k = 1$; $d1, d2, d3$ are distances of points A, B and C respectively, to their nearest neighbour; dx is the distance threshold to consider a given data point to be anomalous.

A.2.4. One-Class K-Means

The K-Means is a clustering algorithm, which is the process of partitioning a set of data (or objects) into smaller subclasses with common characteristics. The number of clusters is defined initially and remains fixed throughout the process. The goal of this algorithm is to find different groups in data defined by the variable k. Based on the data features, this algorithm works iteratively to assign each observation or object to one of the k groups. The algorithms start by estimating the k centroids that are the centers of each cluster. In this step, each object is assigned to its nearest centroid, based on the squared Euclidean distance. As shown in equation A.1, being c_i the collection of centroids in set C, each object x is assigned to a cluster based on where $dist(\cdot)$ is the standard Euclidean distance.

$$argmin_{c_i \in c} \; dist(c_i, x)^2 \tag{A.1}$$

Then subsequently the centroids are recomputed. This process is done by taking the mean of all objects assigned to that centroid's cluster. In equation A.2 the set of data point assignments for each i^{th} cluster centroid is S_i.

$$c_i = \frac{1}{|S_i|} \sum_{x_i \in s_i} x_i \tag{A.2}$$

To apply this algorithm as a one-class classification, the building process of clusters should only use normal data examples. In the classification process, the algorithm calculates all the test data points' distances to the closest cluster. If the calculated distance to each object is higher than a defined threshold value, the sample is classified as an anomaly.

A.2.5. Isolation Forest

Isolation Forest [118] is a method for outlier detection that uses data structures called trees, such as binary trees. Each tree is created by partitioning the instances recursively, by randomly selecting an attribute and a split value between the maximum and minimum values of the selected attribute. Being T an external node of a tree with no child or an internal node designated by a test with exactly two daughter nodes (T_l, T_r). A test is an attribute q with a split value p, where $q < p$ meaning the data points will be divided into T_l, T_r. To build an isolation tree, the data $X = x_1, \ldots, x_n$ will be recursively divided by randomly selecting an attribute q and a split value p, until it reaches three conditions [118]:

- The tree reaches a height limit;
- $|X| = 1$;
- All data in X have the same values;

To detect anomalies, the observations are sorted according to their path lengths or anomaly scores. The path length $h(x)$ represents the number of edges x that go through an isolation tree from the root node until the traversal is terminated at an external node. To calculate the anomaly score, Lui et al. [118] used the analysis from Binary Search Tree (BST) to estimate the average path length of an isolation tree represented in equation A.3:

$$c(n) = 2H(n-1) - (\frac{2(n-1)}{n}) \tag{A.3}$$

The observations from the dataset are represented by n and $H(i)$ is a harmonic number and can be estimated by the Euler's constant. The parameter $c(n)$ was used

to normalize $h(x)$ since it represents the average of given n. Equation A.4 represents the anomaly score s of an observation x:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{A.4}$$

$E(h(x))$ is the average of $h(x)$ from a collection of isolation trees. Using the anomaly score from [118], the authors verified that observations with a s value much smaller than 0.5 are quite safe to be regarded as normal observations, while observations with a s value very close to 1 are definitely anomalies, and observations that return $s \approx 0.5$ don't really mean any distinct anomaly.

A.2.6. One-Class Scaled Convex Hull

The Scaled Convex Hull (SCH) is an algorithm based on a previously proposed method by Casale et al. [37] that uses the geometrical structure of the Convex Hull (CH) to define the class in one-class classification problems. This algorithm uses random projections and an ensemble of CH models in two dimensions, and thus this method can be suitable for larger dimensions in an acceptable execution time [67]. As we can see in equation A.5.

$$CH(S) = \left\{ \sum_{i=1}^{|s|} \theta_i x_i | (\forall i : \theta \geq \theta) \wedge \sum_{i=1}^{|s|} \theta_i = 1, xi \in S \right\} \tag{A.5}$$

the CH of a finite set of points $S \in R^d$ provides a tight approximation among several convex forms being this approximation prone to over-fitting. Fernández-Francos et al. [67] used reduced/enlarged versions of the original CH to avoid the over-fitting problem, where in the training phase an outlier can lead to shapes that do not represent the target class accurately. To resolve this problem they applied the formula presented by Liu et al. [120] to calculate the expanded polytope, where the vertices are defined with respect to the center point $c = (\frac{1}{|S|}) \sum_i x_i, \forall x_i \in S$ and the expansion parameter $\lambda \in [0, +\infty]$ as in equation A.6:

$$v^\lambda : \{\lambda v + (1 - \lambda)c | v \in CH(S)\} \tag{A.6}$$

The parameter λ represents a constant extension ($\lambda > 1$) or constant contraction ($0 < \lambda < 1$) of the CH regarding c. An approximation of the decision made by the expanded CH in the original d-dimensional space by means of an ensemble of τ randomly projected decisions on 2-D spaces was proposed. In this way, the authors defined a decision rule that states that a point does not belong to the modelled class if and only if there exists at least one projection in which the point lies outside the projected CH.

To have a better understanding of this method, Figure A.4 is graphically represented as an example where a 3-D convex figure is approximated by three random projections in 2-D. We can observe in Figure A.4 that the point can be inside one or more projections but in fact, that point lies outside the original geometric form.

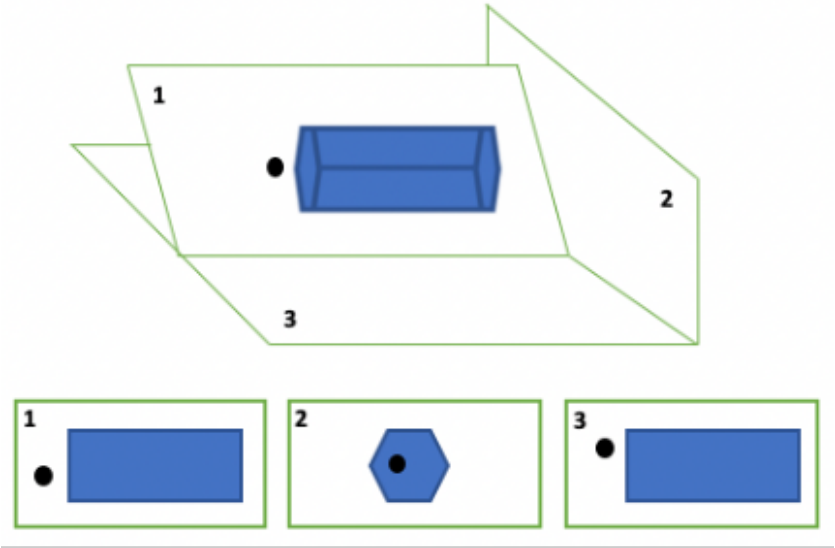


Figure A.4: Ensemble of projected decisions on 2-D based on Fernández-Francos et al. [67].

In the SCH algorithm Fernández-Francos et al. [67] proposed three different definitions of center:

1. The average of all points in the projected space;
2. The average of the CH vertices in the projected space;
3. The average position of all the points in the projected polytope.

Each type of center leads to different decision regions (if a point belongs or not to the target class), giving more flexibility to this method.

A.2.7. One-Class Support Vector Machines

SVM have the capability to solve classification and regression problems. This algorithm focuses on the search for a hyperplane (generalization of a plane in different dimensions, for example in a two-dimensional plane is a line that separates and classifies data) that better divides a dataset into two classes.

SVMs are effective in classifying linearly separable data or having an approximately linear distribution. However, there are many cases where it is not possible to properly divide the training data using a hyperplane. To solve this problem, SVMs can create a non-linear decision boundary by projecting a non-linear function ϕ to a space with a higher dimension. This means that SVMs can project the data from the training set of its original space I to a new space of greater dimension, denominated as feature space F [87].

To calculate the scalar products between objects mapped in the new space, functions called Kernels $K(x_i, x_i) = \phi(x)^T \phi(x_i)$ are used. The usefulness of kernels lies therefore in the simplicity of their calculation and in their capacity to represent abstract spaces. The most used kernels are the polynomials, the radial base function and the sigmoidal. Each of them has parameters that must be determined by the user.

The hyperplane equation is represented by $w^T x + b = 0$, with $w \in F$ and $b \in R$ in two-dimensional space. The constructed hyperplane as mentioned determines the margin between the classes. The use of slack variables ε_i will allow some data points to lie within the margin where the constant $C > 0$ determines the trade-off between maximizing the margin and the number of training data points within the margin [172]. This way they will prevent SVM from over-fitting with noisy data. The objective function of the SVM classifier is represented in equation A.7:

$$
\begin{aligned}
min_{w,b,\varepsilon} \frac{||w||}{2} + c\sum_{i=1}^{n} \varepsilon_i \\
subject\ to: \\
y_i(w^T\phi(x_i) + b) \geq 1 - \varepsilon_i, \forall i = 1, ..., n \\
\varepsilon_i \geq 0, \forall i = 1, ..., n
\end{aligned}
\tag{A.7}
$$

Therefore, for anomaly detection problems the One-Class Support Vector Machines (OCSVM or v-SVM) will only train with data from one class, in this case, the class that represents normal activity in the network. Basically, it separates all the data points from the origin and maximizes the distance from the hyperplane to the origin. This results in a binary function that captures regions in the input space where the probability density of the data lives. The minimization function is given by equation A.8 [172]:

$$
\begin{aligned}
min_{w,b,\varepsilon,p} \frac{1}{2}||w||^2 + \frac{1}{vn}\sum_{i=1}^{n} \varepsilon_i - p \\
subject\ to: \\
(w \cdot \phi(x_i)) \geq p - \varepsilon_i, \forall i = 1, ..., n \\
\varepsilon_i \geq 0, \forall i = 1, ..., n
\end{aligned}
\tag{A.8}
$$

As we can see in equation A.8 the parameter v characterizes the solution, instead of the C parameter that decided the smoothness in equation A.7. The parameter v sets an upper bound on the fraction outliers and a lower bound on the number of training examples used as support vectors. Due to the importance of this parameter, this approach is also known as v-SVM.

A.2.8. LOF

Within distance-based category methods, we can find the density-based method named LOF [27]. This algorithm computes the ratios between the local density area around a specific data point and the local densities of its neighbours. It provides a ranking estimator quantifying how isolated a point is with regard to the density of

its neighbourhood. Because this particular method ranks data points by only considering their neighbourhood density, it might miss potential anomalies if densities are close to those of their neighbours.

A.2.9. PA-I

Martínez-Rego et al. [126] proposed a modification of the One-class classification with passive-aggressive Kernel algorithm [52] and combined it with a Bernoulli CUSUM chart to deal with stream change detection problems. This adaptation provided the method with the capability of accurately fitting the support of normal data in an online fashion. Thus, it can dynamically adapt to changes in data distribution and automatically control the growth of the number of support vectors. The authors showed with empirical results in real datasets that the proposed method presented better change detection capabilities when compared to state-of-the-art algorithms for stream anomaly detection problems.

A.2.10. EADMNC

EADMNC is a new approach proposed by IRL Botana et al. [24] to address explanation with the recent ADMNC algorithm [62]. The authors designed ADMNC method to handle anomaly detection for large-scale problems with a mixture of categorical and numerical input variables. The approach presented in [62] uses a probabilistic perspective. A Gaussian mixture model is used to model continuous input features, while categorical input features are estimated using a logistic model with a maximum likelihood approach optimized with a stochastic gradient descent algorithm. The whole method implemented in Apache Spark with parallelized computation is thus scalable to large datasets. In [24], the ADMNC algorithm was extended, adding a new layer that opens the ADMNC black box by offering pre-hoc explainability. The authors used Regression decision trees to segment input data into homogeneous groups attending to their variables. The clusters, defined as leaf nodes of a shallow decision tree, will group elements with approximately the same level of anomaly, indicated by the average estimator that ADMNC assigned to the elements of said cluster[61]. The variance of the CART regression model provides

information about how homogeneous the estimators of elements in a tree node are. The tree splits turn nodes into more specific groups that contain similar elements. The cluster homogeneity and explanation quality, given by the depth of each path, allows the user to choose the level of detail for explanations.

A.2.11. LSHAD

The Locality Sensitive Hashing for Anomaly detection is a method proposed by Meira et al. [135] and detailed described in Chapter 3 based on the LSH technique initially proposed by [94]. The basic concept is to identify approximate nearest neighbours through the use of hash functions. Each hash function is represented by a random projection in feature space splitting data points into groups of neighbours. This technique works with the basic principle that when two points in the feature space are close, they are likely to have the same hash function. The procedure of generating hash functions presents a high random component, being therefore probabilistic. This randomness leads to the problem of false neighbours detection, where points in the same hash could be points far apart with low similarity. The algorithm concatenates Hash functions to decrease the occurrence of this event. LSH allows to speed up the neighbour's search by reducing the complexity compared to the K Nearest Neighbour algorithm.

The proposed method from Meira et al. [135] adopted LSH to deal with large amounts of data in anomaly detection problems. It uses the number of elements generated by the hash functions to estimate the anomaly score. The model incorporates a process for auto-tuning its hyperparameters and is developed in the Apache Spark framework for distributed environments, thus rendering a scalable algorithm.

A.3. Metrics

A.3.1. Area Under the Curve

A well-known way of comparing the classifier's performance is using the AUC metric. The AUC calculates the area under a ROC curve which is a graph showing

the classification performance at various thresholds settings drawn by two parameters, the TPR $\frac{TruePositives(TP)}{TP+FalseNegatives(FN)}$ and the FPR $\frac{FalsePositives(FP)}{FP+TrueNegatives(TN)}$. Each point on the ROC represents a $\frac{TPR}{FPR}$ pair corresponding to a particular decision threshold.

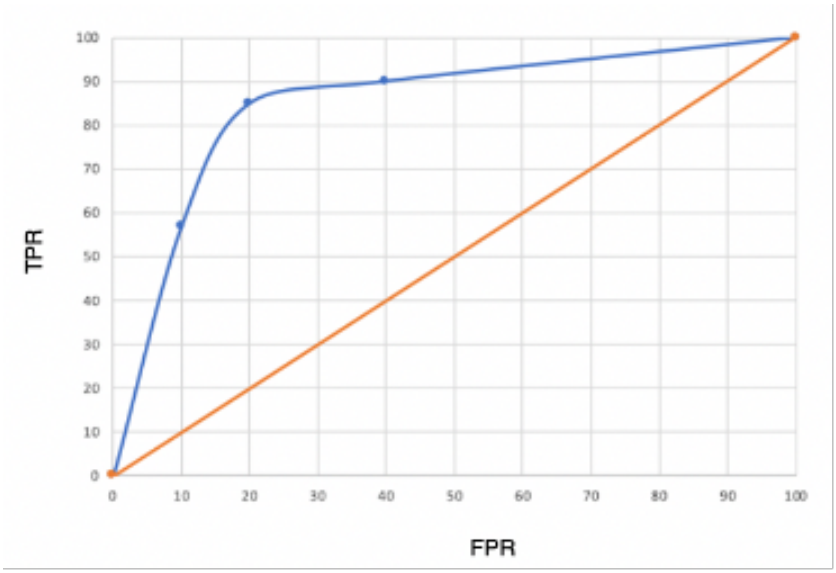


Figure A.5: ROC curve example.

Figure A.5 represents an example of a ROC curve, where we can see the trade-off between TPR and FPR. Those classification algorithms that have a curve close to the top-left corner indicate better performance (as seen in the blue line). As close as the curve is to the diagonal line (marked in orange) where TPR = FPR, the less accurate the classifier is.

The AUC) is a metric that can be useful to summarize the performance of each classifier providing an aggregate measure of performance across all possible classification thresholds. AUC can be seen as the probability of the model distinguishing between the positive class (anomaly) and the negative class (normal activity).

A.3.2. Accuracy, Recall, Precision, F1 Score

Accuracy in equation A.9 measures the percentage of correct predictions made by the ML model, calculated as the number of correct predictions divided by the total number of predictions. Regarding the precision and recall metrics in equation

A.10 and A.11, both measure the rate of FP and FN, respectively. For instance, a high recall value means low FN, while a small precision indicates high FP values. To analyze the balance of these two metrics, we compute the F1 score as the harmonic mean of Precision and Recall. While it is possible to take a simple average of the two scores, harmonic means are more resistant to outliers. When using these two metrics, there is often a tradeoff between them, so it is important to evaluate them together using another metric called F1 score, shown in equation A.12. Thus F1 score is a balanced metric that appropriately quantifies the correctness of models.

- True Positive (TP) - The number of observations correctly identified as an anomaly;
- False Positive (FP or Type I error) - The number of observations classified as an anomaly but corresponding to normal behaviour;
- True Negative (TN) - The number of observations correctly identified as normal behaviour;
- False Negative (FN or Type II error) - The number of observations classified as a normal behaviour but corresponding to anomalies;

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{A.9}$$

$$precision = \frac{TP}{TP + FP} \tag{A.10}$$

$$recall = \frac{TP}{TP + FN} \tag{A.11}$$

$$F1score = 2 * \frac{recall * precision}{recall + precision} \tag{A.12}$$

A.4. Nemenyi Statistical Test

The Nemmenyi post hoc test [56] is a statistical method used to compare the performance of multiple classifiers and it can be used to compare the performance of different models. The Nemmenyi Statistical test is a non-parametric method, meaning it does not rely on assumptions about the underlying distribution of the data. It is based on the idea of ranking the classifiers or treatments based on their performance and comparing the ranks to determine whether the differences between them are statistically significant. To perform the Nemmenyi statistical test, the mean rank for each classifier is first calculated:

Let's say we have K classifiers or treatments, denoted by $C1$, $C2$, ..., CK. For each classifier, we calculate the mean rank across all N data points as follows:

Mean rank for $C1 = \frac{\sum_{i=1}^{N} ranks_{C1,i}}{N}$

Mean rank for $C2 = \frac{\sum_{i=1}^{N} ranks_{C2,i}}{N}$

...

Mean rank for $CK = \frac{\sum_{i=1}^{N} ranks_{CK,i}}{N}$

Next, the CD is calculated. The CD is a measure of the minimum difference in mean ranks that is considered statistically significant. It is calculated as follows [197]:

$$CD = q_\alpha * \sqrt{\frac{K * (K+1)}{(6 * N)}} \tag{A.13}$$

where q_α is the critical value from the Student's t-distribution with $K-1$ degrees of freedom.

Finally, the mean ranks of each pair of classifiers are compared and determine whether the difference between them is greater than the CD. If it is, the difference is considered statistically significant:

For each pair of classifiers or treatments (Ci, Cj), we compare their mean ranks as follows:

If |Mean rank for $Ci-$ Mean rank for $Cj| > CD$, then the difference between Ci and Cj is considered statistically significant.

Appendix B

Resumen del trabajo

La detección de anomalías (AD) es una tarea crucial en muchos campos. La capacidad de identificar y analizar patrones inusuales o inesperados en los datos puede ayudar a las empresas a prevenir problemas potenciales y mejorar sus operaciones. Con el tiempo, la tecnología utilizada para AD ha evolucionado significativamente. Las primeras aproximaciones se basaron en métodos estadísticos simples e inspección manual de datos, pero estos tenían limitaciones en su capacidad para manejar conjuntos de datos grandes y complejos.

Hoy en día, la detección de anomalías es un campo ampliamente estudiado donde surgen constantemente nuevos enfoques. Con el auge de la inteligencia artificial y las tecnologías de big data, el campo de la detección de anomalías ha experimentado avances significativos y seguirá siendo un área importante de investigación y desarrollo en el futuro. A pesar de estos avances, todavía hay limitaciones que abordar, como métodos escalables capaces de manejar grandes volúmenes de datos, la falta de transparencia en la mayoría de los métodos de vanguardia, los costos computacionales del ajuste de hiperparámetros, o la capacidad de lidiar con datos en tiempo real sin la necesidad de etiquetas. Además, con el creciente énfasis en la sostenibilidad y la necesidad de reducir los residuos y minimizar el impacto ambiental, es importante asegurarse de que los métodos utilizados para la detección de anomalías también sean sostenibles. Esto significa considerar no solo la precisión y la eficiencia de los métodos, sino también los recursos que consumen y su impacto a largo plazo en el medio ambiente.

B.1. Desafíos en la detección de anomalías

Como acabamos de comentar, a pesar de los significativos avances logrados en las aplicaciones de detección de anomalías, todavía hay desafíos por alcanzar. A continuación se presentan varios retos actualeds para los métodos de detección de anomalías que están presentes en diferentes áreas de aplicación:

- **Big data**: El crecimiento de dispositivos inteligentes y la sensorización de las actividades industriales contribuyen a la generación de grandes volúmenes de datos previamente no vistos. El tamaño de los conjuntos de datos recopilados ha crecido constantemente, lo que ha despertado interés en los métodos de aprendizaje automático. La capacidad de estos métodos para aprender y realizar tareas complejas se veía limitada por la escasez de datos. Debido al aumento de la disponibilidad de datos, la complejidad de las tareas está ahora limitada por la capacidad del método de aprendizaje automático para extraer conclusiones relevantes. La alta complejidad computacional de los métodos de aprendizaje automático hace que sea poco práctico procesar grandes volúmenes de datos. Aunque hay varias soluciones presentadas en la literatura sobre algoritmos distribuidos y paralelizados para tratar con grandes conjuntos de datos, los enfoques existentes para la detección de anomalías con características distribuidas son escasos.

- **Datos sin etiquetar**: Los datos recopilados y disponibles de dispositivos inteligentes o sensores suelen estar sin etiquetar. Etiquetar los datos requiere un esfuerzo y tiempo significativos ya que suele hacerse manualmente por un experto en el dominio de aplicación. Es aquí donde los métodos de aprendizaje no supervisado pueden ser útiles, ya que pueden ayudar en la descubrimiento de patrones y relaciones ocultos en los datos sin la necesidad de etiquetas. Sin embargo, uno de los principales desafíos con el aprendizaje no supervisado es la dificultad para validar la calidad del modelo, ya que no hay etiquetas con las que comparar las predicciones.

- **Definir anomalías**: Las anomalías que surgen debido a una actividad maliciosa a menudo cambian y se adaptan. Por ejemplo, se entrena un modelo para reconocer patrones maliciosos en mensajes de correo electrónico, pero si apare-

cen nuevos patrones que antes no existían, el modelo puede perder rendimiento ya que no se reconocen como anormales. Este tipo de cambio en los datos se conoce como *concept shift*. Es un cambio en las propiedades estadísticas de los datos en los que se entrena un modelo de aprendizaje automático. Esto puede suceder con el tiempo a medida que la distribución de los datos cambia, o a medida que los objetivos o necesidades de los usuarios del modelo cambian. El comportamiento normal está en constante evolución, y la noción de comportamiento normal ahora puede no ser suficientemente representativa de los futuros comportamientos. El desafío del diseño de algoritmos de detección de anomalías se estudia utilizando solo muestras no anómalas. Como no sería factible desarrollar un marco genérico para cubrir todas las aplicaciones mencionadas anteriormente, se desarrollan varios modelos de detección de anomalías para que cada uno se ocupe de un dominio específico.

- **Ajuste de hiperparámetros**: El ajuste de hiperparámetros puede ser especialmente desafiante en el contexto de la detección de anomalías. El rendimiento de un modelo de detección de anomalías puede ser altamente sensible a la elección de hiperparámetros. La definición de lo que constituye una anomalía puede variar dependiendo de la aplicación, lo que dificulta evaluar el rendimiento de un modelo. Además, el espacio de hiperparámetros posibles puede ser grande y el proceso de optimización es laborioso, lo que dificulta encontrar el mejor conjunto de hiperparámetros para un conjunto de datos dado. Además, en muchos casos, el experto en el dominio que se encarga de ajustar los hiperparámetros de un modelo de detección de anomalías puede no tener una comprensión profunda de los datos subyacentes o las características del comportamiento anómalo. También es un desafío cuando se evalúa el rendimiento de un modelo de detección de anomalías, ya que en la mayoría de las ocasiones no hay una verdad absoluta con la que comparar las predicciones del modelo. Esto puede dificultar saber si los hiperparámetros elegidos son óptimos y si el modelo es capaz de detectar con precisión el comportamiento anómalo.

- **Explicabilidad**: La explicabilidad es particularmente importante en campos críticos como la salud, las finanzas o la detección de intrusiones, donde las consecuencias de tomar decisiones incorrectas basadas en los resultados de un algoritmo de detección de anomalías pueden ser graves. Al proporcionar a

los usuarios una mejor comprensión de cómo funciona el algoritmo y por qué detecta ciertas anomalías, la explicabilidad puede ayudar a mejorar la confiabilidad de los resultados. Sin embargo, la mayoría de los métodos propuestos no son transparentes y carecen de interpretabilidad.

En esta tesis, hemos intentado enfrentar algunos de estos problemas, y se presentaron varios métodos y estudios comparativos para abordar los desafíos descritos. A continuación, detallamos brevemente las principales contribuciones de este trabajo:

B.2. Nuevos algoritmos y modelos

Los nuevos algoritmos y modelos propuestos para abordar los desafíos de AD son los siguientes:

- La primera contribución es el algoritmo LSHAD, que es un nuevo y sostenible método AD basado en LSH capaz de manejar grandes conjuntos de datos. El algoritmo resultante es altamente paralelizable y su implementación en Apache Spark aumenta aún más su capacidad para manejar conjuntos de datos muy grandes. Además, el algoritmo incorpora un mecanismo de ajuste automático de hiperparámetros para que los usuarios no tengan que implementar un costoso ajuste manual. Este enfoque sostenible de AD tiene la ventaja adicional de ser un método no supervisado, que no necesita datos etiquetados para el proceso de aprendizaje. El mecanismo de ajuste de hiperparámetros es capaz de ajustar sus hiperparámetros a pesar del dominio de datos de entrada. El método LSHAD es novedoso, y sus características, automatización de hiperparámetros y propiedades distribuidas, no son habituales en las técnicas AD. Los resultados experimentales con LSHAD en una variedad de conjuntos de datos apuntan a un rendimiento comparable al de los métodos del estado del arte, con el beneficio añadido de poder manejar conjuntos de datos mucho más grandes que las otras alternativas. Además, se realizaron experimentos para buscar un equilibrio entre el rendimiento detectando anomalías y la escalabilidad, que demostraron que nuestro método ofrece ventajas significativas sobre los métodos competidores del estado del arte.

- Otra contribución es el enfoque basado en datos para la detección de anomalías utilizando un modelo de flujo de datos, que está diseñado también para ser más sostenible que otros modelos disponibles. El método propuesto ayuda en la detección temprana de fallos y errores en maquinaria antes de que alcancen etapas críticas. Presentamos un modelo de detección de anomalías siguiendo un enfoque no supervisado, combinando el método Half-Space-trees con One Class K Nearest Neighbor, adaptado para tratar con flujos de datos. Dado que los datos que llegan de los sensores son interminables y recibidos como un flujo continuo, el método propuesto es capaz de tratar con flujos de datos donde los recursos computacionales de los métodos son limitados (memoria, poder computacional, tiempo de procesamiento) dando sostenibilidad al método. El modelo se basa en el aprendizaje incremental ya que los datos se inducen incrementalmente y contemplan un mecanismo de olvido para tratar con la memoria limitada. Evaluamos nuestro enfoque y lo comparamos con el método Half-Space-Trees aplicado sin la combinación One Class K Nearest Neighbor. Nuestro modelo produjo pocos errores de tipo I, aumentando significativamente el valor de precisión en comparación con el modelo Half-Space-Trees. Nuestra propuesta logró un alto rendimiento en detección de anomalías, predeciendo la mayoría de los fallos catastróficas del sistema de tren APU.

- Por último, la tesis presenta un estudio de estrategias con técnicas de PLN (procesado de lenguaje natural) para predecir las preferencias turísticas en base a opiniones de usuarios. En este trabajo, exploramos diferentes métodos de aprendizaje automático para predecir las calificaciones de los usuarios. Utilizamos estrategias de PLN para predecir si una reseña es positiva o negativa y la calificación asignada por los usuarios en una escala de 1 a 5. A continuación aplicamos métodos supervisados como la Regresión Logística, Random Forest, Decision Trees, K-Nearest Neighbors y Redes Neuronales Recurrentes para determinar si a un turista le gusta/no le gusta un determinado punto de interés. Además, utilizamos un enfoque distintivo en este campo mediante técnicas no supervisadas para problemas de detección de anomalías. El objetivo era mejorar el modelo supervisado en la identificación solo de aquellos turistas que realmente les gusta o no les gusta un punto de interés particular, en el cual el objetivo principal no es identificar a todos, sino fundamentalmente no fallar

a aquellos que se identifican en esas condiciones. Los experimentos realizados mostraron que los modelos desarrollados podrían predecir con alta precisión si una reseña es positiva o negativa pero tienen alguna dificultad en predecir con precisión la calificación asignada por los usuarios. El método no supervisado Local Outlier Factor mejoró los resultados, reduciendo los falsos positivos de la Regresión Logística con un costo asociado de aumentar los falsos negativos.

B.3. Aplicaciones prácticas

En esta tesis, también se exploran las aplicaciones de los algoritmos y modelos propuestos en varios campos. La principal aplicación es la detección de intrusos, ya que la seguridad cibernética es un área crítica en los sistemas informáticos, especialmente cuando se trata de datos sensibles. En la actualidad, es cada vez más importante asegurar que los sistemas informáticos estén protegidos de los ataques debido a la dependencia de la sociedad moderna de estos sistemas. Para prevenir estos ataques, actualmente la mayoría de las organizaciones utilizan sistemas de detección de intrusiones (IDS) basados en anomalías. Generalmente, los IDS contienen algoritmos de aprendizaje automático que ayudan a predecir o detectar patrones anómalos en los sistemas informáticos. La mayoría de estos algoritmos son técnicas supervisadas, que presentan lagunas en la detección de patrones desconocidos o explotaciones de día cero, llamados así ya que estos no están presentes en la fase de aprendizaje del algoritmo. Para abordar este problema, en esta tesis se presenta un estudio empírico de varios algoritmos de aprendizaje no supervisado utilizados en la detección de ataques desconocidos. En este estudio, evaluamos y comparamos el rendimiento de diferentes tipos de técnicas de AD en dos conjuntos de datos disponibles públicamente: NSL-KDD e ISCX. El objetivo de esta evaluación es entender el comportamiento de estas técnicas y comprender cómo podrían ajustarse en un IDS para llenar la mencionada laguna. También, la presente evaluación podría utilizarse en el futuro como una comparación de resultados con otros algoritmos no supervisados aplicados en el campo de la ciberseguridad. Los resultados obtenidos muestran que las técnicas utilizadas son capaces de realizar detección de anomalías con un desempeño notable y, por lo tanto, son candidatos adecuados para su futura integración en herramientas de detección de intrusos.

En cuanto a la detección de intrusos, se presentaen esta tesis un estudio de evaluación de nuevos métodos no supervisados para lidiar con ataques IoT. El desafío de proporcionar seguridad a las redes se está haciendo cada vez máis difícil, especialmente con la recente aparición de dispositivos inteligentes con seguridad media a baja que se unen a las redes en todo el mundo. Con la evolución del IoT y los dispositivos cada vez más conectados a Internet, surge el reto de asegurar la seguridad e integridad de la red y de todos los dispositivos conectados. El uso de IDS busca ayudar a proteger las redes, por ejemplo, impidiendo que los dispositivos IoT se utilicen de manera maliciosa o alertando de cuando fueron comprometidos. Esta tesis explora el nuevo conjunto de datos Aposemat IoT-23 de tráfico de red que contiene malware y escenarios benignos ejecutados en dispositivos IoT. Inicialmente, realizamos un análisis exploratorio del conjunto de datos. Lo utilizamos para evaluar varios métodos de detección de anomalías en relacioón a su capacidad para distinguir entre el comportamento de red normal y anormal. Por tanto, una de nuestras contribuciones es presentar una comparación de métodos del estado del arte para detectar intrusiones en IoT. En nuestra evaluación, probamos las tasas de detección de anomalías y el rendimiento del tiempo de procesamento, pero también exploramos la utilidad de los árboles de explicación obtenidos por un nuevo algoritmo de detección de anomalías denominado **EADMNC**. Los resultados mostraron que la mayoría de los métodos examinados tienen una excelente puntuación detectando anomalías, pero el One-Class kNN obtiene sistemáticamente el mejor rendimiento. Con todo, este método tiene la desventaja de su falta de escalabilidad y, por lo tanto, mostramos que los nuevos métodos como **LSHAD** y **EADMNC** son mucho más adecuados para tratar con grandes conjuntos de datos.

Otro campo ampliamente estudiado en AD es el Mantenimiento Predictivo. La aparición de la Industria 4.0 trae automatización y intercambio de datos a la fabricación industrial. El uso de sistemas computacionales y dispositivos IoT permite a las empresas recopilar y manejar grandes volúmenes de datos sensoriales y de procesos comerciales. El crecimiento y la proliferación de las tecnologías de big data y aprendizaje automático permiten tomar decisiones estratégicas basadas en los datos analizados. En esta tesis se presenta un método de mantenimiento predictivo basado en datos para el sistema de unidad de producción de aire (APU) de un tren del Metro do Porto (Portugal). El modelo propuesto también es aplicable en el campo del mantenimiento predictivo, donde se ha demostrado ser efectivo en la

identificación y predicción de fallos en el equipo, reduciendo el tiempo de inactividad y los costos. Esto se logra mediante el uso de datos en streaming, lo que permite procesar los datos en tiempo real y detectar anomalías de manera oportuna.

Además, abordamos el campo de los modelos de diálogo basados en argumentación. Estos modelos han demostrado ser apropiados para contextos de decisión en los que se pretende superar la falta de interacción entre los tomadores de decisiones, ya sea porque están dispersos, son demasiados o simplemente no se conocen. Sin embargo, para apoyar procesos de decisión con modelos de diálogo basados en argumentación, es necesario tener conocimiento de ciertos aspectos específicos de cada tomar de decisiones, como preferencias, intereses y limitaciones, entre otros. La falta de obtención de esta información podría arruinar el éxito del modelo. Buscamos facilitar el proceso de adquisición de información mediante el estudio de estrategias para predecir automáticamente las preferencias de los turistas (calificaciones) en relación con los puntos de interés en base a sus opiniones. El estudio de la predicción de preferencias turísticas basadas en opiniones, utilizando modelos de diálogo basados en argumentación, también proporcionó una visión sobre el uso de técnicas de PLN y no supervisadas en este campo y mostró cómo este enfoque puede facilitar el proceso de adquisición de información y predecir las preferencias de los turistas en base a sus opiniones. Los resultados de este estudio demuestran el potencial de este enfoque en contextos de toma de decisiones y su capacidad para superar la falta de interacción entre los tomadores de decisiones.

Como se ha visto, esta tesis abarca una amplia gama de problemas derivados de la aparición del campo de detección de anomalías teniendo también en cuenta las consideraciones de sostenibilidad. Los enfoques propuestos han demostrado su capacidad para enfrentarse a problemas con grandes cantidades de datos sin etiquetar, proporcionando mecanismos de ajuste de hiperparámetros e interpretabilidad de datos. Por lo tanto, se espera que la contribución de esta tesis abra la puerta al desarrollo de nuevos modelos sostenibles que puedan abordar los problemas de detección de anomalías teniendo en cuenta las limitaciones mencionadas.