Universidade do Minho
Escola de Engenharia

Eduardo Veloso Eiras

Anomaly detection for preventive and predictive maintenance systems

October, 2023

Universidade do Minho
Escola de Engenharia

Eduardo Veloso Eiras

Anomaly detection for preventive and predictive maintenance systems

Master's Dissertation
Master's in Informatics Engineering

Work supervised by
César Analide de Freitas e Silva da Costa Rodrigues
Bruno Filipe Martins Fernandes

October, 2023

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

Braga, 06 de Outubro de 2023

(Location) (Date)

Assinado por: **Eduardo Veloso Eiras**
Num. de Identificação: 15081765
Data: 2023.10.06 10:10:35+01'00'

(Eduardo Veloso Eiras)

Acknowledgements

Throughout the writing of this dissertation, I was fortunate to count on the support of several people that I would like to present my thanks.

Firstly I would like to thank my supervisors, Bruno Fernandes and César Analide, which helped me with the development of this dissertation. It was through their constant advice and counselling that I was able to proceed with the development and writing of this dissertation, so once again thank you both for your dedication and guidance. Also would like to thank We Can Charge, more specifically Ricardo Carvalho and Xavier Rodrigues who were essential in the development of this project. Thank you for your availability to answer my questions, and the data you provided, without your contribution this work would not be possible.

Secondly, I would like to give a special thanks to my parents, my brother, my grandparents and in general to my whole family for the support they provided while I developed this work. I would also like to extend my thanks to all of my friends. Without your friendship and support, this journey would not have been the same.

On a closing note, thank you so much to all the people that were a part of this journey and know I am deeply grateful for all of you.

Resumo

Deteção de anomalias para criação de sistemas de manutenção preventiva e preditiva

Dado o aumento da preocupação relativamente às alterações climáticas, e políticas de redução dos efeitos das alterações climáticas, verifica-se cada vez mais um aumento na procura por veículos elétricos. Esta procura, implica alterações na rede de abastecimento de veículos com a necessidade de incluir estações de carregamento. Estas alterações, criaram condições para que novas empresas pudessem surgir, como é o caso da We Can Charge. Porém, este aumento, implica também novos desafios na monitorização, manutenção e especificamente na deteção de anomalias nas mesmas.

Atualmente, a We Can Charge utiliza o *Open Charge Point Protocol* (OCPP). Este é um protocolo e standard de código aberto, que permite a comunicação entre os diversos componentes de uma rede de carregamento de veículos elétricos, permitindo obter informação relevante sobre os mesmos pela análise da informação enviada nos pacotes de comunicação. Através de uma ferramenta dedicada, é possível à We Can Charge, detetar a ocorrência de anomalias. Apesar disto, a deteção das anomalias não é imediata e usa uma quantidade limitada da informação fornecida pelo protocolo, não utilizando a informação do estado dos conectores e transações de carregamento.

Assim, o principal objetivo desta dissertação é analisar e utilizar os dados recolhidos sobre o estado dos seus conectores e transações de carregamento, aplicando-os a algoritmos de *machine learning*. Para tal, efetuamos a criação de vários modelos, um modelo de deteção em tempo real, onde obtivemos os melhores resultados utilizando o algoritmo Isolation Forest, e dois modelos de previsão baseados em redes LSTMs, um relativo ao estado e outro ao número de erros reportados nos dados relativos ao estado dos conectores. Combinando estes modelos, foi possível a criação da *Connectors Forecasting Network* (CNF), que nos permite a previsão de anomalias futuras nas estações de carregamento.

Palavras-chave: Deteção de Anomalias, *Machine Learning*, *Open Charge Point Protocol*, Previsão de Anomalias, Redes de Carregamento, Veículos Elétricos

Abstract

Anomaly detection for preventive and predictive maintenance systems

In recent years the concern for environmental changes and the increasing measures imposed reduce climate change effects led to an increase in demand for electric vehicles. The growing number of these vehicles, in turn, calls for changes in the re-fuelling network introducing the need to include and manage charging stations where these can charge. This change formed an opportunity for new companies such as We Can Charge to emerge. However, the increase in the size of the network by increasing the number of charging stations comes with an uprising challenge when it comes to managing and ensuring that any anomalies are detected and resolved in the shortest time possible, allowing the continued use of their services.

the *Open Charge Point Protocol* (OCPP) is an open-source standard used in enabling the communication between a charging network's components. Currently, We Can Charge uses a dedicated and custom tool that analyses *OCPP* packets containing information sent by each charging station. However this tool only uses part of the data that *OCPP* provides, not making use of important information like connector status and charging transaction information.

Therefore, this study aims to utilise this data, applying several machine learning algorithms and comparing the obtained results in order to determine which model performs best in the real-time anomaly detection task, but also in forecasting these anomalies. To accomplish that task, we created a real time detection model, obtaining the best results with the Isolation Forest algorithm, enabling us to classify connector status data, and two LSTM forecasting models one for the status and another for the error counts obtained from connector status data. These results enabled us to combine these models in a single pipeline creating the *Connector Forecasting Network* (CFN), which in turn allowed us to predict future anomalies in the charging stations of the network.

Keywords: Anomaly Detection, Anomaly Forecasting, Charging Networks, Electric Vehicles, Machine Learning, Open Charge Point Protocol

Contents

List of Figures ix

List of Tables xi

Glossary xii

Acronyms xiii

Symbols xvi

1 Introduction 1

1.1 Context & Motivation . . . 1
1.2 Goals & Research Questions . . . 2
1.3 Document Outline . . . 3

2 State of the Art 5

2.1 Electric Vehicle Charging Networks . . . 6
2.1.1 Open Charge Point Protocol . . . 6
2.1.2 Electric Vehicle Charging Network Architecture . . . 7
2.1.3 Open Charge Point Protocol Charging Transaction . . . 9
2.2 Anomaly Detection Techniques . . . 10
2.2.1 Cluster Analysis Based Techniques . . . 11
2.2.2 Tree-Based Techniques . . . 13
2.2.3 Density-Based Techniques . . . 15
2.2.4 Support Vector Machines . . . 17
2.2.5 Neural Networks Related Techniques . . . 20
2.3 Model Evaluation Metrics . . . 24
2.3.1 Supervised Learning . . . 24
2.3.2 Semi-Supervised Learning . . . 26
2.3.3 Unsupervised Learning . . . 27

2.4 Anomaly detection applied to OCPP data 28
2.5 Summary 30

3 Experimental Setup **35**
3.1 Methodology 35
3.2 Experiments 37
3.3 Data Analysis 39
3.3.1 Connector Status Dataset 39
3.3.2 Charging Transactions Dataset 43
3.4 Data Preprocessing 50
3.5 Summary 54

4 Results and Discussion **57**
4.1 Real Time Anomaly Classification 57
4.2 Anomaly Forecasting over Connector Status Data 62
4.2.1 Status Forecasting Model 62
4.2.2 Error Counts Forecasting Model 66
4.3 Connectors Forecasting Network (CFN) 69

5 Conclusion **72**
5.1 Conclusions 72
5.2 Future Work 73
5.3 Final Thoughts 74

Bibliography **75**

Appendices **80**

A Detailed Status Forecaster Results **80**

B Detailed Status Ensemble Results **81**

C Detailed Error Counts Forecaster Results **83**

Annexes **86**

I Publications **86**
I.1 Anomaly detection for preventive and predictive maintenance systems 86

List of Figures

1 Architecture of an EV charging network using OCPP. Adapted from [3] 7
2 Random initialisation of centroids [6] 12
3 Centroids converge on the centre of their cluster [6] 13
4 Example visualisation of binary trees generated by the isolation forest algorithm. Adapted from [9] 14
5 DBSCAN visualisation. Adapted from [13] 16
6 Visualisation of a hyperplane separating two distinct classes of data points. Adapted from [16] 18
7 Plot of original data in two dimensions. Adapted from [17] 19
8 Data after the application of the polynomial kernel. Adapted from [17] 19
9 Boundary projection generated of a hyperplane. Adapted from [19] 20
10 General structure of an AE 21
11 Structure of a VAE. Adapted from [23] 23
12 Confusion matrix 24

13 CRISP-DM Methodology Lifecycle. Adapted from [34] 36
14 CFN proposed architecture 37
15 Number of status notification packets sent per day 40
16 Total number of status notification packets sent for each hour 41
17 Distribution of status values in the dataset 41
18 Types of errors present in the dataset 42
19 Types of errors related with the state value registered 42
20 Anomalous behaviour of a connector in the dataset 43
21 Number of packets registered per day 45
22 Number of packets registered per hour 46
23 Stop Transaction values distribution 46
24 Normal transaction Wh curve 47
25 Transaction Wh curve where a power loss was detected 47
26 Wattage *a)*, amperage *b)* and voltage *c)* curves for a transaction in the dataset 48

27 Anomalous wattage curve 49
28 Percentage of normal and abnormal cases in the dataset after classification 52

29 Learning curves of the first LSTM status forecaster network 62
30 Real and predicted results of the first status forecaster in the training set 63
31 Real and predicted results of the first status forecaster in the test set 63
32 Learning curves of the OSF model 64
33 Real and predicted results of the OSF model in the training set 64
34 Real and predicted results of the OSF model in the test set 64
35 Learning curve of the WOSF model 65
36 Real and predicted values for the WOSF model in the training set 65
37 Real and predicted values for the WOSF model in the test set 66
38 First test performed using the errors dataset 67
39 Learning curves after reducing the batch size and learning rate 67
40 Learning curves of the model with dropout, batch size and epoch increase 68
41 Real and predicted values for the errors training set 69
42 Real and predicted values for the errors test set 69
43 Confusion matrix of the CFN model 70
44 ROC and Precision-Recall curves of the CFN model 70

45 Confusion matrix of the status ensemble over the training set 81
46 Confusion matrix of the status ensemble over the test set 82

47 Real and predicted results of the first test in the training set 84
48 Real and predicted results of the first test in the test set 84
49 Real and predicted results of the second test in the training set 84
50 Real and predicted results of the second test in the test set 85

List of Tables

1 Connector Status dataset structure 39

2 Charging Transactions dataset structure 43

3 Status column numerical encoding 50

4 *Error_code*, *error_info* and *vendor_error_code* one-hot column grouping 51

5 Impact of the preprocessing techniques over the Silhouette score on several clustering algorithms 52

6 Impact of the preprocessing techniques over the Calinski Harabasz score on several clustering algorithms 52

7 Classification results using default hyperparameters 58

8 Hyperparameter search space for each algorithm 59

9 Classification results after hyperparameter tuning 61

10 Hyperparameter search space for the status LSTM forecaster 63

11 Classification metrics of the status ensemble in the test set 66

12 Classification metrics for the CFN model 70

13 LSTM detailed status forecaster results 80

14 Detailed classification metrics of the status ensemble 81

15 LSTM detailed error forecaster tests 83

Glossary

Charge Point/Station	The charge point or also refered to as the charge station is a physical system where an electric vehicle can be charged and can contain one or more connectors.
Charging Transaction	The period where a vehicle is charged stating with the user's first authentication in the charging station and ending with the payment after the charging process is complete.
OCPP-compliant	A charging network is considered OCPP-compliant if it holds at least the minimum requirements to be approved by a certification authority. For OCPP networks, the certification authority is the Open Charge Alliance.
OCPP-S	Open Charge Point Protocol using SOAP
OCPP-J	Open Charge Point Protocol over Web Sockets using JSON
OpenADR	OpenADR or Open Automated Demand Response is an open Smart Grid standard that allows a secure exchange of information in a two-way format. The standard promotes the interoperability between utilities, ISOs (Independent System Operators), energy management and control systems
Radio-Frequency Identification	Wireless communication that can be used to identify a person, objects or animals through the use of radio frequencies.

Acronyms

A	Amperes
AE	Autoencoders
AI	Artificial Inteligence
AUC	Area Under the ROC Curve
AUC-PR	Area Under the Precision-Recall Curve
CC	Constant Current
CFN	Connectors Forecasting Network
CRISP-DM	Cross-Industry Standard Process for Data Mining
CS	Charge Station
CSMS	Charging Station Management System
CV	Constant Voltage
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DSO	Distribuition System Operator
EEC	Encoded with Error Counts
EG	Encoded and Grouped
EIF	Extended Isolation Forest
EMS	Energy Management System
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
GAN	Generative Adversarial Networks
IF	Isolation Forest

JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors
kWh	Kilowatt-hour
LC	Local Controller
LP	Local Proxy
LSTM	Long Short-Term Memory Networks
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multilayer Perceptron
MSE	Mean Squared Error
OCA	Open Charge Alliance
OCPP	Open Charge Point Protocol
OpenADR	Open Automated Demand Response
OSF	Optimised Status Forecaster
RFID	Radio-Frequency Identification
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SOAP	Simple Object Access Protocol
SVM	Support Vector Machines
TLS	Transport Layer Security
V	Voltage
VAE	Variational Autoencoder

W	Watts
Wh	Watt-hours
WOSF	Weighted Optimised Status Forecaster

Symbols

ARI	The Adjusted Rand Index value
FN	The number of false negative cases
FP	The number of false positive cases
FPR	The number false positive cases correctly identified
N	The number of elements in both the true and predicted labels sets
p	Represents the probability estimate computed by a machine learning model
RI	The Unadjusted Rand Index value
s	Sillhouette Coefficent value
TN	The number of true negative cases
TP	The number of true positive cases
TPR	The number of true cases correctly identified
w	The weight parameter for the weighted mean squared error
y	The true label value for a dataset entry
y_{pred}	The predicted labels set
y_{true}	The true labels set

Chapter 1

Introduction

This chapter aims to introduce and contextualise the topic and main motivation behind this study, as well as give a brief description of the main objectives and research questions, followed by an explanation of how this document is structured and divided.

1.1 Context & Motivation

The management of Electric Vehicle (EV) charging stations is no easy task. Apart from managing prices and transactions, supervising EV charging stations comes with added challenges when it comes to keeping the network working by constantly surveying the network's status to detect possible anomalies constraining the use of the service provided. The detection of these anomalies must be done as briefly as possible in order to maintain the services at the charging stations, as an interruption of the service can prove to be very costly. The detection of anomalies at these stations is possible due to a structure built over the free open standard Open Charge Point Protocol (OCPP). This protocol was launched initially by the ELaadNL foundation, with certifications currently being distributed by the Open Charge Alliance (OCA) [1].

The OCPP protocol enables the communication between a Charging Station Management System (CSMS) and multiple EV charging stations, allowing one to receive information about the status of each station, for example. Each station regularly informs the CSMS by sending packets whose contents allow knowing, for example, if a vehicle is currently charging, for how long the charging process has been going and billing information regarding the current transaction. Apart from this, OCPP provides other useful features to check a station's activity by exchanging simple heartbeat packets between the stations and the CSMS, which is useful to inspect if a given station is currently online.

We Can Charge [2] is a company dedicated to creating charging networks for private companies and businesses, implementing the network components and structure following the guidelines of the OCPP protocol. The company also develops software that supports the management of their charging networks, with the ability to manage consumption, pricing and monitor each station in the network. Although We Can Charge already monitors the status of their stations, using the information provided by the OCPP protocol, this monitoring process uses a limited part of the data, predominantly heartbeat packets. Although this method enables the detection of a downtime period of a charging station, as the network grows, these anomalies need to be detected at a higher rate to avoid service interruptions. We can achieve this by preemptively analysing historical data in order to predict the occurrence of these anomalies allowing us to preemptively act, for example, by performing maintenance jobs before another failure is detected. Additionally, the existing tools used by We Can Charge to detect if stations are offline, do not allow for real-time detection and anomaly predictions, as it requires to be run at the end of each day over historical OCPP data sent by each station.

Considering the characteristics of these tools, the necessities of the company and the state of the art relative to anomaly detection and prediction, it is possible to devise a more automated solution using machine learning techniques. The application of machine learning, in this context, would allow us to create a maintenance system capable of detecting anomalies using the data from the stations as it comes in, enabling the detection of irregularities in real-time, but also their forecasting. To achieve this, we aim to use additional OCPP information regarding the charging connector status and charging transactions data gathered from several charging stations managed by We Can Charge.

1.2 Goals & Research Questions

Taking into account the problem in question, the main goal of this study is the implementation of machine learning models to detect and predict anomalies in EV charging stations. Nonetheless, to achieve this goal, the following objectives must be achieved:

1. Study of the OCPP protocol structure and the algorithms to be used;
2. Analyse gathered data to identify and raise characteristics or patterns present, and application of data pre-processing techniques;
3. Develop machine learning models using the chosen algorithms;
4. Use optimisation techniques to optimise each model's performance;
5. Test and compare the results obtained from the different algorithms;

Additional to the objectives presented above, this study also aims to provide answers the following research questions (RQ):

RQ1: Is it possible to detect anomalies using OCPP transaction and connector status data?

RQ2: Which machine learning algorithm provides the best performance for anomaly detection when applied to OCPP data?

RQ3: Can deep learning techniques be used to aid the anomaly detection or forecasting process?

RQ4: Can the best-performing model be deployed to aid We Can Charge in detecting and predicting anomalies more preemptively?

1.3 Document Outline

This document, is structured into five chapters, starting with the Introduction (1), followed by the State of the Art (2), Experimental Setup (3), Results and Discussion (4) and finally the Conclusion (5) chapter.

In chapter 1, we describe the motivation behind this study and the current situation regarding the management of the charging networks by We Can Charge, setting the main objectives and clarifying this document's structure, correspondent to this section.

In chapter 2, we approach several topics concerning this study's aim while reviewing articles and studies from other authors in literature and analysing the current state of the art. This chapter starts with a description of the OCPP protocol and the overhaul architecture of the network implemented for communicating and managing charging stations describing each component that constitutes a charging network, in sections 2.1.1 and 2.1.2, respectively. In section 2.1.3, we explain the nominal execution of a charging transaction. In section 2.2, the field of anomaly detection is approached, describing the common paradigms associated with this field, and typical techniques found in the literature. For each one of the techniques, we present some practical examples of their application, in the form of different algorithms, with descriptions located in each section dedicated to each technique, more specifically in the sections between 2.2.1 to 2.2.5. Additionally, and considering we aim to compare the performance of the models we aim to conceive, we approach several evaluation metrics, for the different machine learning paradigms in section 2.3.

Chapter 3 starts with the description of the proposed approach to solve the problem, including the definition of the chosen project methodology, in this case, CRISP-DM and presenting the technologies, frameworks and libraries used in this study. After this, we approach some of the experiments done, explaining the proposed architecture for the forecasting network used to predict the occurrence of anomalies in the connectors, named Connectors Forecasting Network (CFN), and briefly the reasons that lead us to abandon the use of the charging transactions dataset. Later in the same chapter, we explain the data analysis done in both datasets provided by We Can Charge, followed by the data prepossessing required in order to prepare the connector status dataset to be applied to the machine learning algorithms, explaining in greater detail why we were unable to use the charging transaction dataset.

In chapter 4, we aim to discuss the obtained results for the real-time anomaly detection model by initially showing and comparing the different results obtained by the application of multiple anomaly detection techniques and algorithms, describing the optimisations done and discussing the results that lead to the final choice of using the Isolation Forest algorithm. Following this, we describe the process of conceiving the status forecasting models, the model ensemble performed and the error counts forecasting model using connector status data. In the sections that compose this chapter we go through the main tests and describe the results and our thought process behind the manual optimisation of each forecaster. Lastly, we approach the final pipeline construction, corresponding to the CFN model, detailing the model's results on the chosen evaluation metrics.

Finally, in chapter 5, we summarise the conclusions and results obtained through the work developed during the course of this study, providing answers to the questions raised initially, defining the future work and finishing with some final thoughts.

Chapter 2

State of the Art

In this chapter, the state of the art is presented, describing the theoretical foundations accompanied by descriptions of topics related to the scope of this study. The chapter starts with a brief introduction to electric vehicle charging networks, explaining why they exist and are an integral part of the electric vehicle charging infrastructure. Following this introduction, we describe the Open Charge Point Protocol (OCPP), used in the communication between charging stations and their management systems, analysing some advantages when employing this open-source communication standard. After this, we demonstrate the general architecture of an OCPP-compliant network, describing the function of each network component. In the last section regarding the OCPP protocol, we describe an example of a charging transaction and explain how it works in this type of network, describing the main requests and information passed from the charging stations to the charge point management system.

After describing the foundations of OCPP and the network architecture, we approach the topic of anomaly detection referencing common machine learning paradigms associated with it and some standard techniques used and referenced in the literature. Inside each section devised for each technique, we describe one or more application examples in the form of machine learning algorithms, such as the K-Means in section 2.2.1, Isolation Forest in 2.2.2, DBSCAN in 2.2.3, Support Vector Machines (SVM) and One-Class SVM in 2.2.4. For each algorithm, we explain the fundamental aspects of how they function and how each applies the associated technique, referencing relevant studies that discuss and develop these algorithms, with some employing them for anomaly detection. Over the years, new techniques and studies in this field have been developed and applied, with recent studies suggesting the use of autoencoders to aid the clustering process of algorithms such as the K-Means or be used independently for anomaly or outlier detection. This technique is under the artificial neural networks, more precisely in the field of deep learning, where a network is trained to look at the input data and generate output as close to the initial input as it can be. The process involves initially encoding the inputs, and compressing them into a compact representation, which is then decoded to try and reconstruct the original input from the features extracted. This technique is approached in more depth in section 2.2.5.

After this, in section 2.4, we approach in more depth other studies present in literature that also attempt to use OCPP data with machine learning techniques in order to automate and perform anomaly detection tasks. Following this topic, a summary of all this chapter's content is present in section 2.5

2.1 Electric Vehicle Charging Networks

The increase in the use and production of electric vehicles (EVs) has increased the demand for charging stations (CSs) and pressured changes and innovation in the ways t charge these vehicles. In the way combustion engine vehicles need to be refuelled, EVs need to be charged, leading to the reduced number of CSs to steadily grow over the years. As the construction of more stations progresses, new challenges come to decide where to place and how to manage these stations. Some are assembled in gas stations, where refuelling transactions for combustion vehicles, for example, are already monitored and approved by gas station workers. Considering that these stations need to be wildly available, having workers managing each station would not be a solution, as costs would be too high. In addition, charging an EV is a process that takes longer and requires additional control as we need to integrate multiple systems that can be from different vendors. The solution to this problem is the automation and integration of all charging stations in a network with a centralised architecture, where all of these communicate with a management system to authenticate users, approve charging transactions, manage the power supplied to each one, and the pricing demanded to users in these locations, without the need for human intervention.

2.1.1 Open Charge Point Protocol

Implementing an EV charging network can be done using several tools and protocols to enable communication between the core components that assemble a charging network. As stated before, the evolution of EV charging networks was followed by the development of standards and new protocols for their implementation, one of the most popular being the Open Charge Point Protocol (OCPP).

This protocol started as a project of the ElaadNL foundation in 2009, which aimed to enable interoperability and communication between EV charging stations and central management systems from different vendors. Over the years, several versions of the protocol were implemented, each bringing new features to enable better control and security with the communication between the EV charging stations and the central system. With several companies implementing and helping in the development of OCPP, in 2014, the Open Charge Alliance (OCA) was founded to centralise testing and certification of new OCPP implementations worldwide [1].

The main advantages of using an open-source protocol like OCPP are interoperability, flexibility and scalability. These advantages come from the fact that if a station is OCPP-compliant, there are no hardware restrictions due to the compatibility offered by the OCPP implementation, when compared to proprietary charging networks which use their communication protocols, often with hardware-specific implementations which remove flexibility when applied to new hardware.

The protocol also has two possible implementations, OCPP-J, where devices implement web sockets and send data in JSON format, or OCPP-S, where data is sent via SOAP. Currently, the company supporting this study, We Can Charge, uses the OCPP-J approach.

2.1.2 Electric Vehicle Charging Network Architecture

The architecture of an EV charging network is constructed of several components, which communicate with each other using a communication protocol. According to the OCA and the OCPP documentation, a charging network is composed of at least three main components, the on-site components, which include the Charge Station (CS), Electric Vehicle Supply Equipment (EVSE), and an offsite component that coordinate several charging stations called the Charging Station Management System (CSMS). As long as these three main components are present, others can be inserted or removed depending on the complexity needed in the implementation of the network.

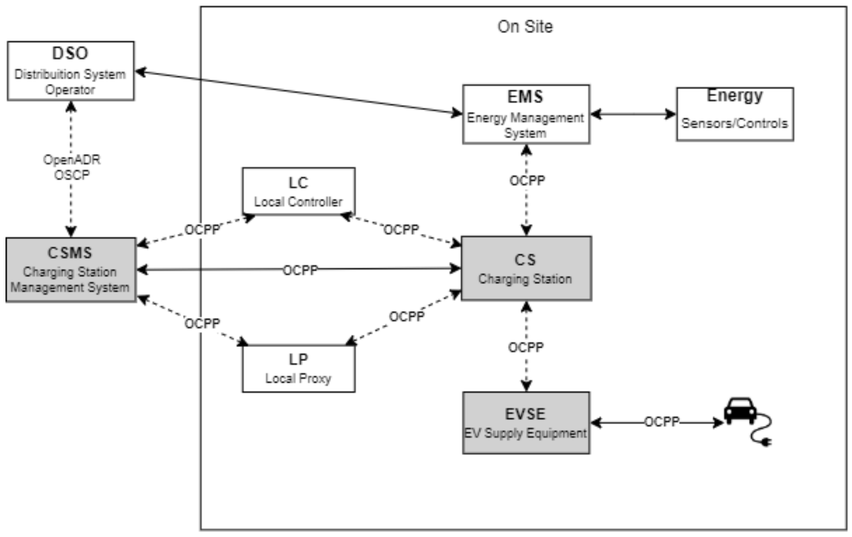


Figure 1: Architecture of an EV charging network using OCPP. Adapted from [3]

In figure 1, located above, the architecture of a OCPP complainant network is shown, along with its components and interactions between them. The descriptions of the central role for each component are described in the next page.

- Charging Station Management System (CSMS): Communicates with the CS and EVSE, collecting information about the power grid status and charging system data. The CSMS also defines the service's parameters and contains authentication information used to authenticate a user at a charging station, also maintaining a booking registry for the service. Additionally, the CSMS can define custom charging profiles to implement when using smart-charging policies, supported by OCPP;

- Charge Station (CS): A charging point is a physical system where one can charge an EV. This component includes the EVSE and multiple EV power connectors per charging point. The control of power limits and state of the CS is done by the CSMS, along with the billing, transaction requirements and authentication. However, the CS can perform user authentication without consulting the CSMS, for example, when the connection is not possible, as it maintains local authorisation lists and cache with user credentials. As for the charging process, the CS has full control, enforcing the limits imposed by the CSMS;

- Electric Vehicle Supply Equipment (EVSE): Consists of a core subsystem of the CS, which allows the exchange of information between the EV and CS, collecting data from the EV regarding the charging process and connectivity status;

- Local Controller (LC): Ensures the control of one or several CSs, setting the charging limits of the CS when the communication between the CS and CSMS is interrupted. It enables the distribution of the requirements and control procedures implemented by the CSMS, facilitating the communication between the CSMS and CS. Using LCs is optional in an OCPP charging network;

- Local Proxy (LP): Component that acts as a router, controlling the flow of messages between CSs. It is an optional component mainly used when the CS is in a location with limited network access so that it can communicate with the LP as if it was the CSMS, as it relays messages;

- Distribuition System Operator (DSO):Utilises EV's data feedback to ensure balance in the power grid by allowing or prohibiting power flow to the charging sites. Usually, this is a third-party component or entity that manages the national power grid of a country. As charging stations put an extra strain on power grid networks, the CSMS can communicate with these entities using Open Automated Demand Response (OpenADR) to control the flow of power supplied to a station, doing so only when required;

- Energy Management System (EMS): Controls the charging process by evaluating energy data generated by a charging EV, enabling energy usage from other sources like renewables or the power grid as required. Commonly renewable sources, from where the EMS allows power draw, are installed on-site, nearby or in the CS.

2.1.3 Open Charge Point Protocol Charging Transaction

A charging transaction occurs when a user has first been authenticated by the CSMS or the charging station itself, allowing him to charge his vehicle and initiating the energy transfer process. To understand how one of these transactions occurs and what information can be transmitted, for this example, we will consider a simple charging network with only the core components.

The authentication phase starts when the user intends to charge his vehicle by presenting or using any means of identification, such as a mobile application, Radio-Frequency Identification (RFID) or contactless card. The charging station proceeds to send an authentication request to the CSMS, which will analyse the identification information provided, and, if valid, authorise the transaction. This authentication can be done by the CSMS, or in case the station is offline, it can be done by itself using the records of users kept in the *Local Authorisation List* or *Authorisation Cache*.

After the approval, the charging procedure can begin. At this stage, the charging station sends to the CSMS a start transaction request, with information such as the connector identifier (number of the plug attached to the vehicle), user identifier, a timestamp, the value of the meter in Watts per hour (*Wh*) at the start of the charging process and an optional reservation identifier. If the user wishes to finish the charging process by unplugging the vehicle, a new authorisation request is sent to the central system to check if it is the same user that started the process. If confirmed, the CS sends a stop transaction request once the user is verified, containing the transaction identifier, an optional identifier used for authentication and the meter values at the end of charging. During the charging process, the CSMS also receives transaction values like the energy imported by the EV in *Wh* sampled from time to time are sent in *MeterValues* requests. Additionally, depending on the implementation, an optional field can be included in the *StopTransaction* request, containing the reason why the transaction stopped. The reason consists of a string whose possible values can be: *EmergencyStop, EVDisconnected, HardReset, Local, Other, PowerLoss* and *Reboot* [4].

The requests sent between the charging stations and CSMS contain valuable information which we can use to detect patterns that could indicate an anomaly is occurring. For example, if there are missing heartbeat packets which are periodically sent to the CSMS, if a charging station is not in use, this could indicate that the station is down. Several authentication packets corresponding to multiple authentication attempts could, for example, imply the execution of a cyberattack and several finished transaction packets with the same reason value could denote an internal communication error or even a hardware error. More specifically, for charging transactions, the stop reason and stop event actor values allow us to quickly determine if a transaction occurred as expected if both values are, for example, *Local* and *station*, respectively.

2.2 Anomaly Detection Techniques

Inside the field of anomaly detection, we can apply several techniques. These techniques vary according to the type of learning paradigm the problem poses, but usually, for anomaly detection, supervised and unsupervised learning are the most common. Nonetheless, semi-supervised learning is also particularly present in literature commonly related to optimising a supervised or unsupervised learning task. Although clustering, tree-based and density-based techniques are still very relevant in the scientific community, it is worth mentioning neural network techniques, as their impact has grown, with the use of Autoencoders (AE) and Generative Adversarial Networks (GAN), both considered unsupervised learning techniques even though they can incorporate some supervised learning characteristics.

Supervised Learning

In supervised learning, applied to anomaly detection, the data is usually labelled, meaning that for each row of information, there is a feature which indicates if the case the row represents is an anomaly or not. Supervised learning datasets, therefore, include a real classification value, which is valuable when the objective is to create models capable of classification. This strategy can more accurately detect anomalies mainly due to using previous knowledge, as one can compare the output label to the real one. Using past knowledge, we can calculate confidence values and several evaluation metrics, such as accuracy, precision, recall and many others. Supervised algorithms are also more appropriate to use with large datasets, obtaining better results the more extensive the dataset is, as there is more information, giving a bigger chance to model the problem accurately. Although it could be advantageous to use this learning paradigm, the lack of labelled data and unbalanced classes on anomaly detection problems means that it is rarely applied.

Some ordinary algorithms used for a supervised learning context include Support Vector Machines (SVM), Multilayer Perceptron (MLP), Decision trees and Linear Regression.

Semi-Supervised Learning

Semi-supervised learning is a technique that aims to combine two learning paradigms, supervised and unsupervised learning. The combination of these two paradigms comes from the fact that a portion of the data used to train a model is usually labelled, corresponding to the supervised part, while most of the remaining data is not labelled at all, corresponding to the unsupervised part.

Using a semi-supervised approach for anomaly detection means that a dataset would include a small percentage of data, labelled, for example, normal or abnormal, with the remaining portion of data being unlabelled. One of the classic methods of conceiving a model using this learning paradigm for classification involves training the model over the labelled input data so that the model can distinguish the anomalous cases from normal ones. Using the model trained over the labelled data, we now classify the unlabelled data, using the predictions the model has given for each unlabelled case as if they were their real labels. Joining the previously unlabelled cases with the true labelled ones, we can devise a new dataset that, when used to retrain the model, can lead to improvements in the initial classification performance, as it now includes more labelled data than the set used in the first iteration of the model.

In the case described in the previous paragraph, the optimisation process is directed to a supervised learning algorithm, as the main objective is classification. However, we can also apply semi-supervised learning in order to improve the performance of an algorithm executing an unsupervised task. The existence of a small percentage of labelled data can help to more accurately describe the distribution and similarity between each case, improving the clustering of the unsupervised portion of the data by providing some indication of which clusters should exist based on the different classes present in the labelled data [5].

Unsupervised Learning

As stated before, when it comes to unsupervised learning, no feature or label indicates what a row represents. Considering that there is no information if a row of data is an anomaly or not, usually, unsupervised algorithms group rows based on similarity values, signalling those with abnormal values, often also called outliers, which deviate from the general distribution and are characterised, in anomaly detection, as having a low occurrence rate. In this field, unsupervised learning strategies are frequently used, as most of the data available is not labelled and manually labelling data is costly and time-consuming.

Although clustering is the most common method, it is not the only one that applies to anomaly detection. In the following sections, we describe some of the most common anomaly detection techniques, mostly related to unsupervised learning, as it is the most common approach encountered for anomaly detection. Despite not being present in the following sections, it is worth mentioning techniques such as Bayesian Networks, Hidden Markov Models and Fuzzy Logic-based outlier detection, as there are also studies in this field that still apply techniques.

2.2.1 Cluster Analysis Based Techniques

Clustering is a technique based on the assumption that, in a dataset, most normal data points are located close to each other in large clusters, whereas anomalous data points can usually be found in a smaller cluster, far away from the closest cluster or do not belong to any cluster. Cluster Analysis Based algorithms allow for unsupervised outlier detection, which is a good advantage as there is no need for labelled data which is scarce when it comes to anomaly detection. Additionally, the clustering process is usually fast as the number of clusters identified in a dataset tends to be smaller when compared to the size of the dataset. However, as the data set increases, so does the time complexity and resource usage from these algorithms being the main disadvantage when applying these methods and the frequent target of several studies that try to optimise the performance and resource usage of these clustering techniques.

An example of an algorithm that implements this technique is the K-Means algorithm. The main goal of this algorithm is to group similar data forming clusters of data points and attributing a given data point to the cluster with the nearest mean. To better understand how the algorithm works, the following figures illustrate the algorithm's execution.

The algorithm starts by initialising and placing centroids, which depict the centre of a cluster, assigning all data points to the closest centroid in the first iteration. Initialising and positioning the centroids is a process usually done randomly, but we can configure it to use other approaches. In figure 2, two centroids were randomly positioned, represented by the circles with a blue and red outline, where the black dots represent data points from a given dataset. In this case, the dataset was randomly generated, and a simulation tool was used to run the K-Means algorithm.

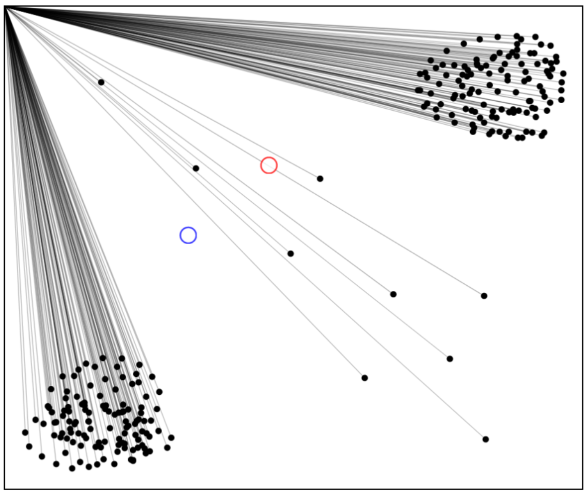


Figure 2: Random initialisation of centroids [6]

After this first step in the first iteration, the data points are attributed to each centroid. The centroid's location is then updated based on the mean values of all data points attributed to it, pushing it further to the centre of that cluster. This process of updating the centroids is repeated several times until the centroids don't move anymore, and all data points are attributed to the closest centroid.

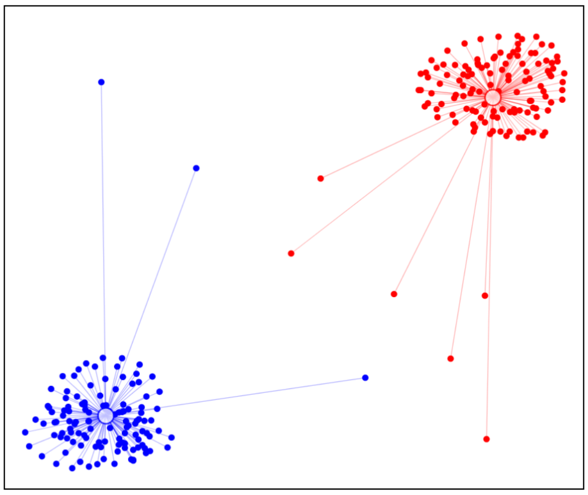


Figure 3: Centroids converge on the centre of their cluster [6]

As it is possible to see in the figure above, because the centroids converge in the centre of the cluster, abnormal data points are easily detected by measuring the distance from their centroid. In figure 3, we can easily visualise what would be considered anomalies in a real data set, by looking at the most distant points coloured the same way as their centroid.

The K-means algorithm is very versatile and with its use comes some advantages, like scalability as it works with larger datasets, guarantees convergence, can easily be adapted to new examples and clusters can be shaped in any way, as points are attributed based on their distance to the centroid not being as limiting as other methods. Although this is true, there are some aspects to consider when using this algorithm, one of which is that the centroids can be moved by outliers as they are placed. In a dataset with an extensive percentage of outliers, clusters might reposition in their direction, probably leading to undesirable results. Additionally, although it works well with larger datasets, it does not scale well with a higher level of dimensions, and the number of clusters has to be manually set [7].

2.2.2 Tree-Based Techniques

Although tree-based techniques are often associated with supervised learning, with algorithms such as Decision Trees and Random Forests, some algorithms apply the principles of this technique for unsupervised learning. A tree-based technique functions by repeatedly splitting the input values into smaller subsets building the decision tree, and defining the rules for separating the values in each node of this tree. In literature, this technique is frequently applied, as it is simple to understand, unlike, for example, neural network-related methods, providing explainable models which allow one to follow the decision process for a particular prediction. One example of a popular algorithm that applies this principle for anomaly detection is the Isolation Forest algorithm.

In December 2008, the eighth edition of the IEEE International Conference on Data Mining occurred.

At this conference, a new method of anomaly detection was introduced based on measuring the distance between data values when organised in a tree structure, promising a linear time complexity and low memory requirements, something other methods at the time could not do. At this conference, the isolation forest or *iForest* algorithm was presented, designed with anomaly detection as its core objective. This particular algorithm works by measuring the isolation value between several data points to determine if a given case represents an anomaly, depending on its distance from the rest of the data. At the time of development, the most common anomaly detection techniques modelled the non-anomalous data points instead of the abnormal data points, commonly having some drawbacks when applied in anomaly detection, such as causing too many false alarms and not being very computationally efficient when used with larger datasets [8].

This algorithm starts by selecting a subsample from the data and randomly selecting a value serving as the root of a tree. After this step, the subsample values are split to the left and right of the chosen value, meaning that the ones lower than the root value are appended to the tree on the left and higher values on the right, building the Isolation Tree. This splitting process repeats for each child node on the tree until it can no longer be applied. Considering that the non-anomalous cases share similarities and their values are in close proximity to each other when organised in a binary tree, anomalies are susceptible to having shorter path lengths inside the tree which means that when added to the tree, a value of an anomalous case, usually it will sit at a higher level and the splitting based on this value node won't be possible.

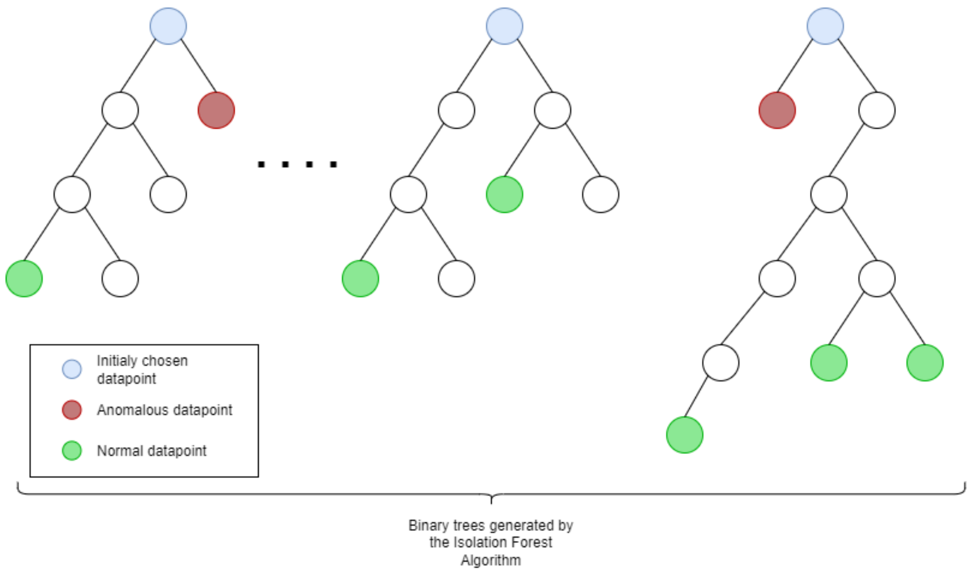

Initialy chosen datapoint
Anomalous datapoint
Normal datapoint
Binary trees generated by the Isolation Forest Algorithm

Figure 4: Example visualisation of binary trees generated by the isolation forest algorithm. Adapted from [9]

As figure 4 shows, the isolation forest algorithm generates several binary trees, each corresponding to a subset of the data, also called partition. Each tree is built by choosing a random data point of this partition to which the algorithm tries to append other values, assembling a tree like the one shown. Here the process of creating the tree is based on the same principles as the decision trees algorithm by selecting a value and comparing that same value to the ones already present in the tree.

In the first tree, located on the left in figure 4, the value shown as a red node was chosen among the partition's values. As no other value could be appended to this one because another exists closer to the one selected, the tree grew to the left of this node. Considering the fact that the objective of the algorithm is to try to isolate a randomly selected data point in a partition, the fact that the red node was isolated with fewer steps means this value is likely to be an outlier since further division is not possible and is found at a higher level in the tree (close to the root). With non-anomalous points, on the other hand, the division is often possible, resulting in these usually being located in lower levels of the tree, as it is possible to see by the green nodes in the same figure. With this process and the generated trees, we can calculate an anomaly score, a number denoting how likely it is that some point could be an anomaly based on the isolation of its values when compared and organised in a tree structure with others.

In recent years, several studies have focused on minimising multiple issues that the isolation forest algorithm poses. One of these issues is related to anomaly score attribution, which in datasets with high complexity, can lead to imprecise score values and eventually cause false negatives, not detecting anomalies as they fall under standard score values where in reality should not. This fact proved to be the central motivation behind the creation of the Extended Isolation Forest (EIF) algorithm, where an article published on the 6th of November 2018 proposed a change to the original isolation forest algorithm. In this article, Hariri et al. (2018) suggested two different methods of minimising the issue, the first one by randomly transforming data before creating isolation trees, averaging the bias effect of the algorithm. The second one by introducing the ability to separate data with the use of hyperplanes and random slopes, allowing new separation methods not just using standard functions, which in the original implementation of the algorithm, create vertical and horizontal lines that are used in the isolation process of a data point, assembling the partitions, as referenced before in this section. The application of the algorithm developed by this study showed that a more uniform anomaly score attribution led to increased accuracy of the EIF algorithm when applied to several data sets and compared to the original implementation [10].

2.2.3 Density-Based Techniques

This kind of anomaly detection technique works by calculating the density of data points in a radius around each point. Typically, a data point considered anomalous occurs when the number of neighbouring points in its radius is undersized compared to that of its neighbours [11]. The main advantages of using these types of techniques include the ability of these algorithms to identify clusters with arbitrary shapes, and there is no need to decide the number of clusters as an input, which occurs with algorithms based on cluster analysis techniques. These are aspects to take into account, as some datasets can have data points in clusters that are not regularly shaped and exist in higher numbers.

As for algorithms, many examples implement this technique, like K-Nearest Neighbors (KNN) and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), which we will analyse in the following paragraphs.

DBSCAN was introduced initially by Ester et al. in 1996. DBSCAN is capable of identifying outliers by analysing the density of neighbouring points of a given data point and checking if it falls within the defined minimum points threshold and radius. According to the literature, there are four categories of data points in the cluster discovery process. The first is the core point, which corresponds to a data point with the minimum number of other points present within its radius. The second one corresponds to a data point situated in a core point's radius, called direct density reachable. Points that are reachable through a path constructed from the initial core point through other core points are referred to as density-connected, and finally, points that are not accessible from any point are called outliers.

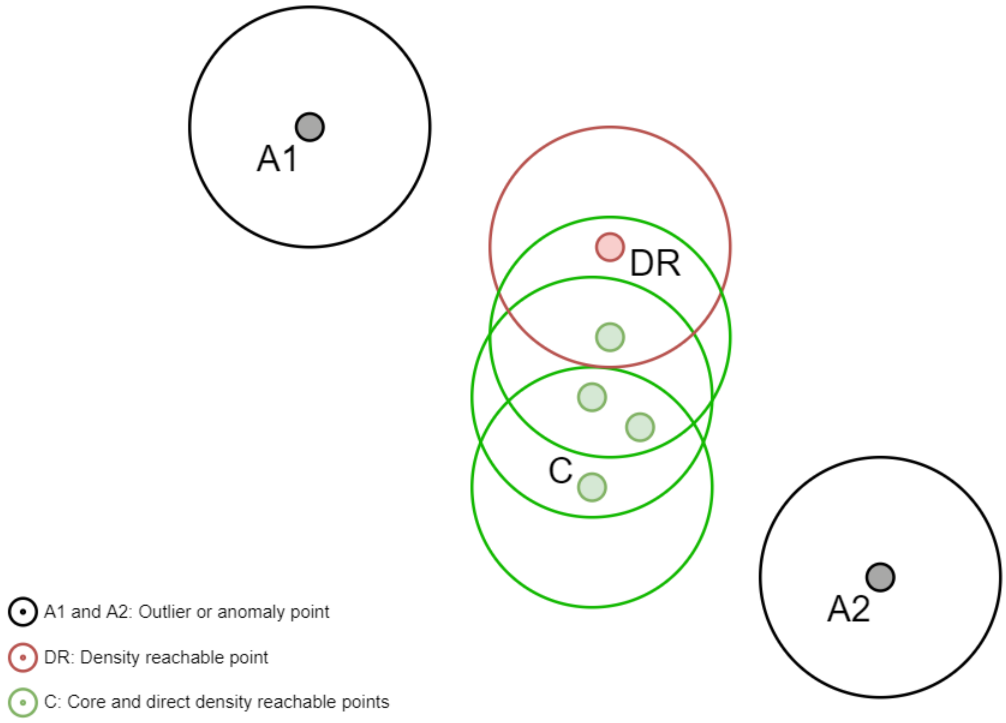

A1
DR
C
A2
A1 and A2: Outlier or anomaly point
DR: Density reachable point
C: Core and direct density reachable points

Figure 5: DBSCAN visualisation. Adapted from [13]

Figure 5, located above, demonstrates these four categories of data points, where point C is a core point, DR are density reachable points as there is a path from C to this one composed from other core and direct density reachable points, and finally, A1 and A2 which cannot be reached and represent outliers.

Regarding the execution of this algorithm, the process is iterative and starts by randomly selecting an initial data point, looking for other neighbouring points in its radius. Any points present in the initial point's radius are then added to the cluster, repeating the neighbour check process for each newly added point in their respective radius, adding them to the cluster if the minimum number of neighbours is respected. When there are no more points to be added to the cluster, the whole process repeats by again randomly selecting a new initial data point, and if we can apply the same process, meaning this initial data point has sufficient neighbours (defined by the minimum neighbours' threshold parameter), a new cluster is created, and so forth. If the newly chosen initial point does not have enough neighbours, it is considered an outlier.

In the literature, we can find several applications of this algorithm for anomaly detection in several fields. One of these is in weather prediction, where the algorithm can be used to detect extreme temperature changes in factories and any business where automatic manufacturing exists, for example, detecting anomalies in machinery through values collected by sensors, city traffic anomaly prediction and even for aviation.

Apart from the studies that cover specific applications, several other studies aim to improve the algorithm's performance by adapting the way DBSCAN works and creating new versions, like the KNN-Block DBSCAN by authors in Y. Chen et al. (2021). This adaptation of DBSCAN aims to improve the algorithm's performance when applied to larger datasets by using the FLANN algorithm to automatically detect and divide the data into three types of blocks, core-blocks, non-core-blocks and noise blocks. A core-block will correspond to a given cluster to which points will be appended from a non-core block if they are density-reachable from an existing core point of that cluster. Core-blocks can also be merged in the same cluster if they are density-reachable from each other, and noise blocks are discarded. This division in blocks leads to a reduced number of computations, as the original DBSCAN computes distances between points on the entirety of the dataset, using a brute force approach, as stated in the study.

2.2.4 Support Vector Machines

As the name suggests, this technique uses Support Vector Machines (SVM) to detect anomalous points in a dataset. The main objective of a support vector machine is to learn a function that describes a hyperplane dividing the data points and creating a decision boundary where data points on one side get classified as one class and data points on the other as another.

One of the most common approaches for anomaly detection using SVMs is One Class SVMs. The principles of a One-Class SVM derive from the standard SVM, typically used in regression and classification problems. These two operate similarly to create a hyperplane that separates data points. However, in the case of the standard SVM, the data points are separated by attributing two distinct classes to values that fall on the left and right of the learned hyperplane. Regarding the One Class SVM, on the other hand, it aims to perform this separation using only one class, where the values inside the boundary of the hyperplane are considered normal, and the values outside are abnormal, representing anomalies.

The SVM algorithm dates to 1995, when the first paper addressing this method was published, referencing an algorithm called Support-Vector Networks [15], the basis of what is now called a Support Vector

Machines (SVM). A SVM algorithm works by trying to learn a mathematical function that represents a hyperplane when working with three dimensions or a line in two dimensions, where in both, the hyperplane represents the decision boundary between the existing classes in a dimensional space.

In a classification problem, in a two-dimensional space, for example, where we want to classify several data points in two distinct classes, the SVM algorithm would search for the best hyperplane that could divide the data points between the two classes. In order to choose the best hyperplane, the algorithm tries to maximise the distance between two data points that serve as the edges of the decision boundary, referred to as support vectors. Iteratively, the algorithm attempts to divide the data points with different hyperplanes, evaluating all the options and eventually choosing the one which maximises the margin, which is the distance between the support vectors. The larger the margin that divides the two classes, the more guarantees it provides that the classification of new cases will be as precise as possible based on the previous knowledge used. We can visualise this in figure 6, located below.

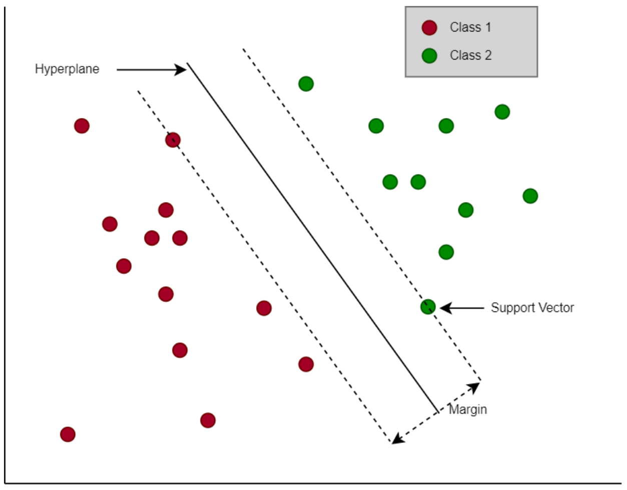


Figure 6: Visualisation of a hyperplane separating two distinct classes of data points. Adapted from [16]

Although this works for problems like classification and regression, when operating with unlabelled data, we can no longer split the data points based on classes, as no actual labels are provided. In this case, as applying the same principles of a standard SVM to create a line or hyperplane that can separate the data points is impossible, we must use a polynomial kernel. In simple terms, employing this kernel adds a third dimension, creating a feature space, where a hyperplane can separate cases by maximising the distance between itself and the feature space origin. This new dimension originates from calculating the similarity between all dataset cases.

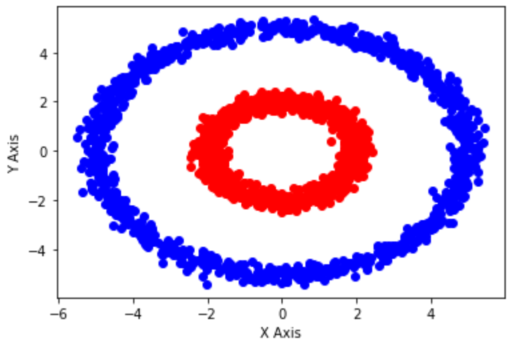


Figure 7: Plot of original data in two dimensions. Adapted from [17]

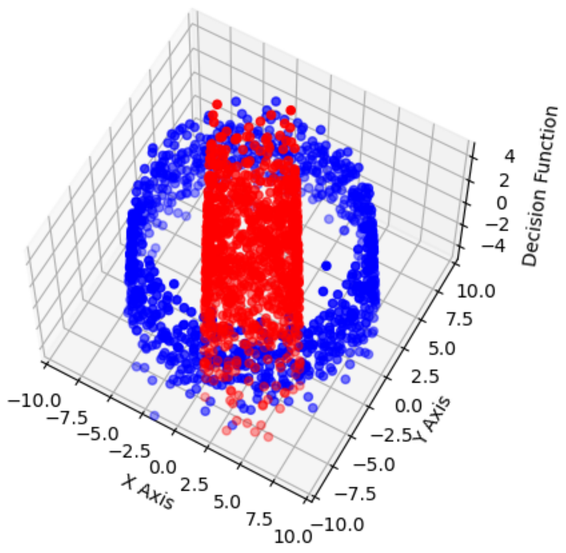


Figure 8: Data after the application of the polynomial kernel. Adapted from [17]

As it is possible to see in the example given by the figures above, in figure 7, we cannot draw a line to separate both data clusters, but by adding a third dimension, as a result of the polynomial kernel application, a separation limit that divides these two clusters of data, can now be drawn, as seen in figure 8 This process is often referred in literature as the kernel trick.

When projected back to the two-dimensional plain, the function generated by this process, which describes the hyperplane, captures a circular area representing the centre where most data points are located. Identifying anomalies is now possible as for data points that reside outside the boundary, the function will return a value of -1, and for points inside the boundary, a value of +1. This method of adapting the SVM algorithm, proposed by Schölkopf et al. (1999), proves the usefulness of this method for unsupervised learning and anomaly detection techniques. In figure 9, we can visualise an example of a generated boundary that would be learned by this algorithm [18].

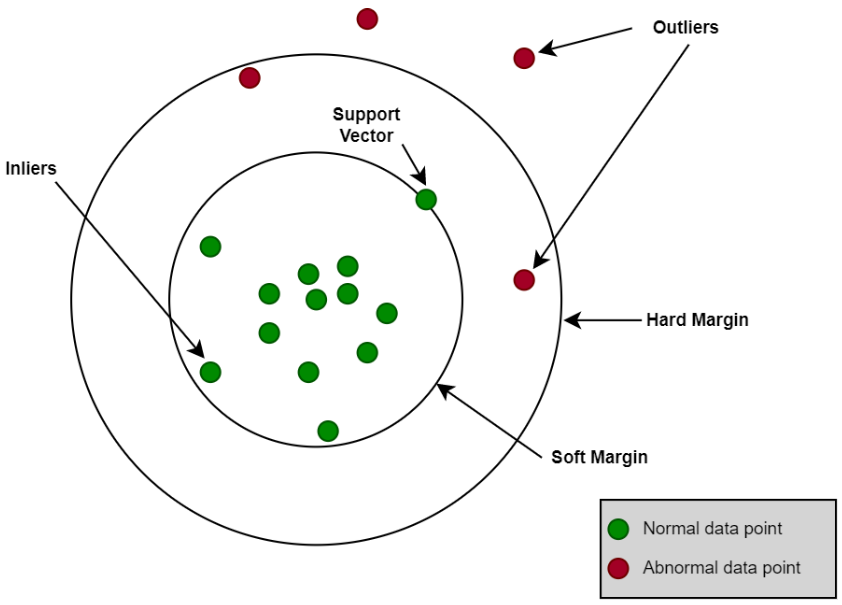


Figure 9: Boundary projection generated of a hyperplane. Adapted from [19]

The ability to create a separation plane that can be a good fit for nonlinear boundaries is an advantage when compared to other methods capable of clustering data. Although this is true, this algorithm also poses some disadvantages, mainly related to scalability. As the size of data used to train the model increases, so does the computational resource usage and time needed to perform a new prediction. Recently in literature, some studies have tried to reduce the computational needs and the time required to predict the occurrence of new anomalies of a model using this algorithm. These efforts concentrate efforts on improving the speed at which operations can be done over the *gram matrix*, as referenced by authors in Yang et al. (2021), as this is usually the bottleneck when performing new predictions using this algorithm, and it contains the encoded relationships between data points [20].

2.2.5 Neural Networks Related Techniques

With the increased success in supervised learning problems, several studies have tried to create or improve existing approaches using neural networks, some conceiving new models with applications for unsupervised learning problems, such as anomaly detection. Some of the most common strategies are branched under the deep neural networks, including Autoencoders (AE), Long Short-Term Memory Networks (LSTM) and Generative Adversarial Networks (GAN).

A deep neural network consists of the implementation of a neural network with one or more hidden layers. In this branch of neural network techniques, some approaches are conceived with a different purpose than others, some without even anomaly detection in mind. However, several studies and tests have shown promising results when conceptualising models for anomaly detection using deep learning approaches and algorithms. Among these, and as referenced previously, are AE, vastly utilised and present in literature. Although mathematical theories date back to the 1980s, autoencoders have, in the past years, shown promising results as a deep learning technique for dimensionality reduction, image compression, noise reduction and even anomaly detection.

AE consist of a deep neural network which aims to learn a lower set of dimensional features that describe the initial input provided to the encoder. This set is usually called *code* or the most common designation in literature, a latent-space representation. From this compressed description of the input data, the decoder tries to reconstruct the initial input, from which we can calculate the reconstruction error, comparing the reconstructed output with the original input. In anomaly detection, if we train a model to reconstruct normal data instances, this error value will rise when applied to abnormal cases, allowing their identification.

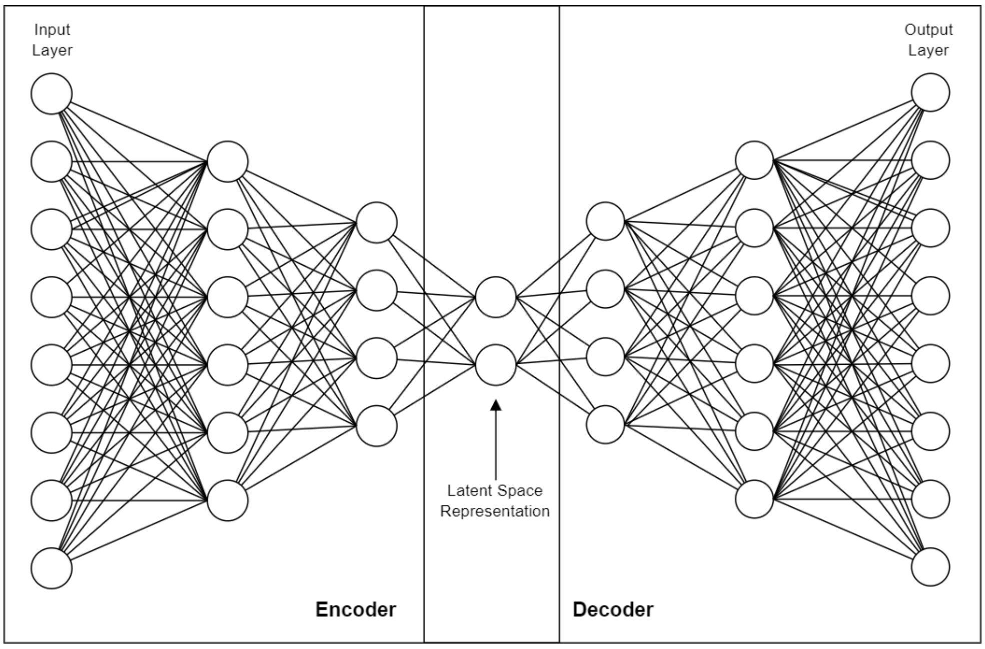


Figure 10: General structure of an AE

Figure 10 shows a general structure of an AE. This structure can vary as there are several types of implementations for different applications, which, in turn, can have a different architectures. Although a configuration like the one shown could be for unsupervised anomaly detection, the training process of an AE would be a supervised task, as we would train the model using only the data labelled as *normal*, teaching the decoder to recreate these samples of data, which would lead to an increase in the output error when abnormal data is used as an input, as referenced before. This increase happens because the model has not learned how to reconstruct something aside from a normal data sample.

Compared to other methods and techniques for anomaly detection, autoencoders usually provide more accurate results as one of the requirements is a set with the normal data instances used to train the network. This is one of the disadvantages of this method as if this set does not exist and the available data is all unlabelled, we can not apply this technique. Despite this and the fact that for most anomaly detection problems, there is a lack of labelled data, this is still a widespread technique in literature, with several studies choosing this as the preferred technique to apply to anomaly detection problems. Some of these problems include the detection of cyberattacks in wireless communication, or even anomaly detection over data from industrial plants by combining autoencoders with frameworks that allow for explainable anomaly detection models.

When applying autoencoders and comparing these to other relevant methods in the literature, authors in Z. Chen et al. (2018) [21] created two autoencoder base networks, one consisting of a standard autoencoder and another of a convolutional autoencoder network. Both of these were applied to a dataset containing network packets classified as *normal* and others as *attacks*, revealing a superior performance when compared to other algorithms like K-Nearest Neighbours and SVM and only slightly below the implementation of the Triangle area based Nearest Neighbours. Although autoencoders and neural networks, in general, have shown promising results in several areas, the main disadvantage when using these methods is the absence of explainability for a given prediction, often offered by other methods. By applying state-of-the-art frameworks that help to explain which features were considered the most important for a prediction, such as SHAP or Custom Local Explainer (CLE), the authors in Tripathy et al. (2022) [22] were able to obtain critical information that allowed them to explain a given prediction based on the value of each feature used by the autoencoder network.

2.2.5.1 Variational Autoencoders

In recent years, several studies in literature have approached a new technique derivative of autoencoders, which is gaining increased popularity in the field of image, sound and text generation, sometimes referenced along with GAN, called Variational Autoencoder (VAE). Not to be confused with standard autoencoders, VAEs are deep generative models, which means the focal point is usually to create new data from the compressed data generated by an encoder in the latent space, whereas a standard AE attempts to use the latent space representation to reconstruct original input and minimise the reconstruction error, being the main objective to generate an output as close to the actual input as possible.

As the purpose of VAEs differs, their structure and operation are also different. In a standard AE, the encoded representation of the data or, as often called, the latent space, would be entirely used by the decoder to try and reconstruct the input. In contrast, in a VAE, the latent space contents are sampled by only selecting a partition of this data to submit to the decoder, as shown in figure 11, located on the top of the next page.

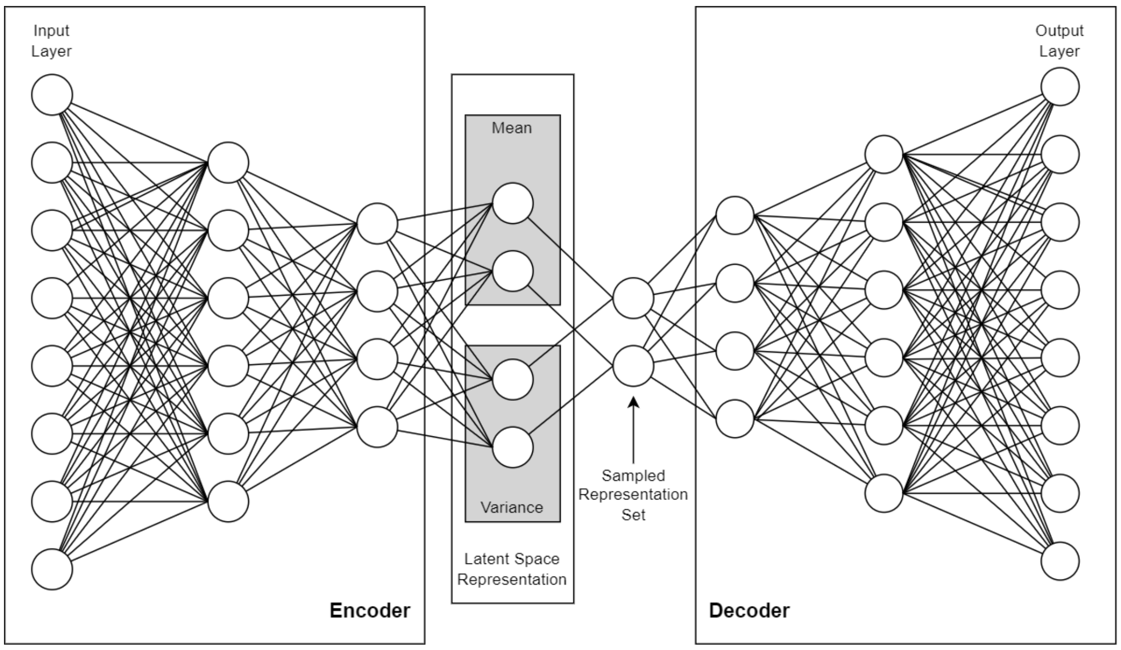


Figure 11: Structure of a VAE. Adapted from [23]

By sampling values from the encoded input, we obtain an altered and more diminutive representation of that initial data. When providing this data to the decoder for reconstruction, the outcome will not be a close representation of the input, as in an AE, but a more distant representation based only on a sample from the original encoding, which leads to often different results. As the main objective is not reconstruction but generation, this method is usually not suitable for anomaly detection, although some articles suggest that in some cases, the regularisation imposed over the latent space can lead to more accuracy. This accuracy increase transpires from the fact that, because the regularisation process is applied over the covariance matrix and mean of the distributions returned by the encoder, distributions are enforced to be close to a standard normal distribution [24]. This process, in turn, leads to similar points being close to each other, improving the measurement of similarity according to Diederik P. Kingma and Max Welling (2013) [25] and Alon Agmon (2021) [26].

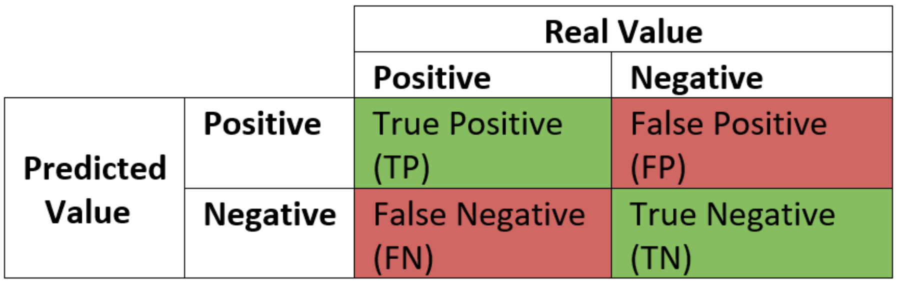
Although not yet thoroughly studied how reliable a VAE method can be for anomaly detection, there are concrete implementations, for example, for health and the pharmaceutical industry, like the one described by authors in Hadipour et al. (2022), where the usage of a VAE in combination with the K-Means algorithm showed an increase in clustering performance and value over other methods. In the study, the authors considered this strategy the most promising, as it was able to not only generate accurate representations of molecules but also enabled a better clustering by leveraging the latent space representation generated by the VAE.

2.3 Model Evaluation Metrics

When it comes to evaluating the performance of a machine learning model, there are several metrics one can use. The application of these metrics can vary according to the learning paradigm used, the algorithm and the scope of the problem. For some areas, a given metric might not be accurate enough in describing the model's performance, usually relying on the combination of several metrics to precisely evaluate a model.

2.3.1 Supervised Learning

As mentioned before, in supervised learning, the data is usually labelled, allowing for a distinction between the predicted values of a model and the real values it should predict. As we know the actual labels we need to predict for problems that reside inside the supervised learning field, we can create a confusion matrix which describes the distribution between the true positives and negatives and the false positives and negative values, as presented in figure 12.



		Real Value	
		Positive	**Negative**
Predicted Value	**Positive**	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

Figure 12: Confusion matrix

Accuracy

One of the most common ways to evaluate classification problems is the accuracy value, calculated by dividing the true positives (TP) plus the true negatives (TN) by the sum of the true positives and true negatives with the false positives (FP) and false negatives (FN).

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{1}$$

This results in a proportion that demonstrates the percentage of true results over the entire result set, predominantly used for problems where the dataset is balanced, as an imbalanced dataset can easily modify this value, and does not account for the chance of a model overfitting and reproducing exact values of the dataset. In this situation, a model can report a high accuracy but not have any value when applied to new data, as we will not verify the same accuracy.

Precision and Recall

Another classic evaluation metric is *precision*, which is the result of dividing the TP by the sum of the same value with the FP. This metric is usually referenced together with *recall*, which has a similar formula with the only difference being that, instead of using the number of FP in the sum, we use the FN.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

The difference between the *precision* and *recall* metrics is that precision outlines the proportion of the predicted positives that are truthfully positive, whereas recall expresses the ratio of the true positives that were successfully identified. In simple terms, the *precision* tells us how the model behaved in correctly predicting the class of something and *recall* how many of those were correctly identified.

F1-score

The combination of the *precision* and *recall* metrics leads to a new one called *f1-score*, which combines both properties of those two metrics.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

This metric is more valuable than *precision* and *recall* on their own. Usually, one would want a model with a high *precision* and *recall*, as that means that the model is capturing the problem correctly, identifying a class while capturing most of its occurrences on the dataset. This metric can also be used to prove the model's accuracy metric, for example, if a dataset is not balanced and contains only two classes where one corresponds to 90% of the cases and the other only 10%, if the model guessed the same for every case with the highest occurring class, the accuracy value would be high as most of the cases in the dataset are of that class. In this case, the *f1-score* metric is applied for the lowest occurring class, both *precision* and *recall* would be 0, thus showing that the accuracy value is not confirmed as it underperforms for one of the classes [28].

Log Loss

$$L_{log}(y, p) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{5}$$

Log loss, also known as cross-entropy loss, allows measuring a classification model's performance by outputting a probability value between 0 and 1. In order to calculate this value, the function takes into account the true label, y, and the probability estimate, p, that the model has computed. The furthest the probability estimate value (p) is from the true value (y), the higher the loss value outputted. This function is furthermore often used in logistic regression models or neural networks as the optimisation target, as lowering the loss values yields better model accuracy in supervised models that output a probability estimate or binary results.

Area Under the ROC Curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of a model's performance by plotting the True Positive Rate (TPR), often called sensitivity, with the False Positive Rate (FPR) or one minus the specificity [29].

$$Sensitivty = TPR(TruePositiveRate) = Recall = \frac{TP}{TP + FN} \tag{6}$$

$$1 - Specificity = FPR(FalsePositiveRate) = \frac{FP}{TN + FP} \tag{7}$$

The Area Under the ROC Curve (AUC) corresponds to the two-dimensional area measured under the ROC curve, which, in turn, can be used to estimate a model's performance. For a given set of cases, if a model has a low AUC, it is incorrectly classifying most of these cases, however if the AUC is high, the classification is being performed correctly for most of the observed cases in the dataset [30].

2.3.2 Semi-Supervised Learning

As the name suggests, semi-supervised learning entails using both supervised and unsupervised learning. Applying this paradigm to a classifier, for example, means that a model might initially be trained over labelled data and later be used to classify the unlabelled portion of data. The predicted labels for the unlabelled data are added to the corresponding unlabelled cases and later mixed with the original labelled cases, creating a new dataset used to train the model again, hoping to improve that model's performance.

Considering that a classic use of this paradigm is to improve the performance of classification models where high amounts of labelled data are unavailable, it is possible to apply the same metrics used for a supervised learning algorithm described in the previous section. For research purposes, a common way to prove the increase in the model's performance is to remove labels from a fully labelled dataset, leaving only a short amount of the original labelled cases. By knowing the real labels for all cases, one can still apply this process and evaluate the final results with supervised learning metrics, therefore measuring the performance gain resulting from this paradigm's application.

However, this is not common in real applications where we might not know the true label for every instance of the dataset and thus can not measure the performance obtained with the same precision. If there is no way to determine the actual label or class of the unlabelled instances, there are no guarantees that a given evaluation metric yields accurate results for the model's performance. Although this is one of the several factors to consider that can influence the values obtained by evaluation metrics, we also ought to assess if the number of labelled cases in the dataset is enough for the model to capture the differences between each class. If it is not enough, training the model while adding the unlabelled data might lead to worse performance instead of enhancing it [5].

2.3.3 Unsupervised Learning

When it comes to unsupervised learning, as there is no label to indicate if, for example, a dataset sample belongs to a given cluster, we must approach the performance measurement differently. When evaluating the performance of a model that clusters data, for example, problem or domain knowledge is crucial, as there is a need to understand if the created clusters and similarity associations are correct and hold up in the problem's domain. Therefore a metric that mathematically represents the performance of an unsupervised model and holds for every context does not exist. Although this is true, there are several methods that help in measuring the performance of the clustering process, some using previous knowledge in the form of ground truth class assignments or labels, comparing these to the predicted labels obtained by the clustering process, which is similar to what is done in supervised learning. Other methods do not require previous knowledge, but their evaluation capacity is limited, as they can only evaluate how accurate the clustering process was by measuring the density and separation between clusters.

Rand Index

The rand index is one of the options to consider when we need to evaluate an unsupervised model's performance. The index measures the similarity between the ground truth labels and the assignments predicted by the model. In literature, we often find two implementations of the Rand Index, the Unadjusted Rand Index (RI) and the Adjusted Rand Index (ARI).

$$RI = \frac{a + b}{C_2^{n_{samples}}} \tag{8}$$

When using the RI, the score can range from 0 to 1, calculated by the sum of the number of pairs of elements that are in the same set, both in the ground truth labels and clustering labels sets (a), with the number of pairs of elements that are in different sets both in the ground truth and clustering labels sets (b), divided by the total number of possible pairs in the dataset ($C_2^{n_{samples}}$).

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \tag{9}$$

To the contrary of RI, the ARI's score can range from -1 to 1, accounting for the possibility of random assignments, meaning that there is no assurance that when applying the RI, a value close to 0 represents a random assignment. To achieve this property and calculate the ARI, we must know the subtraction result between the RI value and the expected RI ($E[RI]$) value, divided by the subtraction of the expected RI ($E[RI]$) to the maximum RI value.

When applying each one of the methods, we can interpret the model's performance by analysing the output value. The closest the value is to 1, the more accurately the clustering process was, so the furthest this value is from 1, the worst the model performs.

Silhouette Coefficient

When no ground truth label set exists, the Silhouette Coefficient provides a way to evaluate the model's performance based on the separation and density of the clusters.

$$s = \frac{b - a}{max(a, b)} \tag{10}$$

The Silhouette Coefficient value s can assume any number between -1 and 1, calculated via the subtraction of the mean distance between a sample and all other points in the same class, a, to the mean distance between a sample and all other points in the next nearest cluster, b, divided by the maximum value between a and b. A high output value represents high-density clustering, and a low output value an imprecise clustering of the values.

2.4 Anomaly detection applied to OCPP data

From the research conducted, OCPP data has been used before to detect anomalies in charging networks, with most authors focusing on creating systems capable of detecting cyberattacks or just analysing the charging network structure and the messages exchanged between network components to look for possible vulnerabilities.

Some of these studies propose using machine learning models to predict possible attacks over critical network infrastructure, such as local power grids, from energy theft attacks or even pricing cyberattacks, which are among the most common types of attacks investigated in the literature. Over the years, different possibilities have been considered when it comes to detecting these types of attacks, for example, Y. Liu et al. (2014) [31] proposed the use of support vector regression to analyse changes in the guideline pricing curve, monitoring peaks in the energy load which is usually the objective of hackers when conducting a pricing cyberattack. This type of attack has the goal of increasing costs for the customer who is using the service and straining the charging network. The same authors later in, Y. Liu et al. (2016) [32], proposed an improved approach to reduce and predict the impact of the same type of attack by creating an observable model based on the Markov decision process detection.

This idea of applying machine learning to EV charging networks, specifically ones that use OCPP as the communication protocol, is not recent and has been approached since the initial versions of the protocol. These versions lacked the implementation of several security features, which proved to be the central motivation for researchers to attempt to resolve this, using several techniques, with some applying machine learning, such as the authors referenced previously in this section. Even more than before, cybersecurity is becoming more prevalent as we grow more dependent on automation and the systems around us, which includes the charging stations and EVs themselves. This dependence and the need to secure our communications are some of the main reasons why this topic is so prominent in the literature. Recently, and by analysing several articles available in the literature that relate anomaly detection with machine learning applied to OCPP, we can identify two principal forms of research. In some studies, researchers suggest and try to implement solutions using machine learning models over OCPP data, like in the study described previously, while in others, there is an attempt at underlying the main flaws or challenges that come with the use of the OCPP protocol. In both cases, when it comes to OCPP, the main focus in literature is related to the cybersecurity field, as the protocol is still growing with new implementations of security features to prevent the occurrence of some of the most common cyberattacks.

In the recent article by Garofalaki et al. (2022) [3], the release of OCPP 2.0 is approached, describing the new security features implemented in this version of the protocol, such as secure firmware updates, security logging, event notification, security profiles for authentication and secure communication using the Transport Layer Security (TLS) protocol. Additional to the research conducted in analysing vulnerabilities of the OCPP protocol, the article also describes several types of attacks performed over a charging network, ranging from cyber to physical attacks and a combination of both cyber-physical attacks. In the study, for each attack referenced, the authors clarify how each one could work in a charging network, referencing countermeasures that currently exist in the literature and explaining how these could be applied to prevent or reduce the impact of these attacks. One of the several examples given in the article is a man-in-the-middle attack. The premise of this attack is simple and consists of a perpetrator attempting to intercept communications between two systems or a user and a system. Attackers can do this in two ways, capturing network packets of a conversation, therefore eavesdropping, or engaging directly in this conversation, pretending to be one of the participants. In the network structure, the study references some communication paths where this can occur, such as between the CS and the EV, as the study claims that solely using the TLS protocol is not enough to prevent this kind of attack. The reasons given by the authors are mainly due to TLS not providing long-term authenticity or non-repudiation, introducing overhead and allowing the transmission of proxied data in clear text. Here a suggested solution would be to use a machine learning model, as proposed by Kabir et al. (2021) [33], where the CSMS implements a Back Propagation Neural Network (BPNN) trained to detect man-in-the-middle attacks by analysing charging and discharging requests that occur during a charging transaction.

Upon this literature analysis, it is possible to state that OCPP data has already been used in machine learning projects to try and improve the overhaul network security by attempting to predict cyberattack occurrences. Although this is true, and at the date of writing this document, there is no attempt in the literature to use transaction and connector status data sent between the charging stations and the CSMS to predict possible anomalies and perform maintenance tasks in a shorter amount of time. This is where this study comes in, aiming to fill this gap by conceiving not only one machine learning model capable of doing so but several models, comparing each method and algorithm used to determine which one provides the best results for the problem at hand.

2.5 Summary

With the increase in the number of EVs, the number of CS also increases, introducing a new load in the electric vehicle charging networks. The construction and configuration of these charging stations pose a challenge as multiple vendors produce different hardware used in several charging stations, but ensuring interoperability, compatibility, flexibility, and scalability is demanding without some standard communication protocol that allows the integration of all the different hardware components. This was the aim of the ElaadNL foundation, which created the Open Charge Point Protocol (OCPP), and later saw the foundation of the Open Charge Alliance (OCA) to centralise testing and certification of new OCPP implementations worldwide.

According to the OCA and OCPP documentation, a charging network requires at minimum three main components. The first one is the CSMS which is responsible for depicting the service's parameters, performing user authentication, specifying custom charging profiles for smart-charging policies, maintaining a booking registry for the service, communicating with the CS and the EVSE. The second one is the CS, which allows the management of several charging points, with their status, power limits, billing and transaction requirements controlled by the CSMS. The CS enforces all the limits imposed by the CSMS, for example, over the energy used in the charging process. Finally, the third component is the EVSE, classified as a core subsystem that enables the communication between the EV and the CS, helping to read values correlated to the charging process and connectivity status. Although not necessary, more complex networks can implement components such as the LP or the LC, which aid the communication between the CSMS and CS or the DSO and EMS, which generally assist in the transmission and control over the power supplied to each CS in the network. In an OCPP-compliant network, when a user intends to charge an EV, he must authenticate in the CS. When the CSMS is not reachable, the CS can perform user authentication by using the *LocalAuthorizationList* or *Authorisation Cache*. After this step, the charging transaction begins where several packets are transmitted between the CS and the CSMS, containing information valuable in the anomaly detection process, such as the measured power values supplied to the EV, sent in the *startTransaction*, *MeterValues* and *stopTransaction* requests.

Currently, when it comes to using machine learning for anomaly detection, the most common learning paradigms are supervised, unsupervised and semi-supervised learning. In supervised learning, there is a label for each case on the dataset, generally to express if that case is an anomaly or a normal instance. In unsupervised learning, there are no labels present in a dataset, and in semi-supervised there is a subset of labelled data and a subset of unlabelled data, combining supervised with unsupervised learning. By analysing the literature, we can identify the most common techniques associated with anomaly detection, like Cluster Analysis Based Techniques, Tree-Based Techniques, Density-Based Techniques, Support Vector Machines and Neural Networks Related Techniques. Cluster Analysis Based Techniques work by the assumption that normal instances of data have similar values, being near each other, while abnormal instances of data stand isolated and with values different from the norm. An example application of this technique is the K-Means algorithm, which aims to group similar data in clusters by attributing each data point to the closest centroid and adjusting the centroid's position based on the mean values of all the points in the cluster. A Tree-Based Technique aims to isolate anomalies by constructing decision trees, splitting a subset of values to the left and right of the tree based on a set of rules. This technique uses the fact that anomalous values are easier to isolate in a tree in a small number of steps, are commonly located near the root and have no child nodes, as similar values tend to be children of each other because of their value's proximity. An algorithm that applies this principle is the Isolation Forest, which calculates several of these trees in several subsets of data, using a data point's position in the tree as one of the factors to calculate the anomaly score, a value that determines how likely a data point is an anomaly. Density-Based Techniques calculate the number of other data points present within a radius around each data point, usually in an iterative process by initially defining the radius value and the minimum number of neighbouring data points in that radius. DBSCAN is an algorithm which applies this technique, classifying each data point based on the number of neighbours and radius. A data point that contains the same or higher number of neighbours in its radius is considered a core point. If a data point has less than the defined number of neighbours, but a core point is in its radius, it is density-reachable, and if this point is reachable from a path originating from the initial core point, it is density-connected. A data point with none of these characteristics is an outlier and likely to be an anomalous data point. Support Vector Machine techniques employ the principles of the SVM algorithm, which aims to divide different data points using a separation limit between the numerous data points, called a hyperplane. In the two-dimensional space, the SVM algorithm calculates several hyperplanes choosing the optimal one. As there are only two dimensions, this hyperplane is a function that describes a line, which is the optimal divider, maximising the margin between the data points of two different classes. An example in literature commonly used for anomaly detection using this technique is One-Class SVMs. This algorithm employs a polynomial kernel, which enables the projection of data to a new dimensional space by adding a new dimension based on the similarity of each case. With this in an initially two-dimensional problem, separating the two types of data points, normal and abnormal, is now possible as the third dimension allows to separate data points from each other, allowing to draw a hyperplane that can separate the two types of data points. One of the most studied topics in literature stands under Neural Networks Related Techniques, with the most promising networks falling beneath the deep neural networks branch. We can define Deep Neural Networks as the implementation of a neural network with one or more hidden layers between the input and output layers. Using these networks involves varying architectures for several problem types, like image classification, processing and generation, which is one of the most popular uses presently. An implementation that uses this technique with applications in anomaly detection is AE. Generally, an AE is divided into two neural network blocks. One is called the encoder, which aims to identify and compress the input features into a diminutive representation, often called latent space. This latent space is the input of the second block called the decoder, which aims to reconstruct the encoder's input with the lowest error possible. Applying an AE for anomaly detection would first involve training it over normal data instances, which allows the model to reconstruct these cases with minimal error. When an anomalous sample of data is provided to the trained model, the reconstruction error is bound to increase over the observed reconstruction error in training, as the model has not yet learned how to reconstruct those samples. A limit can then be defined over the reconstruction error, flagging a data sample as anomalous when that limit is surpassed. Still inside the Neural Networks Related Techniques, in literature, AE are often referenced together with VAE. Although some studies suggest that for anomaly detection and clustering problems, VAE can overcome the performance of other techniques, VAE is a deep generative model mainly used in areas such as image generation or language processing to create new data from given inputs. As the objective is different, so is the architecture of the network, where the encoder outputs the mean and variance for the latent distributions. The decoder here does not get the latent space representation directly but a sample of the latent distributions that, when used to reconstruct the initial input, will generate a different result.

As this study aims to conceptualise several machine-learning models applied to OCPP data and determine which is most suitable for anomaly detection, we must use several evaluation metrics. Depending on the learning paradigm used, evaluation metrics vary. As in supervised learning, the actual labels are known, and we can calculate the model's accuracy value using the true and false positive and negative values. This metric represents the percentage of true results over the whole result set but using this metric alone to prove the model's performance is not enough, as it can be easily affected by an imbalanced dataset and an overfitting model which learns and outputs the exact labels seen in the dataset, having no real value when applied to new data. As accuracy can easily be misleading, we need to consider other metrics like F1-score, log-loss or AUC. The F1-score is the combination of two widespread metrics *precision*, which represents the number of the predicted positives that are truly positive and *recall*, which outputs the ratio of the true positives successfully identified. The F1-score value increases as precision and recall also increase, favouring models that capture the problem correctly. By analysing the model's f1-score for each class, we can identify how accurate the model performed in predicting those class values as a model that predicts the highest occurring class for most of the dataset cases will have an f1-score value for the lowest occurring class close to 0, as it did not correctly identify those cases. For supervised models that output a probability estimate or binary results, Log Loss is a classic metric to apply. This metric is also known as cross-entropy loss, outputting a probability value between 0 and 1 based on the distance between the probability estimate value and the real value. The further these are, the higher the loss value is, and the closest they are, the lower the loss value is. Regarding the AUC, this metric allows us to measure the performance of a model over most cases observed in the dataset based on the two-dimensional area measured under the ROC curve. The ROC curve corresponds to a graphical representation of a model's performance using the True Positive Rate (TPR) and False Positive Rate (FPR) values, through which we measure the AUC value, which is proportional to the performance of the model, meaning the higher the performance, the higher the AUC value is.

These metrics that apply to a supervised learning model can not be used in an unsupervised learning context as there are no labels to indicate, for example, to which cluster or class we should attribute a data point. As most unsupervised techniques rely on similarity or proximity-based calculations, which ultimately result in clusters, the available metrics for unsupervised learning aim to evaluate the accuracy of the clustering process. Although the application of unsupervised learning techniques and algorithms usually means that there are no real labels for the data, one of the best ways to assess the performance of an unsupervised algorithm is by using the RI. The RI measures the similarity between the ground truth labels and the assignments predicted by the model, calculating a value between 0 and 1, representing the accuracy of the attributions of the model. Although through the value of the RI, we can estimate the performance of the model, having a value close to 0 does not mean that the model is randomly assigning data points to clusters, but by using the ARI, we can have more guarantees over this aspect. The output value of the ARI varies from -1 to 1, where a value close to 1 means a more accurate clustering process, and one closer to -1 means an increased number of random and incorrect assignments. Concerning semi-supervised learning, as this paradigm uses a combination of supervised learning and unsupervised learning, evaluating a semi-supervised model can involve employing supervised or unsupervised learning evaluation metrics due to the fact that the most common use for this paradigm is the optimisation of a supervised model classifying the unsupervised part of the dataset using a model trained over the supervised part of the set. Here the final model is a supervised model, so the appropriate evaluation techniques are those usually applied to this paradigm. However, when the objective is the optimisation of an unsupervised model, where the supervised portion of the dataset can be used to help the cluster identification, for example, the result is nonetheless an unsupervised model where we can apply unsupervised learning evaluation metrics.

As this study aims to use OCPP data to conceptualise machine learning models, we analysed the literature to scan for similar work in this field. Upon this analysis, we can identify several types of scientific work developed using OCPP data. Most studies examined fall under the cybersecurity field, where some use machine learning techniques to conceptualise models capable of detecting a specific cyberattack and others aim to study the charging network infrastructure and the OCPP protocol to look for potential security vulnerabilities and identify possible solutions, some with the use of machine learning. Among the studies analysed, we can mention the authors in Y. Liu et al. (2014) [31], which used support vector machines to monitor pricing curves and the energy load looking for anomalous peaks, which can indicate a pricing cyberattack on an OCPP charging network. The same authors in Y. Liu et al. (2016) [32] later proposed an observable model based on the Markov decision process to allow the explanation of results obtained using the model and the detection process. The main reason behind the motivation for research in this field, according to authors such as Garofalaki et al. (2022) [3], is related to the increase in cyberattacks directed at the charging network infrastructure, which is only likely to increase as electric vehicles become more commonly used. Apart from this, although new OCPP versions come with new and upgraded security features, there are nevertheless many possible ways an attacker can compromise a network, but also several solutions for those problems, some proposed by researchers working in this field. The authors used this knowledge, creating a survey approaching the release of OCPP 2.0, describing the new security features while underlining the main flaws of the protocol that can allow, for example, man-in-the-middle attacks to happen even in this version of the protocol. The authors give an example regarding the communication between the CS and the EV, where an attacker can position themselves between the two, usually to pose as one of the conversation's participants, claiming that using the TLS protocol is not enough to prevent this attack. In the article, the authors propose a solution, referencing the research of authors in Kabir et al. (2021) [33], where a Back Propagation Neural Network (BPNN) could be trained over OCPP data, analysing charging and discharging requests during a charging transaction looking for anomalous changes in the values indicating a possible manipulation of these values, which can originate in a man-in-the-middle attack.

Chapter 3

Experimental Setup

In this chapter, we initially focus our attention on the methodology chosen for this study, explaining the phases of the project and the technologies used and presenting details about the machine where we performed the tests. After this, in section 3.2, we describe the proposed approach to solving the problem in question, approaching some of the reasons that led to the charging transaction data not being used in the conception of machine learning models. In sections 3.3 and 3.4, we describe the data analysis done on both datasets, together with the data preprocessing for the connector status dataset, respectively. Lastly in section 3.5, we summarise the main aspects referenced in all section of this chapter.

3.1 Methodology

In order to accomplish the objectives set for this study, this section shows the proposed approach taken regarding the problem at hand. The first step to developing the machine learning project is data collection and acquisition, already done as this study was conducted together with We Can Charge, which actively gathers data from several of their charging stations. The data received was organised into two datasets, one regarding several connectors, including their status and error details, and the other regarding charging transactions mainly with the transferred energy values. We approach the data analysis of both datasets in more detail in section 3.3. After the initial dataset and business case analysis, we initiate the conception of the machine learning models. First, we start with the pre-processing phase, resolving any discrepancies detected in the data or dataset structure. Secondly, we move to the algorithm application phase, where the already pre-processed data is provided to the chosen algorithms to train and test them.

Concerning the project management strategy, this study will use the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology [34], which defines six phases to guide the model development process. In the first phase, called *Business Understanding*, we try to understand the problem and define the objectives and requirements of the project by conducting research and communicating with the company, We Can Charge. After this, we move to the second step, corresponding to the *Data Understanding* phase, where the collected data is explored and described, before moving to the *Data Preparation* phase. In this phase, we aim to treat the data, decide which attributes will be used and calculate others, if possible, from the ones in the dataset. In the *Modeling* phase, the goal is to conceive the machine learning models using the chosen algorithms, which are tested in the Evaluation phase, using evaluation metrics and interpreting the results using the domain knowledge available. If a model performs within the required parameters, we move to the *Deployment* phase, where we plan and execute a deployment of the model using it in production [34]. All these phases that form the CRISP-DM Lifecycle and their execution order are illustrated in figure 13, located below.

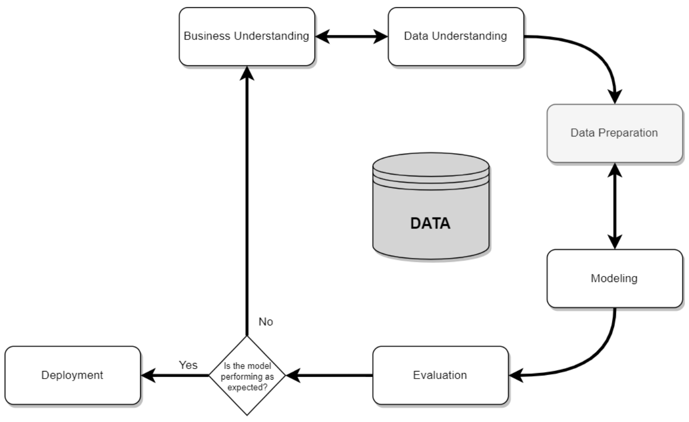


Figure 13: CRISP-DM Methodology Lifecycle. Adapted from [34]

In the phases described previously, we utilised several technologies. Some of these are mainly associated with the data understanding and preprocessing phases of the lifecycle, such as Seaborn [35] and Matplotlib [36] used to visualise the data and obtain graphical representations and Numpy [37] and Pandas [38] for data and file manipulations. We developed the project using the Python programming language, which allowed us to use three frameworks for the machine learning algorithm implementation. Scikit-learn [39] provided us with several implementations of known algorithms in the literature, evaluation metrics, and other helpful features for the preprocessing phases, such as normalisation and standardisation implementations. Tensorflow [40] and Keras [41] were mainly used to implement the deep neural network models.

All of the models and tests performed were run locally on a computer with an Intel Core i7-9750H with a base clock of 2.60 GHz and a maximum turbo boost frequency of 4.50 GHz, 16 GB of DDR4 RAM, a 512GB SSD drive and a GTX 1650 graphics card. The computer was running version 22H2 of Windows 11 with all patches installed.

3.2 Experiments

Regarding the experiments conducted during the development of this dissertation, as we have two datasets containing different types of data, we can divide our experiments into two stages.

Regarding the first stage, related to the connector status data, we attempted to leverage the simple structure of the dataset combined with the OCPP protocol specification to classify the data, allowing us to apply a semi-supervised and fully supervised approach. For the semi-supervised approach we experimented conceiving models with several algorithms, each implementing a different anomaly detection technique, using the known labels to tune the model's performance and therefore improve the results [5]. Here the benchmark of the results was performed using supervised learning metrics such as precision, recall, f1-score and accuracy, with a manual but also automatic hyperparameter tuning using the grid search cross-validation method. Considering that the real labels are known, we also attempted to conceive an *MLP* classifier using the same evaluation metrics mentioned previously. Although the best model of this first phase will be able to classify, in real-time, whether a connector is reporting an anomaly, we also aimed to conceive a model that could enable us to predict if anomalies will happen in the future. This corresponds to the second phase of our experiments, where we aim to conceive the Connectors Forecasting Network (CFN). The structure of this network is shown in figure 14.

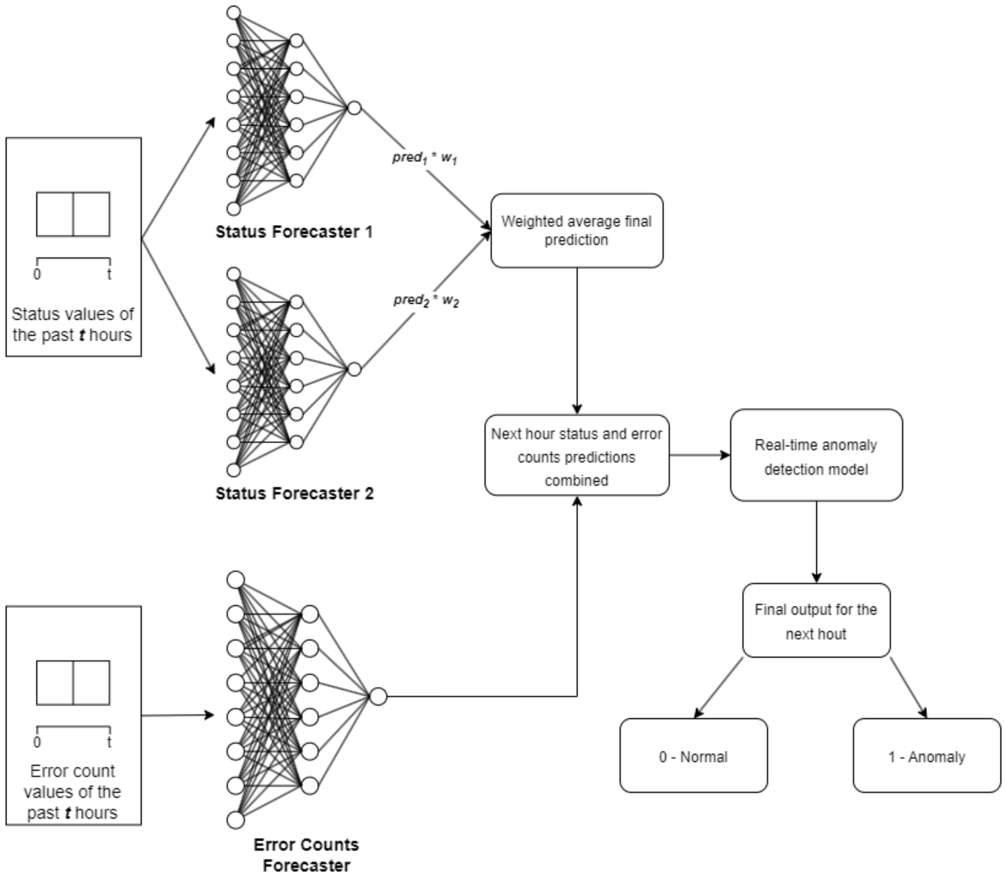

Status values of the past *t* hours
0
t
Status Forecaster 1
Status Forecaster 2
$pred_1 * w_1$
$pred_2 * w_2$
Weighted average final prediction
Next hour status and error counts predictions combined
Real-time anomaly detection model
Final output for the next hout
0 - Normal
1 - Anomaly
Error count values of the past *t* hours
Error Counts Forecaster

Figure 14: CFN proposed architecture

This network can be divided into three modules. The first module corresponds to the status prediction module where two LSTM status forecasting networks are used. Here we aim to ensemble these two models, calculating a final value for the next hour based on a weighted average of each model's predictions ($pred_1$ and $pred_2$) multiplied by each model's weight parameter (w_1 and w_2). This is a process similar to what has been done before in other studies, such as authors in Tan et al. (2020) [42], where a complex model ensemble between several LSTM models was used for short-term forecasting in reduced sliding window datasets. In the ensemble performed, we experimented several hyperparameter combinations to obtain the first status forecaster, denominated as Optimised Status Forecaster (OSF), and used the same parameters combined with a weighted Mean Squared Error (MSE) loss function, creating the Weighted Optimised Status Forecaster (WOSF), in an attempt to improve the OSF model's capacity in predicting the anomalous states (*Faulted* and *Unavailable*). This function is calculated as shown in equation 11 below, where N is the total number of elements in the y_{true} and y_{pred} sets and w a penalisation value usually between zero and one.

$$\frac{1}{N}\sum_{i=1}^{N}(y_{\text{true}}^{(i)} - y_{\text{pred}}^{(i)})^2 \cdot (1 + w \cdot y_{\text{true}}^{(i)}) \tag{11}$$

Regarding the second module, this corresponds to the LSTM error counts forecaster, which outputs the error counts for the next hour. This model was obtained performing manual hyperparameter optimisation through the analysis of the evaluation metrics and learning curves. The output value of this model is combined with the final status prediction and used as the input of the last module. The latter corresponds to our real-time detection model, which outputs the anomaly classification value for the next hour. As we can see in figure 14, both the status and error count modules receive as input the respective values of the past t hours. This strategy allows us to predict using historical data in a given time window what the status and error count values of the next hour are going to be, anticipating the occurrence of an error and therefore improving the existing maintenance system, with the data the OCPP protocol provides.

For the charging transaction data, in the data analysis step, we encountered issues applying this dataset in the forecasting and real-time detection aspects. These are related to the lack of information provided by the OCPP protocol which would be necessary to assert the developed model's accuracy when detecting or forecasting anomalous behaviour during the charging transaction. As the protocol only sends the instant or accumulated energy values, with optional instantaneous amperage and current values, there are other variables to consider that directly influence values, such as the type of charger or charging method of a given station, the maximum allowed power draw allowed by the vehicle and the percentage of the battery at the beginning of the charging process are only some examples of additional information which is necessary to validate a model's performance over the instantaneous power curves, as we describe in more detail in section 3.3.2.

3.3 Data Analysis

As stated before, We Can Charge supplied us with two different datasets, one regarding connector status packets and the other regarding charging transactions registered in several charging stations.

The first dataset consists of *StatusNotification* packages, which are sent between the charging station and the CSMS, informing the central system of the status of the connectors periodically. The second dataset includes data regarding several charging transactions, more particularly consisting of *StartTransaction*, *SampledValues* and *EndTransaction* packets, from which we can extract the several measured values in the start, middle and end of a given transaction.

3.3.1 Connector Status Dataset

The connector status dataset contains 12635 rows of data, each one corresponding to a *StatusNotification* message as defined by the OCPP documentation. Each message relates to a specific connector of a given charging point. When the connector is not currently in a charging transaction, it only reports to the CSMS periodically, indicating its availability through the *Available* status. If this status suffers any changes due to an error or because a charging transaction started, a new *StatusNotification* message is sent. Table 1, located below, describes this dataset's structure, showing for each column a description and the data type.

Table 1: Connector Status dataset structure

Column Name	**Column Description**	**Data Type**
connector_pk	Unique identifier for the connector.	Integer
status_timestamp	Timestamp with time and day when the status of the connector was read.	Timestamp
status	The status of the connector, which can have nine different values. The status values of *Available*, *Preparing*, *Charging*, *SuspendedEV*, *SuspendedEVSE*, *Finishing* and *Reserved* are considered normal values. The *Unavailable* or *Faulted*, states are considered error values as the connector is not operational when these status values are present.	String
error_code	Contains an error code if the connector is exhibiting any errors, otherwise the value is *NoError*.	String
error_info	If an error code exists, this column contains more information about that error code.	String
vendor_id	The identifier for the specific vendor of the connector.	String
vendor_error_code	If an error is detected and it is vendor-specific, this column contains its error code.	String

After verifying the dataset's structure, we applied basic mathematical measurements, analysing values such as the mean, mode, maximum and minimum values for all dataset columns. Through the mode, we verified that the most common occurrence in the dataset is related to packages which demonstrate no errors at all and contain a status value of *Available*. Additionally, and through the maximum and minimum values for the *status_timestamp* column, we were able to confirm the data collection period ranges from July 1, 2022, at 01:51h until December 30, 2022, at 20:33h, therefore spanning over five months with an irregular sampling rate as it contains missing days.

Following this analysis and using the *status_timestamp* once again, we can visualise the number of packets transmitted between the CS and the CSMS per day, plotting the temporal window for when the data collection happened, as we can see in figure 15.

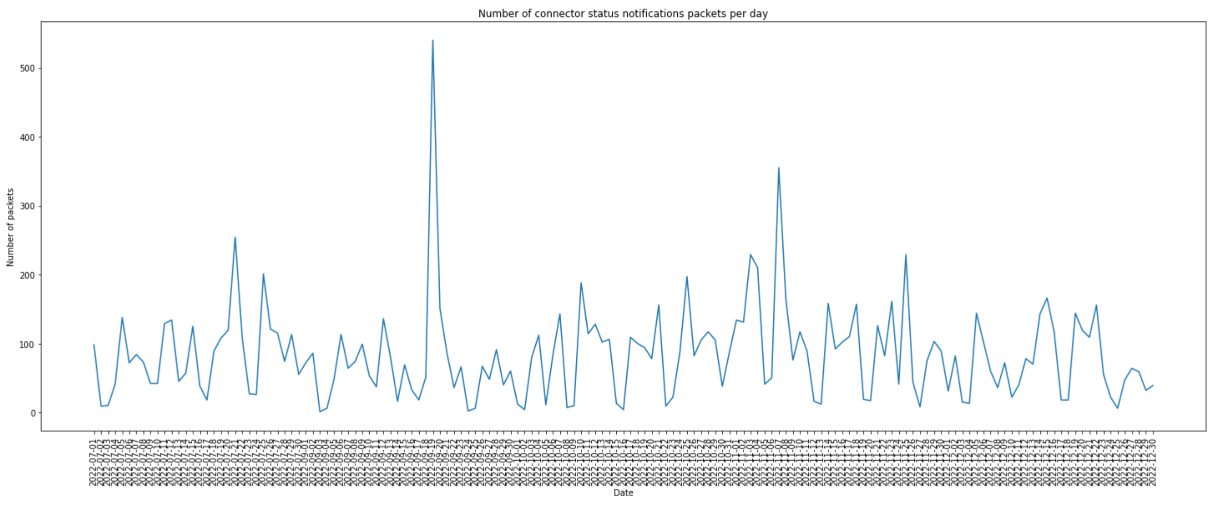


Figure 15: Number of status notification packets sent per day

From the figure above, we can see that the activity of the connectors varies between the several days, which can be due to the affluence of users and charging transactions that it registers on that day, as the station has to inform the CSMS of the status before, during and after a charging transaction. In the figure, we can see that some peaks recurrently happen and that the station had a peak utilisation on September 9, 2022, as the maximum number of status requests was registered that day.

When analysing the total number of packets sent for each hour, as presented in figure 16, it is possible to verify the connector's activity where during the night, as fewer users use the station, the connectors only send periodic status information to the CSMS. In contrast and during the day, the number rises as more users start utilising the charging station and, therefore, the connectors, reaching peak utilisation at approximately 9 o'clock.

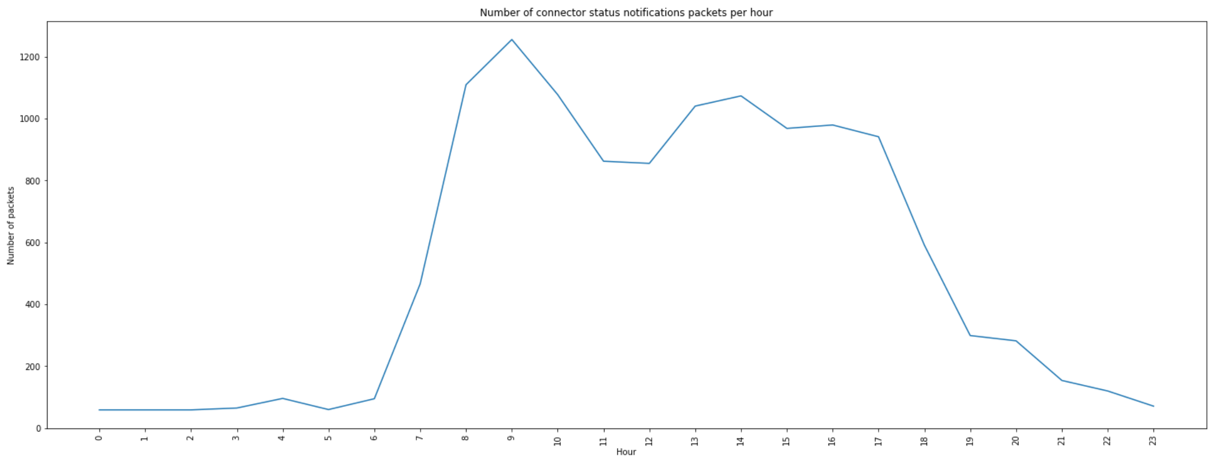


Figure 16: Total number of status notification packets sent for each hour

After this and considering the values of the *status* column, we can verify the distribution of normal and abnormal status values, as figure 17 shows.

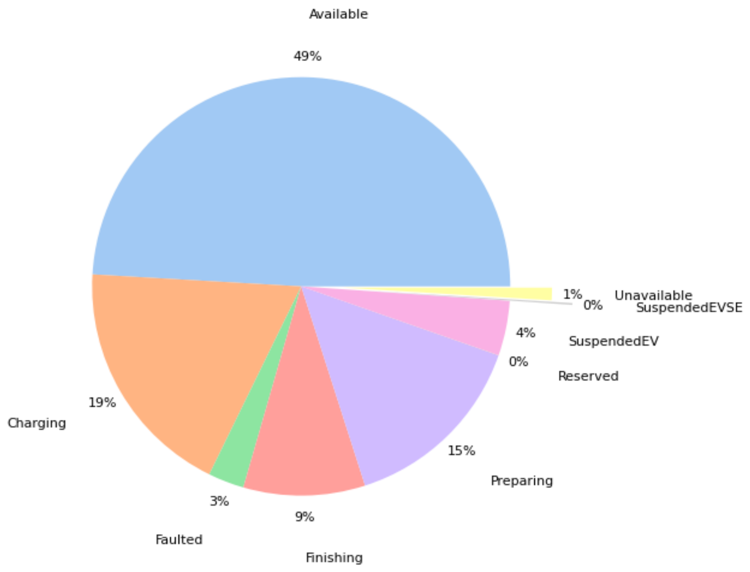


Figure 17: Distribution of status values in the dataset

In the figure above, we can see that 4% of cases include an error state, where 2% registered a *Faulted* and 1% an *Unavailable* state, rendering the connector unusable while these states are present. Analysing the *error_code* column and grouping these errors by the state of the connector, we can see the distribution of the error types and identify a relationship between these and the values in the state column, as expected. The result of this analysis can be verified in figures 18 and 19, located at the top of next page. Through this analysis, we were able to detect that most error codes are related to connectors on a *Faulted* state, with some exceptions. All of the *OverVoltage* errors registered are associated with the *Charging*, *Finishing*, *SuspendedEV* and *Available* states. Out of the 295 *OtherError* values, 156 are associated with the *Available* state as well as with all of the *LocalListConflict* errors.

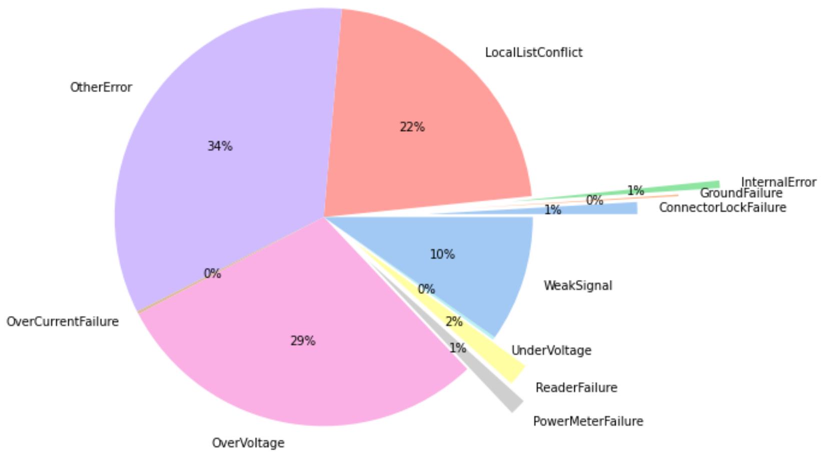


Figure 18: Types of errors present in the dataset

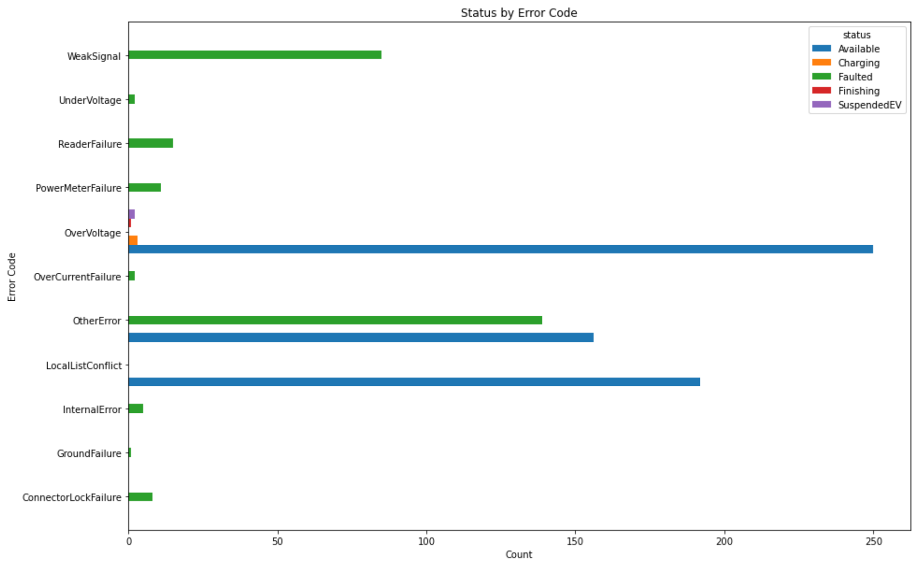
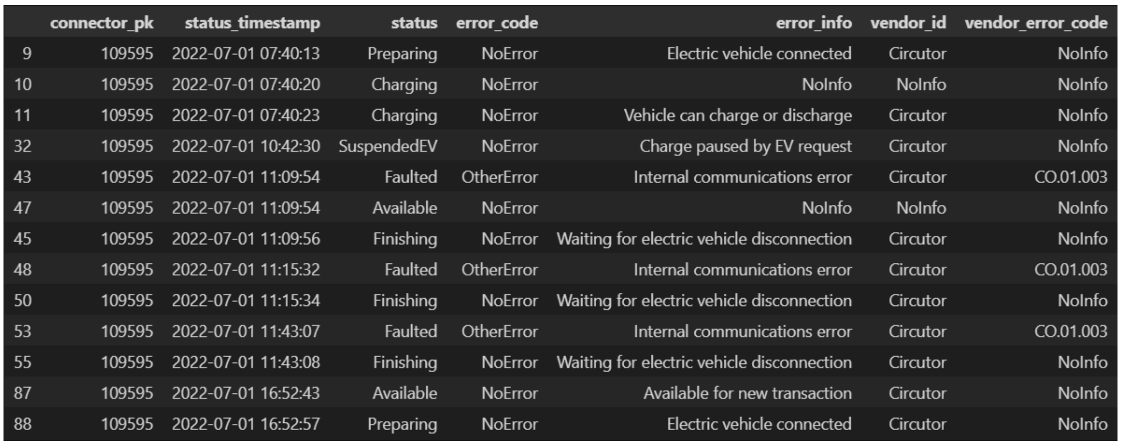


Figure 19: Types of errors related with the state value registered

To detect how these errors can occur during the normal functioning of a connector, we analysed the connectors in the dataset to capture anomalous behaviour manually, which was possible, as figure 20 shows for connector number 109595, at the top of the next page. Following the timeline, we see that at 07:40:13h, the connector starts a charging transaction as the state value is *Preparing*, and the following rows indicate a *Charging* state. At 10:42:30h, the EV suspends the charging process as it has reached full charge, but at 11:09:54h, the connector reports a *Faulted* status, immediately reporting to be available. Following this behaviour, the connector switches between the *Finishing* and *Faulted* status repeatedly until, at 16:52:43h, it returns to *Available*. Subsequently, a new charging transaction starts at 16:52:57h, with the same connector where no errors were detected. As there is no more information regarding the vendor error code, and after consulting We Can Charge, we can state that the error is related to that specific vendor implementation, as others report normal behaviour without this error and verified this to be a recurrent problem with some OCPP implementations of this vendor.



	connector_pk	status_timestamp	status	error_code	error_info	vendor_id	vendor_error_code
9	109595	2022-07-01 07:40:13	Preparing	NoError	Electric vehicle connected	Circutor	NoInfo
10	109595	2022-07-01 07:40:20	Charging	NoError	NoInfo	NoInfo	NoInfo
11	109595	2022-07-01 07:40:23	Charging	NoError	Vehicle can charge or discharge	Circutor	NoInfo
32	109595	2022-07-01 10:42:30	SuspendedEV	NoError	Charge paused by EV request	Circutor	NoInfo
43	109595	2022-07-01 11:09:54	Faulted	OtherError	Internal communications error	Circutor	CO.01.003
47	109595	2022-07-01 11:09:54	Available	NoError	NoInfo	NoInfo	NoInfo
45	109595	2022-07-01 11:09:56	Finishing	NoError	Waiting for electric vehicle disconnection	Circutor	NoInfo
48	109595	2022-07-01 11:15:32	Faulted	OtherError	Internal communications error	Circutor	CO.01.003
50	109595	2022-07-01 11:15:34	Finishing	NoError	Waiting for electric vehicle disconnection	Circutor	NoInfo
53	109595	2022-07-01 11:43:07	Faulted	OtherError	Internal communications error	Circutor	CO.01.003
55	109595	2022-07-01 11:43:08	Finishing	NoError	Waiting for electric vehicle disconnection	Circutor	NoInfo
87	109595	2022-07-01 16:52:43	Available	NoError	Available for new transaction	Circutor	NoInfo
88	109595	2022-07-01 16:52:57	Preparing	NoError	Electric vehicle connected	Circutor	NoInfo

Figure 20: Anomalous behaviour of a connector in the dataset

3.3.2 Charging Transactions Dataset

The charging transactions dataset, as stated before, consists of three types of requests sent between the charging station and the CSMS, *StartTransaction*, *StopTransaction* and *SampledValue*. From the *StartTransaction* request, we can extract information like the start date for the transaction and the measured meter value in Wh. We can derive the same information from the *StopTransaction* request but related to the end of the transaction, with additional information in the *Reason* field regarding the cause for stopping the transaction, during which the charging stations can, for example, measure the Wh values transferred to the vehicle by sending *SampledValue* requests to the *CSMS*.

The structure and detailed descriptions of each field gathered from these packets can be observed in table 2. The dataset contains a total of 19694 rows from 69 different transactions.

Table 2: Charging Transactions dataset structure

Column Name	Column Description	Data Type
Value Timestamp	Timestamp for when the value was measured.	Timestamp
Value	The measured value on a given timestamp for a certain transaction	Float64
Reading Context	Indicates the context in which the value was read, with values such as *Sample.Periodic* for a measurement done while the transaction is occurring, *Transaction.Begin* and *Transaction.End* for the start and end of a transaction, respectively.	String

Continued on next page

Table 2: Charging Transactions dataset structure

Column Name	**Column Description**	**Data Type**
Format	Optional field to indicate how the data should be interpreted. If the value is omitted then *Raw* is assumed, meaning the data should be interpreted as integer/decimal numerical data. If the value is *Signed-Data* the data is represented as a signed binary data block, encoded as hex data.	String
Measurand	Determines the type of measurement and has a default value of *Energy.Active.Import.Register*, meaning the value measured is the energy imported by EV in Wh or kWh. In the dataset, it also includes other values such as: *Current.Import*, *Current.Offered*, *Power.Active.Import*, *Power.Offered*, *Power.Reactive.Import*, *Voltage* and *SoC*.	String
Location	Indicates where the measurement took place. Can assume values such as *Outlet*, default value, which specifies that the measurement was performed at a Connector, and *EV* when the measurement was taken by *EV*.	String
Unit	Unit of the value measured, with the default value of Wh. In the dataset other units are also used such as: W, A, V and *Percent*.	String
Phase	Optional field which specifies how the measured value should be interpreted. When omitted, the measurement is an overhaul value otherwise a phase must be supplied. There can be three phases, *L1*, *L2* and *L3*.	String
Transaction ID	The unique identifier for a transaction.	Integer
ChargeBox ID	Unique identifier of the charge box which is the equipment that contains the connector.	String
Connector ID	Unique identifier of the connector being used in the transaction.	Integer
OCPP ID Tag	Corresponds to a unique identifier for a user that interacts with a charging station.	String
Start Date/Time	Start date and time for the transaction.	String
Stop Date/Time	Stop date and time for the transaction.	String
Start Value	The meter value read at the start of the transaction in Wh.	Integer
Stop Value	Meter value read at the end of the transaction in Wh.	Integer

Continued on next page

Table 2: Charging Transactions dataset structure

Column Name	**Column Description**	**Data Type**
Stop Reason	Reason for stopping a transaction. Can include normal termination values such as *Local* and *Remote*, but also error termination values, for example: *EVDisconnected*, *Other*, *PowerLoss* and *DeAuthorized*	String
Stop Event Actor	The actor which terminated the transaction. This can assume values such as: *manual* or *station*.	String

In this data analysis step, and performing a count of the values present in the dataset, we confirmed the presence of missing values in the *Reading Context*, *Format*, *Measurand*, *Location*, *Unit* and *Phase* columns. These were expected, considering that these are optional in a *SampledValue* request. Additionally, by verifying the maximum and minimum values of the dataset, we can see that some transactions include a measured meter value at the start of the transaction larger than 0, as some stations accumulate this value from one transaction to another. We can also verify that the dates range from October 10, 2021, to March 7, 2023, with a non-continuous data collection process, as there are missing dates.

Grouping the packets by their timestamps, we can see that the data collection process was non-continuous as some moths contain more transactions than others, as presented in figure 21, located below. Plotting the number of packets per hour, we can see a similar evolution as observed for the connector dataset in figure 22, located at the top of the next page.

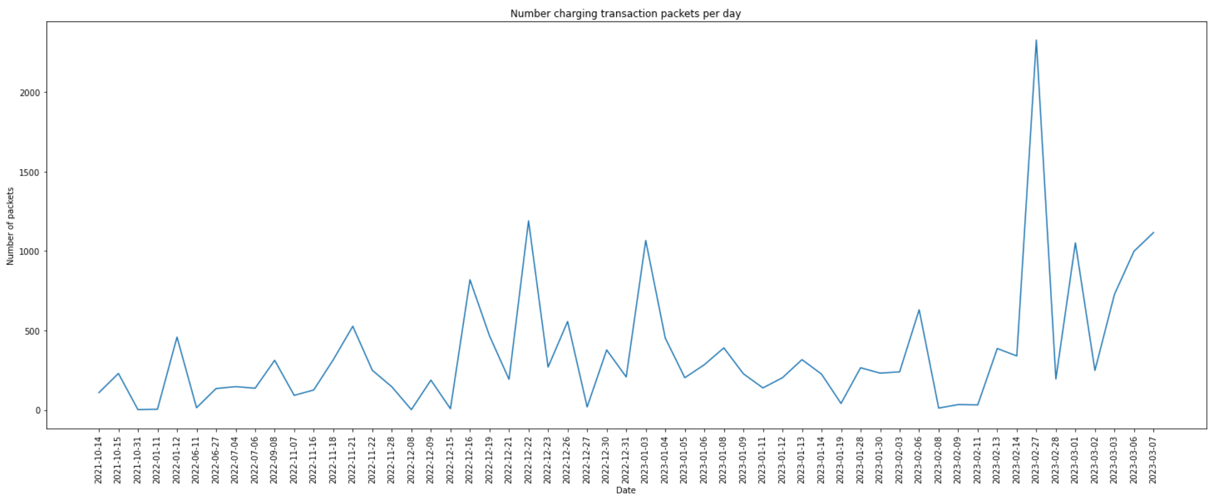


Figure 21: Number of packets registered per day

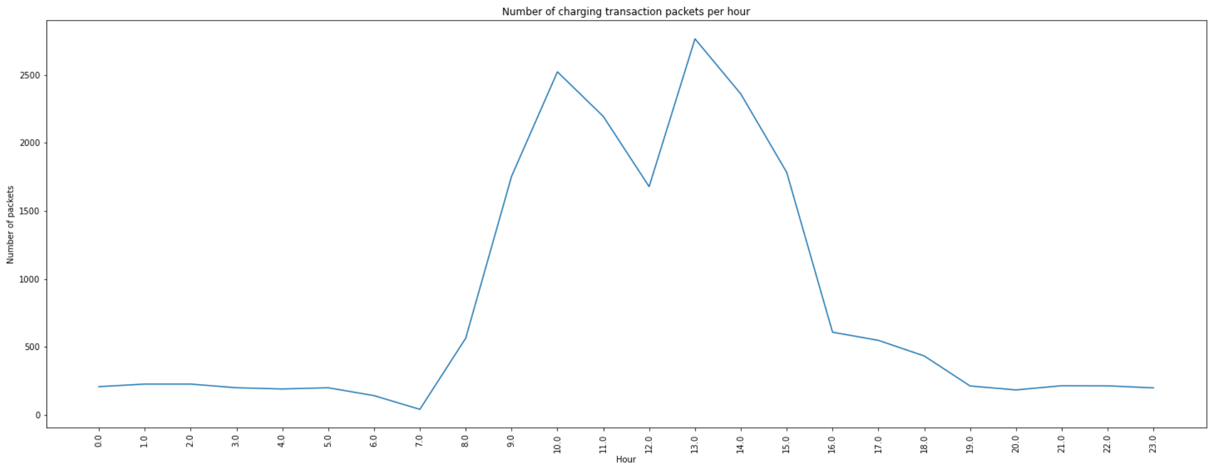


Figure 22: Number of packets registered per hour

As we aim to perform anomaly detection for the transactions, one of the most important columns to analyse is the *Stop Reason* column, which, as described previously in table 2, contains the reason for stopping a transaction. From the values present in this column, we were able to verify that 53% of the transactions terminated normally, with 42% reporting a *Local* termination, 9% a *Remote* and 2% with a *NoInfo* termination value. The *NoInfo*, is considered a normal termination as this is a required field in version 1.6 of the OCPP protocol, which through the documentation states that the reason can only be omitted if the transaction terminated with a *Local* state. The remaining 47% of the transactions report an anomalous termination state. The distribution of these values in the dataset can be verified in figure 23, located below.

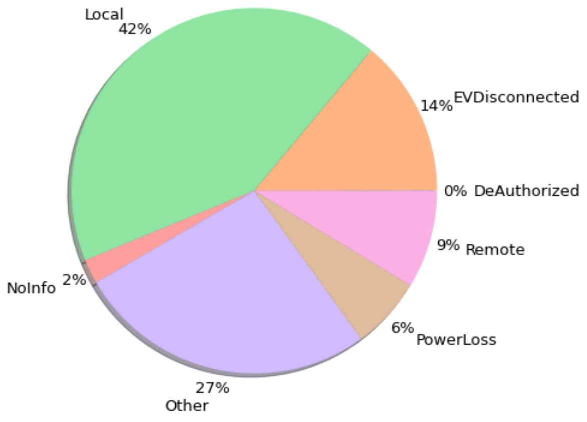


Figure 23: Stop Transaction values distribution

Regarding the transaction's values, analysing the *Measurand*, *Unit* and *Value* columns, we can see that for the majority of the transactions, the values were measured in Wh, as 34% of the values contain the *Wh* string in the *Unit* column. To these, we add 24% of the data, which, although did not have any value in the *Unit* column, as this it was omitted, by having an *Energy* type in the *Measurand* column, we conclude that these values were also measured in Wh, as stated in the OCPP documentation. The second largest group of values was measured in W with 28%, followed by voltage and amperage data, both with 6% and finally 2% with the percentage of the vehicle's battery during the transaction. Furthermore, by selecting a specific transaction, we explored the power, amperage and voltage curves, plotting each value measured using the *SampledValue* timestamp data. From these, we can identify nominal patterns in the data for these curves, information which can be advantageous, for example, in the data preprocessing phase.

Regarding the measured values in Wh, these gradually increase over time since the start of the charging transaction as the vehicle's battery starts receiving the energy transferred through the connector. Although through these curves we can identify the effects of a power loss in that station, considering that a drop in the wattage means the accumulated Wh values will increase at a lower rate than before, these curves are predominantly similar, as we can see in figures 24 and 25.

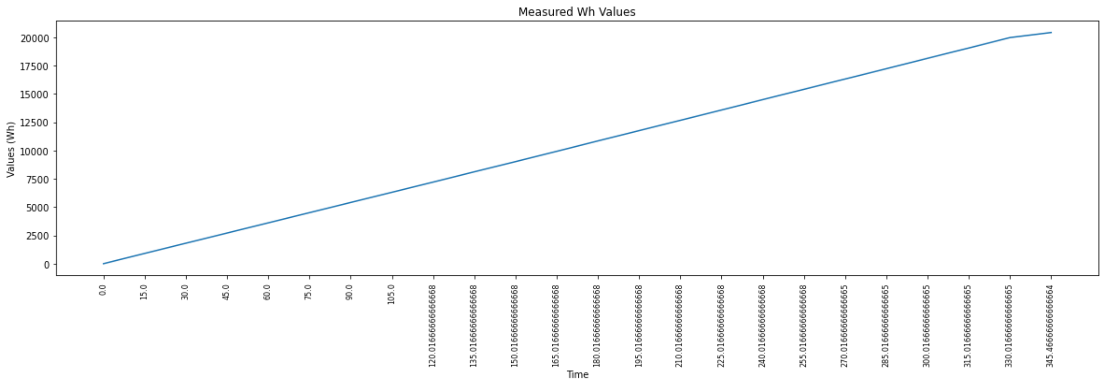


Figure 24: Normal transaction Wh curve

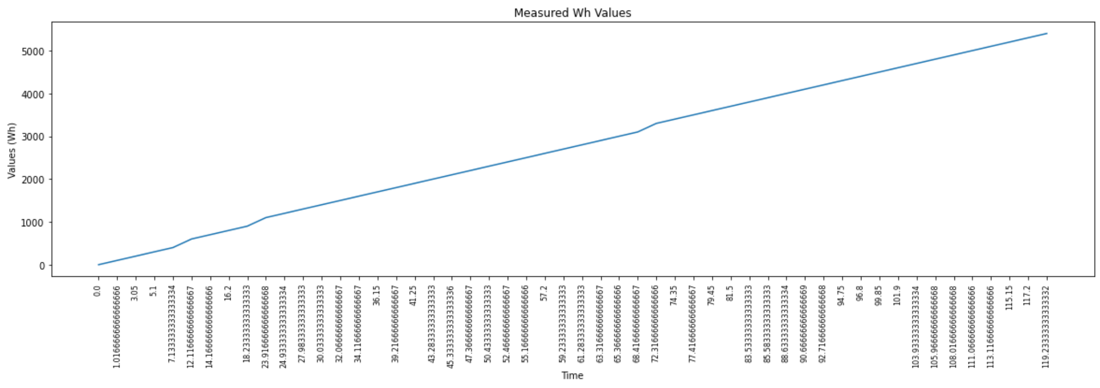


Figure 25: Transaction Wh curve where a power loss was detected

Now focusing on the wattage values, these follow a pattern directly related to the measured current and voltage, as: $Wattage = Amperage * Voltage$. This relationship is related to the charging method applied, which in this case, follows the pattern of the most widespread technique. This technique results from combining two charging methods Constant Current (CC) and Constant Voltage (CV). With this method, the current initially rises and reaches a stable value, defining the first phase of the charging process, which corresponds to the CC method, during which the W values gradually increase until they reach the maximum energy transfer peak, stabilising after this. Following this phase, and as the vehicle charges, the voltage increases until the maximum value allowed by the battery, retaining a constant value as the amperage starts to drop and the battery nears its full capacity, which results in the wattage assuming constant values. This phase results from applying the CV method. Finally, as the amperage reduces further, the power in W starts to decrease until it reaches a value of 0 when the battery is unable to hold any more energy [43].

In figure 26, we can see the watts, amperage and voltage curves for one of the transactions in the dataset.

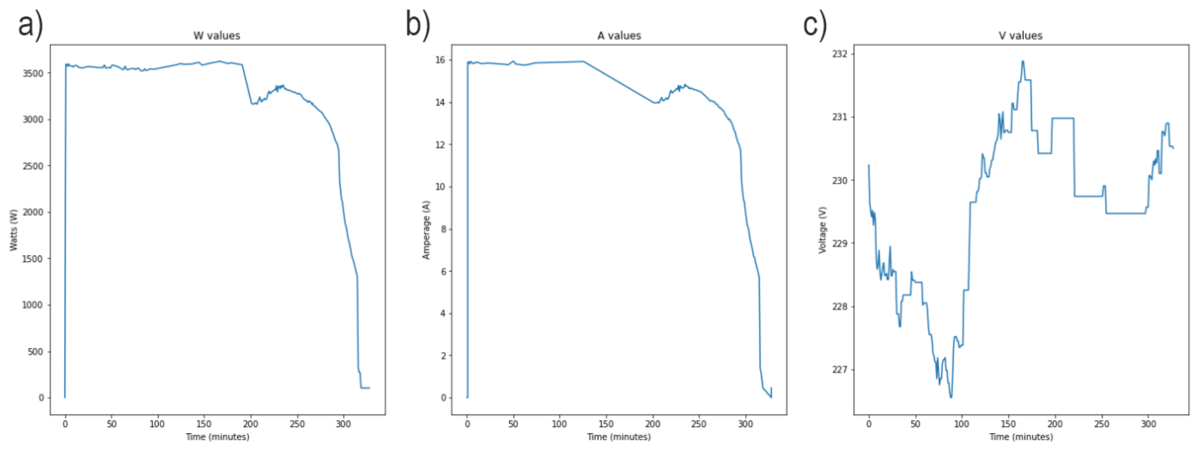


Figure 26: Wattage *a)*, amperage *b)* and voltage *c)* curves for a transaction in the dataset

In these curves, specifically in the wattage and amperage curves, anomalies like a power loss during the charging process are more noticeable, as the measured values correspond to instantaneous power and amperage levels. As seen in figure 27, located at the top of the next page, the wattage frequently drops during the transaction. For example, in the first most extensive spike, located close to 200 minutes, we verify a 78,62 W decrease in 9 minutes, with this behaviour repeating throughout the transaction, which is not the expected behaviour especially considering the length of this transaction.

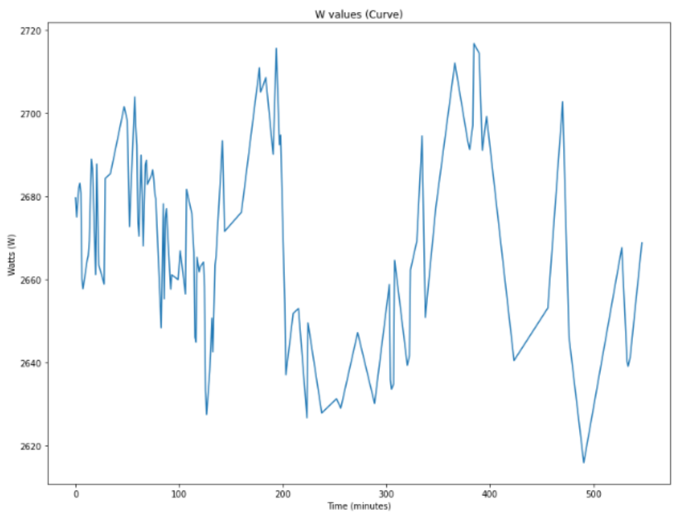


Figure 27: Anomalous wattage curve

Although through this data analysis, we can conclude that we can detect anomalous patterns through the instantaneous power curves, this data won't be usable as the wattage values rely on several variables for which the dataset does not account. These can be the vehicle's total battery capacity, maximum input power capacity, the charging station's maximum power transfer and the battery percentage at the start of the charging process. All of these variables have a significant impact on the scale of the measured values, as well as the time of the transaction in minutes, as if the vehicle is close to full charge, both the Wh and W curves will posses lower values than a vehicle with an empty battery as this one can accept more power, which consequently will also lead to a more prolonged transaction. Additionally, by manually analysing all the transactions in this dataset, we can verify that most of them follow this anomalous behaviour, some more than others, leaving only a reduced number of nominal transactions, which also prevents the use of some machine learning techniques, such as autoencoders, as we don't have enough data to train the network to recognise and reconstruct these samples.

Considering these observations, the transactional data obtained from the OCPP protocol does not provide enough information on their own to confidently conceive a machine learning model with accurate results when deployed in a real scenario, as the values can vary, being higher or lower depending on the station, vehicle and even OCPP implementation. Moreover, the lack of nominal transaction data prevents the application of techniques like autoencoders which could be helpful in this scenario as they have shown promising results when applied to similar problems. For instance, an example of this is the study conducted by Pereira and Silveira (2018) [44], where a proposed model using a variational recurrent autoencoder for anomaly detection over solar energy data was successfully applied.

3.4 Data Preprocessing

After the data analysis step, we performed the necessary prepossessing by applying several techniques. Considering that we aim to identify connectors that exhibit anomalous behaviour, the first step was the removal of the *connector_pk* and *vendor_id* columns, as these do not possess any meaningful information for the anomaly detection process. Analysing the categorical values in the *status* column, we can verify that these change during a transaction exhibiting a specific order. Considering this, we used ordinal encoding, transforming the categorical values to a numerical representation which denotes the order by which these should appear on the dataset. Furthermore, after verifying the number of cases per each state, we confirmed that there were only two rows with the *Reserved* state in the dataset, which led to their removal, as no technique could be applied to justify maintaining these values. The correspondence between the new numerical values, with each state's description, can be seen in table 3.

Table 3: Status column numerical encoding

Categorical Value	**Value Description**	**Numeric Value**
Available	Connector available for a new user.	1
Preparing	Connector is occupied after a user presented a tag, connects a cable to the vehicle or occupies the parking bay.	2
Charging	Contactor of the connector closes and charging process of the vehicle begins.	3
SuspendedEV and SuspendedEVSE	Contactor is open due to the EV not being ready or by request of the EVSE.	4
Finishing	Transaction has finished but the connector is not yet available for a new user as the vehicle is still in the parking bay or the cable has not been removed for example.	5
Faulted	An error was detected in the connector and it is not available to charge a vehicle.	6
Unavailable	The connector is not working due to a Change Availability command or entered this state by definition of the Charge Point.	7

As for the *error_code*, *error_info* and *vendor_error_code*, and considering that these are also categorical, we performed one-hot encoding obtaining a new column for each value in these columns. As some error codes have specific error information and usually an equivalent vendor error code, we grouped the columns corresponding to the same error type. This process allowed us to remove the features where the values would be directly related, leaving only the essential ones that indicate the presence of each error. This preprocessing technique, which we named Encoded and Grouped (EG), was among the first techniques applied to the dataset.

In table 4, we can verify the resulting features from employing this technique accompanied by a description. The remaining preprocessing techniques are approached later in this section.

Table 4: *Error_code*, *error_info* and *vendor_error_code* one-hot column grouping

Column Name	**Column Description**
ErrInfo_ConnectorBootAndCurrent	Indicates if there is an error related with the connector's boot or with the supplied current.
ErrInfo_EV_Disconnected	Indicates errors related to the disconnection of the EV.
ErrCode_ConnectorLockFailure	Errors related with the locking or unlocking of the connector necessary to allow power transfer during a transaction.
ErrCode_OtherError	Any type of vendor specific error, like an internal communication error.
ErrCode_PowerMeterFailure	Failure to read the power meter.
ErrCode_InternalError	Internal connector error weather it is in the software or hardware.
ErrCode_UnderVoltage	Voltage provided under the minimum required for the connector.
ErrCode_OverCurrentOrVoltageFailure	Voltage or current surpassed the limit specified for the connector.
ErrCode_GroundFailure	Ground fault circuit interrupter has been activated.
ErrCode_WeakSignal	Signal of the wireless communication is weak.
ErrCode_ReaderFailure	Failure with *idTag* reader.
ErrCode_LocalListConflict	Conflict between the authentication information from the CSMS when compared to the *LocalAuthorizationList*
ErrInfo_NotInitialized	Connector is not initialised.
ErrCode_SocketError	An error was detected in the socket and charging is not possible.
ErrInfo_StatusUpdate	Error related with the update of a connector.

After this step, and through what we verified in the data analysis step, although this dataset contains no labels, the classification of each row was possible. As we want to detect any error that a connector can experience, we can classify the dataset defining a nominal case as any row where the status is a nominal value with no errors detected, following the specification of the OCPP protocol.

Figure 28, located below, shows the distribution between normal and abnormal cases in the dataset after performing the mentioned classification.

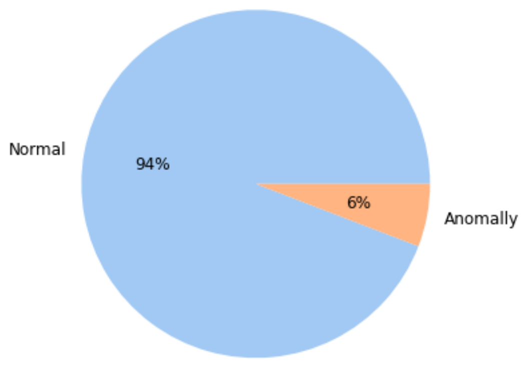


Figure 28: Percentage of normal and abnormal cases in the dataset after classification

Although it was possible to classify the data in this dataset, previously to that step, we also attempted other prepossessing strategies, comparing them by applying each prepossessed version of the dataset to several unsupervised machine learning algorithms, using the Silhouette and the Calinski Harabasz score as the evaluation metrics. The results are visible in tables 5 and 6.

Table 5: Impact of the preprocessing techniques over the Silhouette score on several clustering algorithms

Algorithm	**Control**	**EG**	**EEC**
Isolation Forest	0.535	0.411	**0.645**
KMeans	0.671	0.894	**0.963**
DBSCAN	0.987	0.990	**0.999**

Table 6: Impact of the preprocessing techniques over the Calinski Harabasz score on several clustering algorithms

Algorithm	**Control**	**EG**	**EEC**
Isolation Forest	3314.992	1776.488	**23338.644**
KMeans	11510.569	18002.526	**373915.272**
DBSCAN	13575.738	12213.884	**2949947.695**

The first technique consisted of applying numerical encoding over the status and one hot encoding of the remaining columns, with the additional removal of the timestamp column. This dataset was denominated as *Control*, as we only applied the minimum preprocessing required to format this dataset and its features into the format required by the chosen unsupervised algorithms. In the second technique, denominated EG, as a reference before in this section, we applied a similar process to the *Control* technique, but grouping the directly related binary columns which result from the application of one-hot encoding over the error affiliated features. In more detail, this means that, for example, for a given error code column generated by performing one-hot encoding over the *error_code*, *error_info* and *vendor_error_code* features, contain the same binary value in all cases, a logic OR is performed, maintaining the value of one if any of the columns hold this value, guaranteeing the preservation of an error even if only one of the binary error columns associated with that error is active. The final technique, named Encoded with Error Counts (EEC), consisted initially in applying the numerical and one-hot encoding as in the Control technique, creating a new feature called *error_counts*, which results from performing a count of the number of ones present in the binary error columns of each row. The final dataset generated by this technique contains only two features, the *status_encoded*, with the result of the numerical encoding over the original status column, and the *error_counts* with the number of errors detected for that row, counting the number of activated one-hot encoded column. As we can see, in tables 5 and 6, using the EEC technique, we seem to obtain an improvement on all algorithms for both scores. With these tests, we can confirm that the preprocessing steps done to the best performing technique were able to improve the clustering done by these algorithms and that the decreased number of features obtained by grouping the columns creating an error count feature produces similar or better results than the control test where all the one hot encoded columns were present. We, therefore, selected this version of the dataset by applying the set to the algorithms chosen for the model conception phase of the project, referenced before in section 3.2. The dataset was used with the same data splits for the semi-supervised tests, with 85% of the data for training and 15% for testing. For the supervised approach, however, we used 65% for training, 15% for testing and 20% for validation.

Although the dataset developed previously with the EEC technique can allow us to conceive the real-time anomaly detection models, as we also aim to predict future occurrences, we processed this dataset generating two new datasets, a status and error counts dataset. We used each dataset to train the LSTM models as explained in section 3.2. As we want to predict the values of the next hour, each dataset will be univariate, containing only one column, where each row consists of hourly measurements. To create the error counts dataset, using the *status_timestamp* column, we performed a count of the number of errors registered by the connector for each hour of the several days of operation, filling the missing hours with an error count of zero. Here, we can confidently fill these absent hours, as according to the OCPP documentation, the connector status will change if an error is detected, therefore forcing the charging station to send a *StatusNotification* package to the central system, with an error indication in the correspondent columns. Therefore, if no change was registered in that hour, we can safely assume that the value of the error count is zero. Regarding the creation of the status dataset, as connectors report the state only when there is a change, the frequency of state changes may vary from hour to hour, depending on the activity of the given connector. To solve this issue, the state for each hour in the dataset results from two conditions where if no anomalous state was detected in the hour, the resulting state value is the last state registered otherwise, the first occurrence of the anomalous state value is considered. As hourly data collection was not possible, this is the best compromise to reduce the loss of information within the hour while also considering the anomalous states that occurred while the connector was active.

After these steps, both datasets were min-max scaled for a feature range between zero and one, which, as proved before, can help LSTM forecasting models achieve better results, as these networks are sensitive to the scale of the inputs. With the data scaled, we converted both datasets to a sliding window variant with a window of 24 hours. Given that the dataset includes several missing hours, due to the way stations report the connector status, this window size was chosen in order to avoid the disadvantages of a smaller, but also a bigger window size. In this context, a smaller window size, would mean less information is considered per window and consequentially less patterns could be detected by the model. A large window size, on the other hand can lead to overfitting or possibly lead the model to miss patters due to the added information in each window, potentiated by the unbalanced nature of the dataset.

This dataset format will allow the models to receive historical status and error counts data to predict the next hour value, also used as the target value. For both forecaster models, the initial tests used a training set size of 60%, a testing size of 20% and finally, a validation size of 20%.

3.5 Summary

In this chapter, we initially explained the methodology followed during the development of the project, in this case Cross-Industry Standard Process for Data Mining (CRISP-DM). During this development, we used several technologies, such as TensorFlow and Keras, for the deep learning models implementation, like the LSTM and MLP classifier models. Scikit-learn, was used as it provided implementations for the machine learning algorithms we will use, the Isolation Forest, DBSCAN, KMeans and One-Class SVMs, as well as data scalers and evaluation metrics. Additionally, regarding data and file manipulations, we used the Numpy and Pandas Python packages with the Seaborn and Matplotlib packages, mainly used during the data analysis step.

Following this, we explained the two stages defined for the experiments conducted during this study. The first one, where the proposed approach aims to perform supervised and semi-supervised anomaly detection over the connector status data, using the previously referenced algorithms and the status and error count values of the *Status Notification* package. In the second stage, we aim to develop the proposed CFN model, composed of three modules. The first module is responsible for the status prediction, with an ensemble of two LSTM status forecasting models, which allows us to predict the status of the next hour using a weighted average of each model's results. The second module uses a single LSTM forecaster network to predict the error counts value for the next hour. Finally, the third module contains the real-time detection model, which receives as input the combined predicted values of the two previous modules to output a final anomaly classification value for the next hour. Regarding the status model ensemble of the first module, the first forecaster, denominated as OSF, corresponds to an optimised model result of a manual hyperparameter search, and the second model corresponds to the application of the weighted MSE loss function to the OSF model, to which we attributed the designation of WOSF.

We later analysed the connector status dataset identifying relevant information to solve the problem in question. Through this analysis and using domain knowledge, we verified that the dataset included a small percentage of connectors with an error state or an error registered. We also identified relationships between the state of the connector and the error types reported in the dataset. Additionally, by manually analysing the dataset, we detect patterns in the anomalous behaviour of the connectors, such as alternating the state in quick succession between *Available* and *Faulted* states during a transaction. We also analysed the structure of the charging transaction dataset, where we verified the presence of some errors identifying and analysing the possible causes, for example, for a power loss event. Through the data analysis, we spotted evidence of irregularities on the connectors or the local power grid, as anomalous patterns were verified in the accumulated power curves, but even more evidently in the wattage, amperage and voltage information sent by the charging stations. Unfortunately, due to the lack of data regarding the vehicle's battery and the charging station capabilities and charging method, using the power curves in the conception of a machine-learning model was not possible. This fact is due to the variables that can influence these curves both on the length and values of a transaction, raised during the data analysis, some of which include: the vehicle's total battery capacity, maximum power input capacity, maximum power transfer allowed by the charging station and the percentage of the battery at the beginning of the transaction.

In the data preprocessing step, we attempted several preprocessing techniques, varying the grouping of the one-hot encoded error columns and numerically encoding the status column, as there is a standard order for the status values during nominal transactions. We performed tests over the preprocessing techniques, evaluating their effects on the performance of multiple unsupervised algorithms with the Silhouette coefficient and Calinski Harabasz score as the evaluation metrics. Here we confirmed that the dataset with the numerical encoded status and error count values, which we named EEC, was the best performing technique. Through the OCPP documentation, the classification of this dataset was also possible, resulting in a dataset containing 94% nominal and 6% abnormal cases. We selected this dataset to be applied to the real-time detection algorithms, using a training set size of 85% and testing size of 15%, for the semi-supervised, and 65% for training, 15% for testing and 20% for validation for the fully supervised approach.

Finally, and considering we also aimed to predict the occurrence of anomalies, this dataset was later used to generate two unvaried datasets. The first one, containing the status values in each hour, where if we register a status of *Unavailable* or *Faulted* within the hour, the first occurrence of each one of these states is the value for that hour, otherwise, we used the last valid registered state. If we verified that no state value was reported in the hour, we considered an *Available* state for the connector on that hour. The second one, contained the error counts for each hour in the dataset, where missing hours were filled with a value of zero. Both datasets were afterwards min-max scaled with a feature range between zero and one, using a locked data split of 60% for training and 20% for validation and testing.

Chapter 4
Results and Discussion

4.1 Real Time Anomaly Classification

As stated before, concerning the real-time anomaly classification model, we experimented two distinct methods, a semi-supervised method and a supervised method. Regarding the semi-supervised method, in the case of the Isolation Forest and KMeans algorithm, the final predictions, obtained for the test set, were based on a threshold which we manually tuned to optimise the performance. For the Isolation Forest, this threshold was applied over the distribution of the anomaly scores, while with the KMeans, we used the minimum distances to the closest centroid, verifying and evaluating the results after each change using the labelled data and supervised learning metrics. The evaluation metrics chosen for all models trained were: precision, recall, f1-score and accuracy. For the One-Class SVM model, we obtained the final labels using the distances between each case to the separating hyperplane, also defining a threshold parameter over these distances. When using the DBSCAN model to obtain new predictions, we conceived a Nearest Neighbours model, trained over the training set with the default parameters. Applying this model to the test set allowed us to acquire the indexes of each new point to its neighbours. Using this method, we classify a new case as anomalous if we detected a noise point within the neighbourhood, or as nominal point otherwise attributing it to the most common cluster among its neighbours. For this model we did not apply a threshold parameter, obtaining the final labels through this Nearest Neighbours model.

For the supervised approach, we used an MLP classifier, with an architecture including an input layer with two neurons, one for each input feature, followed by a hidden layer with ten neurons, using the *ReLU* activation function, and finally the output layer with a single neuron and the *Sigmoid* activation function. This last layer outputs a probability value where we consider any case above 0.5 as anomalous, otherwise as nominal. For this model's optimiser, we selected Adam and evaluated the model using the same supervised evaluation metrics discussed before.

In table 7, located below, we can see the classification results obtained by each model in the test set with the default hyperparameters.

Table 7: Classification results using default hyperparameters

Algorithm	**Hyperparameters**	**Precision**	**Recall**	**F1-score**		**Accuracy**
				Negative Class	**Positive Class**	
Isolation Forest	*n_estimators* = 100 *max_samples* = auto *contamination* = auto *max_features* = 1 *bootstrap* = False	0.73	0.92	0.91	0.64	0.86
KMeans	*n_clusters* = 8 *init* = k-means++ *max_iter* = 300 *algorithm* = lloyd	0.44	0.50	0.93	0.00	0.88
DBSCAN	*eps* = 0.5 *min_samples* = 5 *algorithm* = auto	0.95	0.50	0.94	0.02	0.89
One Class-SVM	*tol* = 0.001 *gamma* = scale *kernel* = rbf *shrinking* = True *nu* = 0.5	0.57	0.58	0.27	0.25	0.26
MLP	*n_neurons* = 10 *learning_rate* = 0.001 *epochs* = 30 *batch_size* = 32	1	1	1	1	1

After reviewing the results with the default parameters, we observed that the MLP network correctly identified all test cases. While the initial performance appeared ideal, we manually tested the model with unseen cases, such as a connector with a *Faulted* status but no reported errors. In this scenario, the model misclassified the case as nominal despite the anomalous status being present. This was the expected behaviour since the classifier was never trained on such cases, as they were absent from the dataset. This limitation is one reason we believe unsupervised learning algorithms when combined with supervised data, might outperform the classifier model. To enhance the performance of the unsupervised learning algorithm, we conducted an extensive hyperparameter search using the Scikit-learn implementation of grid search with cross-validation. Since these algorithms are unsupervised, we defined a custom scoring function that evaluates the accuracy score of the model by comparing its results to the target labels. This approach is applied to the DBSCAN, Isolation Forest, and One-Class SVM models, as the Scikit-learn implementation allows us to obtain an anomaly classification class value. In the case of the KMeans algorithm, which does not provide an anomaly classification value, we calculated the minimum distances and determined a default threshold by taking the mean value of those distances, which allowed us to apply the same scoring function used for the other algorithms. However, since the threshold is a crucial hyperparameter that can significantly affect performance, we manually adjusted and tested this value for the best-performing combinations from the grid search of each algorithm. The full hyperparameter search space for each algorithm can be seen in table 8, located below.

Table 8: Hyperparameter search space for each algorithm

Algorithm	**Hyperparameter Search Space**
Isolation Forest	*n_estimators* = 20, 40, 60, 80, 100 *max_samples* = auto, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 *contamination* = auto, 0, 0.1, 0.2, 0.3, 0.4, 0.5 *max_features* = 1 *bootstrap* = True, False
KMeans	*n_clusters* = 1, 2, 3, 4, 5, 6 *init* = k-means++ *max_iter* = 100, 200, 300, 400, 500, 600 *algorithm* = lloyd, elkan
DBSCAN	*eps* = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 *min_samples* = 5, 10, 20, 40, 60, 80, 100, 200, 300 *algorithm* = auto, ball_tree, kd_tree, brute
One Class-SVM	*tol* = 0.001, 0.0001, 0.000001 *gamma* = auto, scale, 0.001, 0.01, 0.1, 1, 10, 20, 50, 100 *kernel* = rbf *shrinking* = True, False *nu* = 0.01, 0.1, 0.3, 0.5

Regarding the Isolation Forest, we varied the *contamination* parameter, testing values from 0 to 0.5 in increments of 0.1. We also attempted the *auto* setting, enabling the automatic threshold determination. Considering that the dataset contains a low percentage of outliers, we expect the lower contamination values to favour the model's performance. We varied the *n_estimators* parameter between 20 and 100, incrementing it by 20 at each step. Through manual hyperparameter tuning, we found that values above 100 seem to have no impact on improving the algorithm's performance. Hence, we chose the value range discussed previously. The *max_features* parameter, which determines the maximum number of features used in each isolation tree, was set to a value of one to include all features of the training set. The *max_samples* parameter was tested with the 'auto' setting, as well as values ranging from 0.1 to 0.9, incrementing by 0.1 at each step. Finally, for the *bootstrap* value, we tested both *True* and *False* options, as their impact can depend on the nature of the problem and data. Enabling it allows the creation of isolation trees with more randomised subsets of data.

For the DBSCAN algorithm, we focused our attention on changing the *eps*, the minimum number of samples (*min_samples*), and the algorithm used internally by the Nearest Neighbours module of the DBSCAN implementation. We varied the *eps* value between 0.1 and 0.9, incrementing in steps of 0.1. For the *minimum_samples*, we tested values in the range of 20 to 100 in increments of 20, but also for higher number with 200 and 300. Finally, we experimented with all the available options for the *algorithm* hyperparameter. The expected result, given the distribution of anomalies in the dataset, is that a lower *eps* value will provide better results.

Using the KMeans algorithm, we varied the *n_clusters* in increments of one, testing all values between one and six, where we expect that a value of two will provide an increase in performance, as, through the elbow method, we verified that this would be an appropriate value. For the initialisation method, we used k-means++, which enables positioning the initial centroids based on an empirical probability distribution of the data, hoping to achieve faster convergence compared to the random method. Additionally, we tested several *max_iter* values ranging from 100 to 600 in increments of 100 and two K-means algorithms, *lloyd* and *elkan*.

In table 9, located at the top of the next page, we present the optimised unsupervised algorithm's results after the hyperparameter search and manual optimisation of the best combinations using the threshold hyperparameter. Considering the results from the algorithms tested, we verified that almost all of them were able to correctly identify all test cases, apart from the KMeans and One-Class SVM algorithms, which failed to classify 45 and 56 cases, respectively. Both algorithms classified the failed cases as nominal, while the target label was anomalous. Although the results in table 9 show a very high test accuracy, precision, recall and f1-score for all models, these results were expected, as the dataset structure is simple and the target variable can easily be interpolated from the two dataset columns, which was in fact what allowed the initial data classification process to occur, as described previously in section 3.4, chapter 3, resulting in a high correlation between the features and the target.

Table 9: Classification results after hyperparameter tuning

Algorithm	**Hyperparameters**	**Precision**	**Recall**	**F1-score**		**Accuracy**
				Negative Class	**Positive Class**	
Isolation Forest	*n_estimators* = 100 *max_samples* = auto *contamination* = 0.1 *max_features* = 1 *bootstrap* = True *threshold* = 0.04	1	1	1	1	1
KMeans	*n_clusters* = 2 *init* = k-means++ *max_iter* = 100 *algorithm* = lloyd *threshold* = 1.05	0.99	0.90	0.99	0.89	0.98
DBSCAN	*eps* = 0.1 *min_samples* = 300 *algorithm* = auto	1	1	1	1	1
One Class-SVM	*tol* = 0.001 *gamma* = 0.01 *kernel* = rbf *shrinking* = True *nu* = 0.2 *threshold* = 0	0.99	0.88	0.99	0.87	0.97

To choose which model to use as the last step on our proposed architecture, we focused our model assessment on the evaluation metrics chosen. Here the KMeans and One-Class SVM models were the only ones with failed test case predictions, obtaining the lowest scores, even though the overhaul results were high, failing to identify some anomalies in the test dataset. Focusing our attention on the DBSCAN and Isolation Forest, as the objective of this model is to use the predicted state and error counts to output a final anomaly classification value, solely through the evaluation metrics, we can state that both of these algorithms can achieve this goal with the same confidence. However, when manually tested with new cases, we verified a desirable property in the Isolation Forest algorithm. Despite that in the dataset, there was no status value of *Faulted* with no errors registered, the Isolation Forest correctly classified this unseen case while still identifying all test cases. As we did not verify this behaviour in the DBSCAN and MLP models, the generalisation observed in the Isolation Forest model is superior, as it captures all the nominal cases while accounting for new anomalous state values.

4.2 Anomaly Forecasting over Connector Status Data

In this section, we approach the conception, optimisation and results for the two forecasting modules that compose the CFN model. The status forecasting models are approached in section 4.2.1, where we also describe the model ensemble performed between the OSF and the WOSF models. The errors forecasting model is approached in section 4.2.2 describing the model conception, the manual optimisation steps taken and the final results for each forecaster. For all the forecasters developed we analyse the results using the Mean Squared Error (MSE) loss, with additional information regarding the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) values present in appendix A for the status forecaster tests, and appendix C for the error counts forecaster tests. Regarding the status ensemble, we employed supervised learning metrics as we utilised the classification process previously described in section 3.4, only applied to the status feature.

4.2.1 Status Forecasting Model

Regarding the status prediction model, we initially defined a stacked LSTM model, with three LSTM layers (with 64, 32 and 16 units), with a final *Dense* layer with the *Sigmoid* activation function. This function was chosen as it outputs a value between 0 and 1, matching the min-max scaling done to the dataset. We used the Adam optimiser with the default learning rate, a batch size of 128 and trained for 50 epochs, achieving a training MSE loss of 1.4495 compared to a test loss of 1.1207. Analysing the training and validation loss curves, presented in figure 29, we can verify that the training loss is higher than the validation loss, demonstrating an overfitting situation starting from approximately the third epoch.

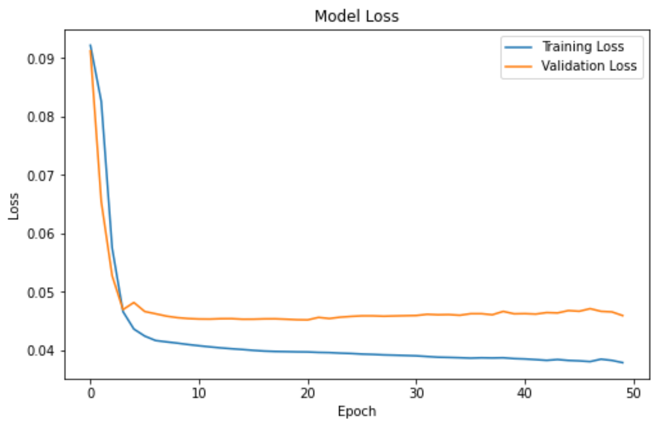


Figure 29: Learning curves of the first LSTM status forecaster network

Plotting the predicted versus the real results, we can verify that the model seems to be able to predict status values only below the anomalous values (6 and 7), both in the training and test sets, as figures 30 and 31 demonstrate.

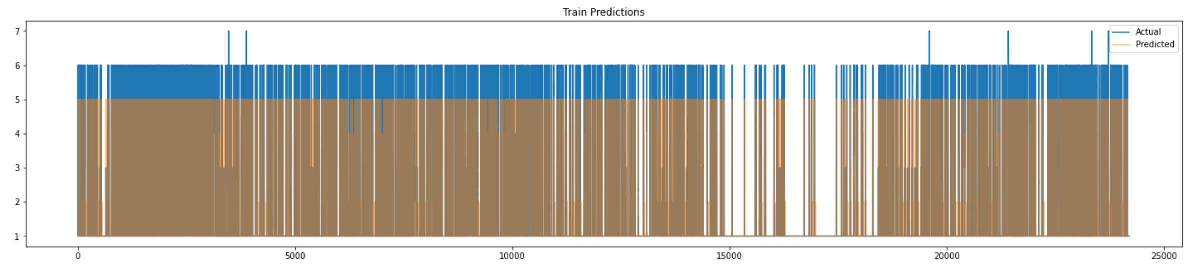


Figure 30: Real and predicted results of the first status forecaster in the training set

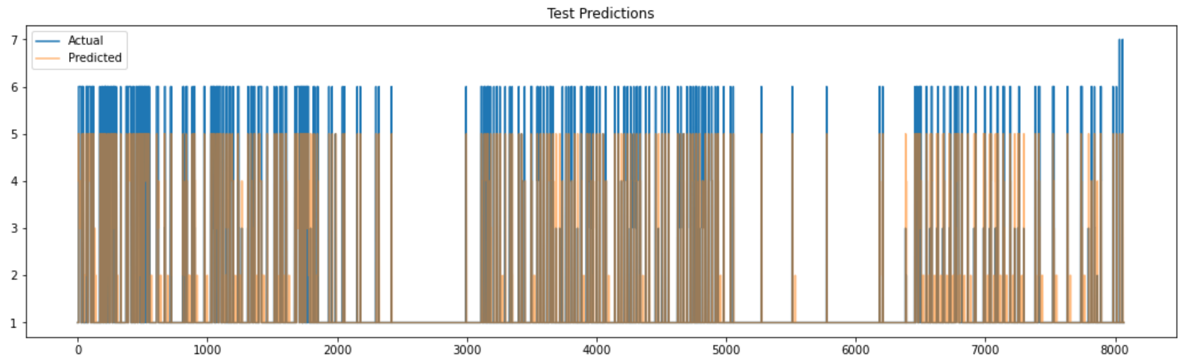


Figure 31: Real and predicted results of the first status forecaster in the test set

In an attempt to reduce the overfitting and hopefully improve the prediction of the anomalous status values, as referenced before, we performed a manual hyperparameter search testing the parameter combinations shown in table 10, located below.

Table 10: Hyperparameter search space for the status LSTM forecaster

Hyperparameters	**Tested Values**
Dropout rate	0, 0.1, 0.3, 0.5, 0.8
Learning rate	0.0001
Epochs	50, 100, 150, 200
Batch size	16, 32, 64, 128, 512

Based on the results of the first test, we chose to experiment with the effect of introducing regularisation with a single dropout layer between the first two and last LSTM layers. Here we varied the dropout rate from 0.1 to 0.8, hoping to reduce overfitting, following the same logic when variating the remaining hyperparameters. Regarding the learning rate, we opted for a lower value of 0.0001, as we frequently observe overfitting signs when testing the default value of 0.001. Regarding the epoch number, we chose to experiment with values between 50 and 200, increasing in intervals of 50, as we found that increasing the number of epochs, specifically for tests with smaller batch sizes (16, 32 and 64), resulted in insignificant to no changes (less than 0.0001) in the measured MSE loss values. Finally, and as the batch size can significantly impact the occurrence of overfitting, we experimented with batch size values ranging from 16 to 128, but also with a higher value of 512.

After the hyperparameter search, the best combination was achieved with a dropout rate of 0.8, a learning rate of 0.0001, and a batch size of 128 for 200 epochs. Through this, we were able to obtain an improvement in the learning curves and the predicted results, as we can see in figures 32, 33 and 34. In this test, we measured a training loss of 1.4544 compared to a test loss of 1.0972, registering the lowest test loss score in all tests performed, therefore obtaining the OSF model.

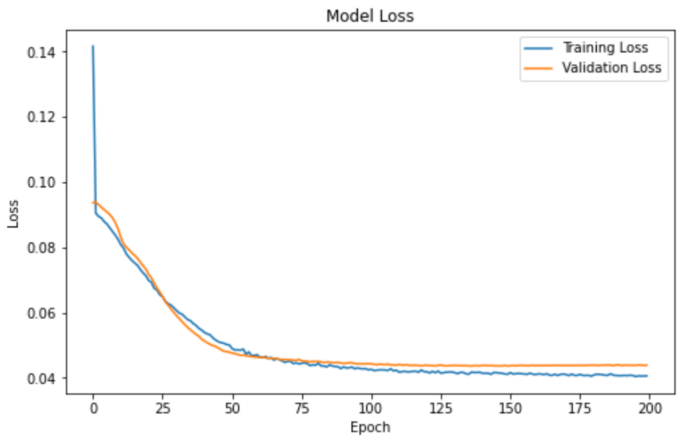


Figure 32: Learning curves of the OSF model

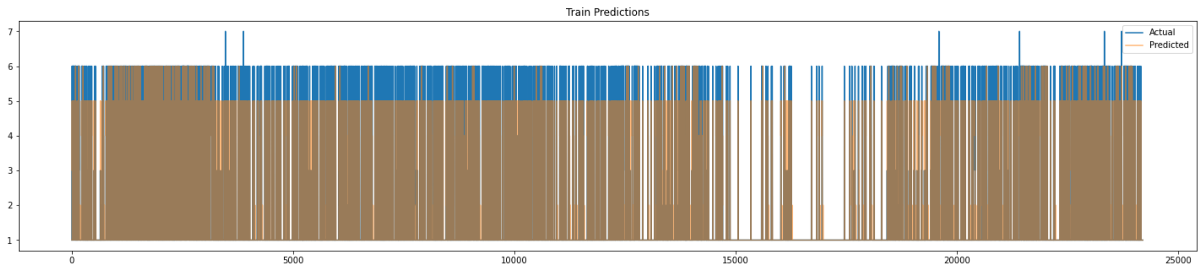


Figure 33: Real and predicted results of the OSF model in the training set

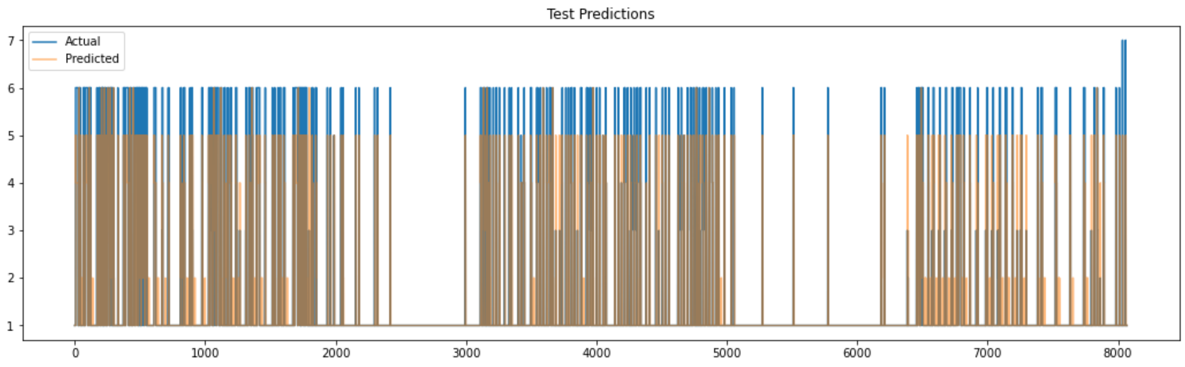


Figure 34: Real and predicted results of the OSF model in the test set

Following this, and in order to attempt to improve the OSF model's results in predicting anomalous cases, we applied the weighted MSE loss function, penalising the model from predicting values further away from the true value, as described before in section 3.2, obtaining the WOSF model. We used the same hyperparameter configuration used for the OSF model, performing a manual hyperparameter search for the weight parameter, testing all values between 0.1 and 2.9 in steps of 0.2. Through these tests, we found that by increasing the weight to a value of 2.9, we were able to increase the model's capacity in detecting anomalous status values. However, the test loss also increased compared to the OSF model, as we obtained 1.3040 in the test set and 1.7266 in the training set. The loss increase occurred as the penalisation led the model to predict higher values for cases with a status below *Faulted* (6) and *Unavailable* (7), as we can see in the real and predicted values for both the training and test sets in figures 36 and 37, respectively. In figure 35, we can also verify the impact of this change in the learning curves, which show significantly higher values for the training and validation loss.

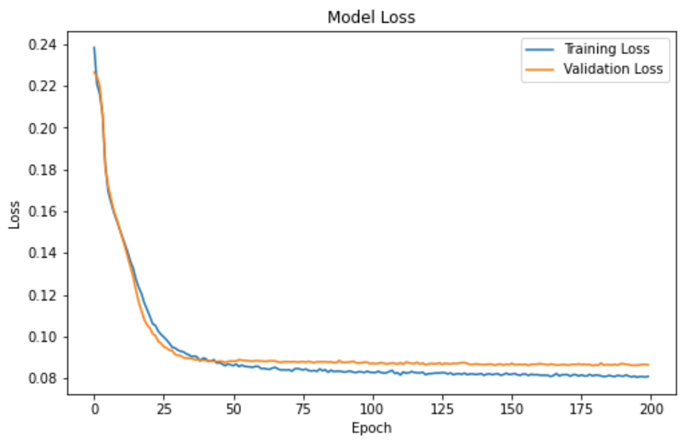


Figure 35: Learning curve of the WOSF model

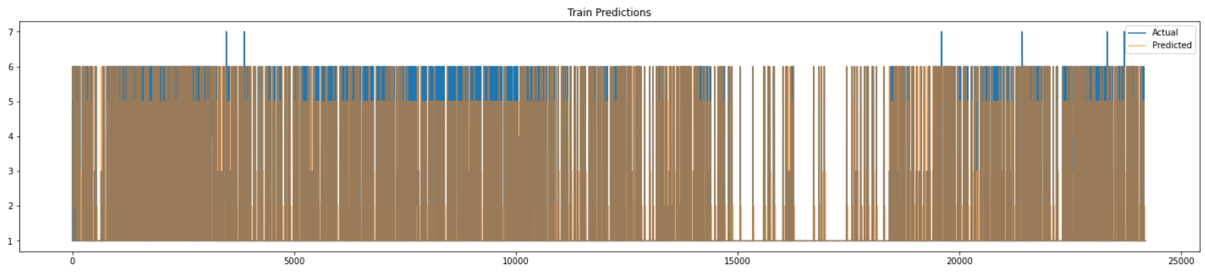


Figure 36: Real and predicted values for the WOSF model in the training set

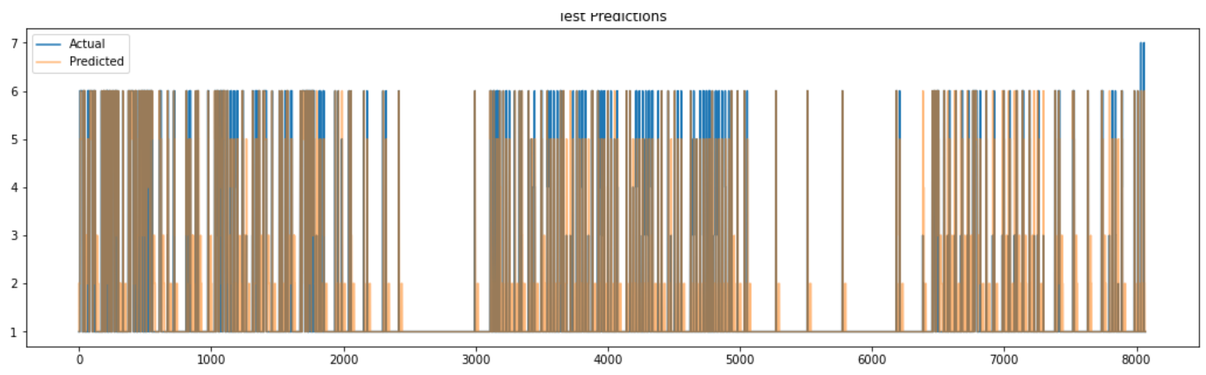


Figure 37: Real and predicted values for the WOSF model in the test set

From the figures above, we can see that the WOSF model also fails in predicting the anomalous status values of *Faulted* (6) and *Unavailable* (7). However, this model presents an improvement when compared to OSF model. Even though we verified that using a lower weight could result in better loss values, as we aim to perform a model ensemble, we chose to use this model in combination with the OSF, as it shows an increased capacity in predicting higher status values. As one model shows an increased forecasting power over the other in predicting higher status values, we can leverage both models' forecasting capabilities by tuning each model's weight in the weighted model ensemble.

To perform this ensemble, we tuned the weights of both models using supervised evaluation metrics. As the OSF model demonstrated a lower capacity in predicting the anomalous status values than the WOSF, we chose to variate this model's weight experimenting with the values of 1 and 0.8. For the WOSF model, we experimented with increasing the weight, experimenting with all values between 1 and 2 in steps of 0.2. We obtained the best combination using a weight of 0.8 for the OSF and a weight of 1.8 for the WOSF, acquiring the results shown in table 11. These weights were chosen in order to prioritise the predictions of anomalous cases, resulting in more false positives than negatives, which was desirable in this case, as we are attempting to forecast anomalies and want to capture the maximum number of cases. Detailed information regarding the ensemble test can be consulted in appendix B.

Table 11: Classification metrics of the status ensemble in the test set

Model	**Precision**	**Recall**	**F1-score**		**Accuracy**
			Negative Class	**Positive Class**	
Status Forecaster Ensemble	0.83	0.90	0.97	0.76	0.94

4.2.2 Error Counts Forecasting Model

For the error counts prediction model, we initially defined a baseline performance training a model with the same characteristics as the first status prediction model, with the only change being the final activation function set to linear, as we saw better results when compared to the sigmoid function. Analysing the learning curves of this model, shown in figure 38, we observe that the training loss is initially superior to the validation loss. As training continues, validation loss increases until it meets the values observed for the training loss at the end of training. In this test, we achieved a final training loss value of 1.3266 and a test loss of 1.0050.

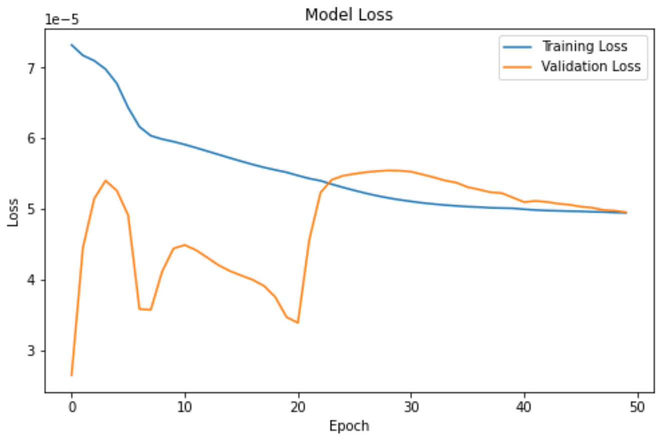


Figure 38: First test performed using the errors dataset

Considering the behaviour observed in the learning curves, we experimented with reducing the batch size to 8 and setting the learning rate to 0.0001. Through this, we verified a significant improvement in both the training and test loss values, with the training process stopping at epoch 24 due to the early stop callback, configured with a patience parameter of eight epochs. In this test, we saw the training loss decrease to 0.5507, while the test loss decreased to 0.0215. Through the learning curves in figure 39, and although the training loss was always higher than the validation loss, we can verify signs of overfitting close to epoch 15, where the validation loss started to increase, and the training loss continued to decrease.

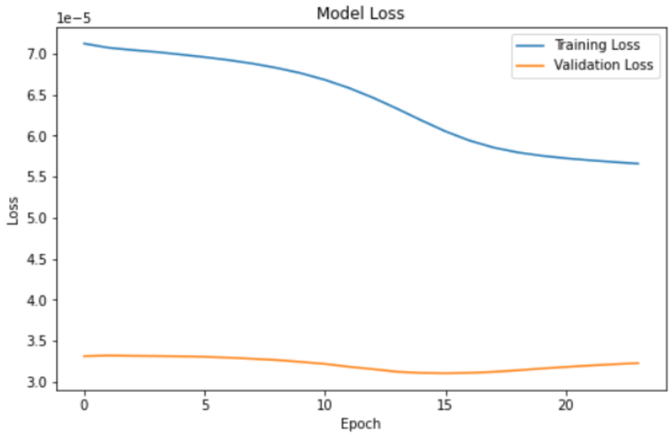


Figure 39: Learning curves after reducing the batch size and learning rate

We attempted several hyperparameter variations after the last experiment in an attempt to get the model to converge without the occurrence of overfitting. We obtained the best results by introducing regularisation, including a dropout layer with a rate of 0.1, between the second and last LSTM layers. We also set the batch size to 512 after experimenting with several batch size values (8, 16, 128 and 256) and increasing the training epochs to 500. Through the learning curves in figure 40, we can see that we were able to decrease both the training and validation loss while reducing the distance between the two values. Ultimately this resulted in a very similar test loss result, with a value of 0.0216, with a more considerable reduction in the training loss with a value of 0.456.

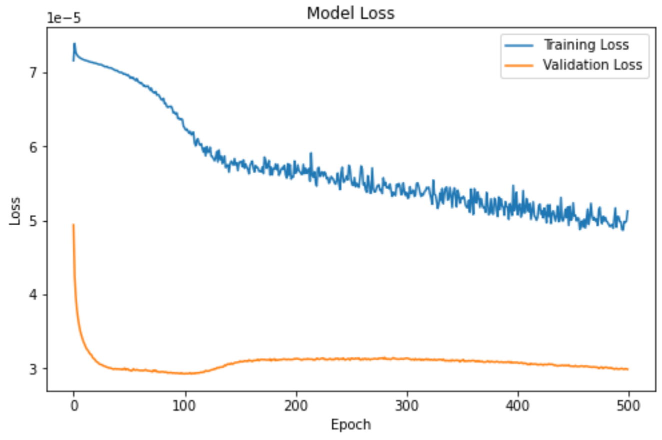


Figure 40: Learning curves of the model with dropout, batch size and epoch increase

Analysing the real and predicted values for the training and test sets, we can see a good coverage of the error peaks both in the training and testing sets, as figures 41 and 42, located at the top of the next page, demonstrate. However, when performing anomaly classification over the predicted and real values test sets, using the same method described before in section 3.2, we verified that the model failed in predicting anomalous instances, being largely affected by the unbalanced nature of the dataset which contained 16602 nominal compared to 30 anomalous cases. This distribution leads the model to obtain a low f1-score of 33% for the positive class, compared to 100% regarding the negative class. Detailed evaluation metrics results and further information related to the tests presented in this section can be consulted in appendix C.

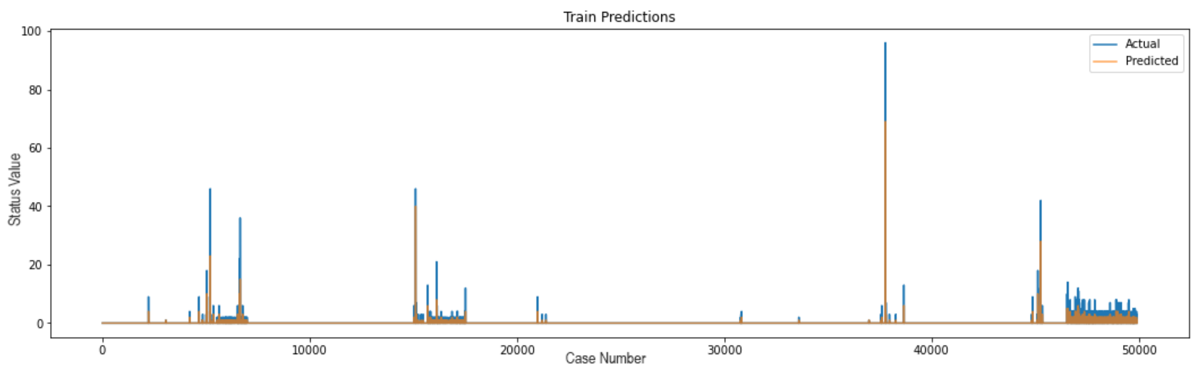


Figure 41: Real and predicted values for the errors training set

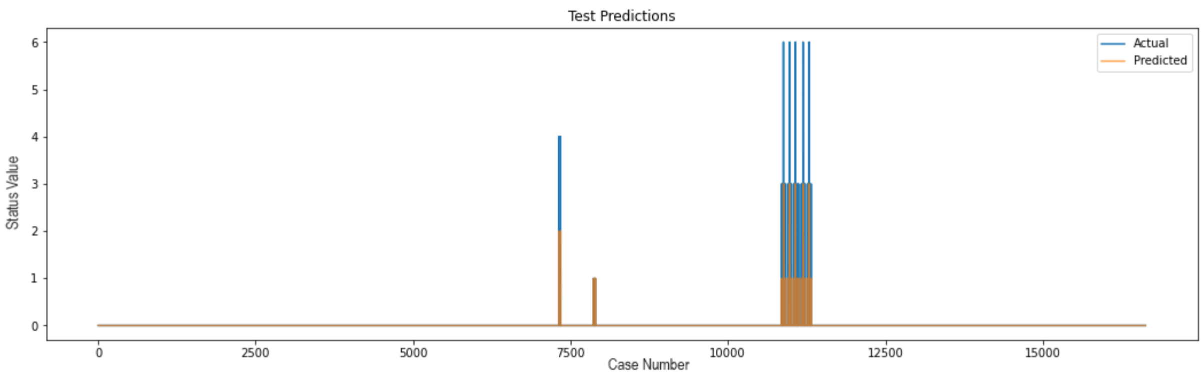


Figure 42: Real and predicted values for the errors test set

4.3 Connectors Forecasting Network (CFN)

With the real-time detection model and the LSTM forecasting models conceived, we built the final model architecture by combining the best-performing models and creating a single pipeline, therefore conceiving the CFN model. The forecasters take as input the status and error counts of the past 24 hours, outputting the predicted values for the next hour. These are forwarded to the isolation forest model, which outputs a final binary anomaly classification value where one indicates an abnormal case.

We evaluated this final model using a new dataset containing 19317 status request packets gathered from the start of April to the middle of June, with a total of 26496 observations, one for each hour after the creation of the univariate datasets. We chose to use a new dataset in order to guarantee that the models were not biased towards the data, as the previous dataset was entirely used for training and testing the forecasters. To this new dataset, we applied the same preprocessing steps, including the same data scalers used before in the model conception phase of both the status and error count forecasting networks. The classification metrics are presented in table 12, located at the top of the next page, together with figure 43, where we show the confusion matrix, and figure 44, with the ROC curve and AUC, as well as the Precision-Recall curve with the measured AUC-PR, through which we can assess the model's performance.

Table 12: Classification metrics for the CFN model

Model	**Precision**		**Recall**		**F1-score**		**Accuracy**
	Negative Class	**Positive Class**	**Negative Class**	**Positive Class**	**Negative Class**	**Positive Class**	
CFN	0.93	0.54	0.89	0.66	0.91	0.59	0.85

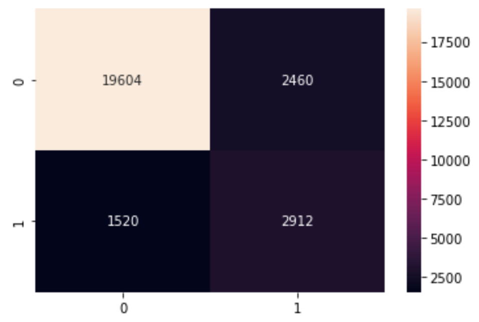


Figure 43: Confusion matrix of the CFN model

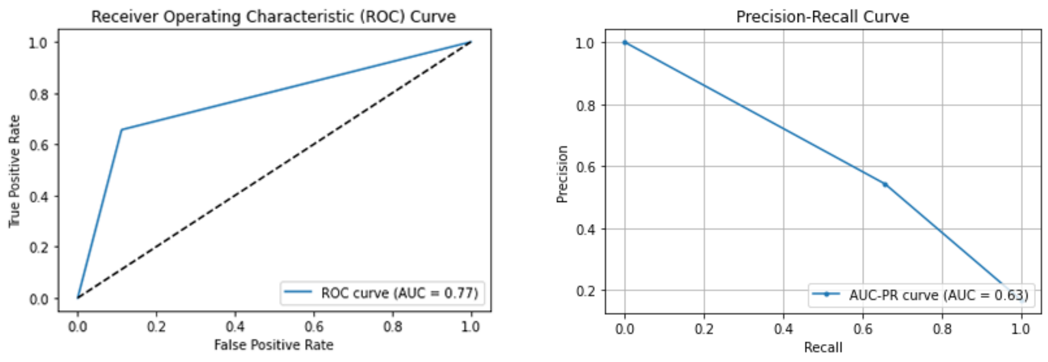


Figure 44: ROC and Precision-Recall curves of the CFN model

As we can see from the results of the CFN model presented in this section, the resulting model exhibits an f1-score of 59%, for the minority class. Considering the highly unbalanced nature of the dataset, in this case, with 22064 nominal compared to 4432 abnormal cases, this value is within the expected considering the individual performance of all the forecasters used, particularly the error counts forecaster, which demonstrated a lower capacity of identifying the anomalous cases with an f1-score of 33% for the positive class. In general, and considering the model's classification metrics, the model can correctly identify most of the nominal cases with high confidence and shows balanced precision and recall for the majority class. Regarding the minority class, the model exhibits a higher recall than precision, obtaining 0.66 and 0.54, respectively. From the ROC curve, we can see a reasonable performance of the model in distinguishing the normal from abnormal cases with an AUC of 0.77. Analysing the Precision-Recall curve, we can more evidently see that the model can still be improved, as it only achieves an AUC-PR score of 0.63, indicating some forecasting capacity for anomalies. Additionally, and through the classification metrics results, we can notice the impact of the status ensemble in the CFN results. We verify a similar proportion between the precision and recall values with close values for the negative class f1-score.

Chapter 5

Conclusion

5.1 Conclusions

With the work developed in this study, we can confidently answer the initial research questions raised by the problem in question. The data obtained from several charging stations regarding connector status data and charging transactions provides powerful insights into the current status of the OCPP network. This data allowed us to apply anomaly detection and forecasting techniques to the connector status dataset. However, in the charging transaction data, we were unable to employ machine learning techniques as the scale of both the time and measured values registered during a transaction could vary depending on several variables for which the OCPP protocol does not provide any information. Although this was verified, through data analysis, we detected anomalies in the wattage curves of the connectors demonstrating problems with the devices or in the local power grid, as some of these curves revealed a high oscillation of wattage values during a transaction (RQ1).

Using the connector status data, we tested several algorithms to perform the anomaly detection task using a supervised and semi-supervised approach. The algorithm which provided an increased performance when detecting anomalies regarding connector status values was the Isolation Forest, as it provided a clear separation between anomalous and normal status values while also classifying unseen values correctly (RQ2).

Following this, the Isolation Forest model was employed alongside two LSTM forecasting networks. One correspondent to an ensemble of two status forecasters, and the other includes a single forecasting model for the error count values. These two networks allow us to forecast the status and error counts for the next hour, using the previous 24 hours as input (RQ3). Here through a manual grid search of a selected hyperparameter space, we verified that the highly unbalanced nature of the dataset impacted both the conception of the status and error counts forecasters. Regarding the status forecasters, we were able to minimise the effects of the unbalanced dataset through the use of the weighted loss function, conceiving a model capable of predicting the anomalous status values of *Faulted* and *Unavailable* (6 and 7), which when combined with the best case from the hyperparameter search in a weighted model ensemble, significantly improved our results in capturing the anomalous instances.

By merging the status ensemble, the error counts forecaster, and the Isolation Forest model in a single pipeline, we created the CFN model, which was evaluated using a new connector status dataset. Through the evaluation metrics, we verified that this model was able to correctly forecast and capture most of the nominal instances, failing mostly in capturing the anomalous cases with an expected increased percentage of false positives when compared to false negatives due to the tuning performed over the status ensemble weights. However, we still verified a positive f1-score of 59%, with the model identifying 2912 anomalous cases from a total of 4432 in a dataset with a total of 26496 samples. Although this model could still be improved, through the presented results, we verify a valuable outcome in forecasting anomalies using the connector status data that can be integrated into an existing maintenance system, automating the analysis and reporting of anomalies in the existing maintenance system (RQ4).

5.2 Future Work

Considering the results obtained in this study, and although we were able to answer the initial research questions, it's crucial to acknowledge the constraints in time that affected the scope of this research. Due to limited time resources, we were unable to explore some potentially valuable techniques that could have further improved the models' performance. Additionally, the unexpected lack of information provided by the OCPP protocol forced us to look for potential solutions, which in the case of the charging transactions dataset, we did not find due to the high variability and missing information on very impactful variables related with the power curves.

However, as future work, an expanded collection of connector status data with a smaller collection window could lead to improvements in the conceived models by offering an increased number of samples for each status value and the number of errors, possibly improving the capacity of the models to identify particular patterns and perhaps reducing the overfitting observed due to the dataset's imbalanced nature. Further experimenting with errors LSTM, investigating the possibility of performing a model ensemble, could improve the general performance of the CFN model. Additionally, other architectures could be applied, in an attempt to enhance the performance, such as LSTM Autoencoders. For instance, a study conducted by Gensler et al. (2016) [45] demonstrated the effectiveness of an LSTM Autoencoder network approach in accurately forecasting the generated power using the power curves of renewable energy power plants. Integrating LSTM Autoencoders into our anomaly detection and forecasting models could potentially enhance their ability to identify unusual patterns and improve forecasting accuracy. By assessing the performance of LSTM Autoencoders and comparing it to the techniques already used, we can gain valuable insights into their applicability to our specific problem domain.

Regarding the charging transactions, a new dataset could be conceived with the missing data regarding the variables mentioned in the data analysis section, attempting to implement a machine learning solution capable of detecting or forecasting the values of the transaction's power curves.

5.3 Final Thoughts

With the growth of the charging networks and subsequent infrastructure, the difficulty of supervising and maintaining the network's components increases, creating a need for a more automated solution to more efficiently and preemptively manage any occurrences that can lead to the interruption of these services. Machine learning solutions are becoming increasingly attractive in facilitating and aiding the decision-making process, not only in the charging networks and electric vehicle fields but also in other areas, creating more effective and efficient maintenance systems.

This study can be viewed as a contribution to making these maintenance systems a reality for the OCPP charging networks, exploring the data offered by charging stations and techniques for anomaly detection and forecasting. Additionally, we were able to submit a scientific article detailing the core conclusions obtained by the development of this work which was published in the ICAART, 16th International Conference on Agents and Artificial Intelligence.

In conclusion, we hope this work is proven valuable for other studies and can lead to future advancements field of charging networks, maintenance systems and machine learning for OCPP networks.

Bibliography

[1] Open Charge Alliance. “Open Charge Allience: Home.” In: 2022. url: https://www.openchargealliance.org.

[2] We Can Charge. “Homepage.” In: 2022. url: https://wecancharge.com/.

[3] Z. Garofalaki, D. Kosmanos, S. Moschoyiannis, D. Kallergis, and C. Douligeris. “Electric Vehicle Charging: A Survey on the Security Issues and Challenges of the Open Charge Point Protocol (OCPP).” In: *IEEE Communications Surveys & Tutorials* 24.3 (2022), pp. 1504–1533. doi: 10.1109/COMST.2022.3184448.

[4] Open Charge Alliance. “Open Charge Point Protocol 1.6.” In: 2022. url: https://www.openchargealliance.org/protocols/ocpp-16/.

[5] H. H. H. Jesper E. van Engelen. *A survey on semi-supervised learning*. 2020. doi: 10.1007/s10994-019-05855-6. url: https://doi.org/10.1007/s10994-019-05855-6.

[6] Middle East Technical University. Computer Engineering Department. “K-Means Clustering Demo.” In: 2014. url: https://user.ceng.metu.edu.tr/~akifakkus/courses/ceng574/k-means/.

[7] Google Developers. “k-Means Advantages and Disadvantages.” In: 2022. url: https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages?hl=en.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou. “Isolation Forest.” In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. doi: 10.1109/ICDM.2008.17.

[9] C. Jung, Y.-G. Lee, J.-W. Lee, and S. Kim. “Performance Evaluation of the Multiple Quantile Regression Model for Estimating Spatial Soil Moisture after Filtering Soil Moisture Outliers.” In: *Remote Sensing* 12 (May 2020), p. 1678. doi: 10.3390/rs12101678.

[10] S. Hariri, M. C. Kind, and R. J. Brunner. “Extended Isolation Forest.” In: *CoRR* abs/1811.02141 (2018). arXiv: 1811.02141. url: http://arxiv.org/abs/1811.02141.

[11] B. Tang and H. He. “A local density-based approach for outlier detection.” In: *Neurocomputing* 241 (2017), pp. 171–180. issn: 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.02.039. url: https://www.sciencedirect.com/science/article/pii/S0925231217303302.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.” In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231. url: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[13] Chire. *File:DBSCAN-Illustration.svg*. 2011. url: https://upload.wikimedia.org/wikipedia/commons/a/af/DBSCAN-Illustration.svg.

[14] Y. Chen, L. Zhou, S. Pei, Z. Yu, Y. Chen, X. Liu, J. Du, and N. Xiong. “KNN-BLOCK DBSCAN: Fast Clustering for Large-Scale Data.” In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.6 (2021), pp. 3939–3953. doi: 10.1109/TSMC.2019.2956527.

[15] C. Cortes and V. Vapnik. “Support-Vector Networks.” In: *Machine Learning*. 1995, pp. 273–297.

[16] E.-H. A. Rady and A. S. Anwar. “Prediction of kidney disease stages using data mining algorithms.” In: *Informatics in Medicine Unlocked* 15 (2019), p. 100178. issn: 2352-9148. doi: https://doi.org/10.1016/j.imu.2019.100178. url: https://www.sciencedirect.com/science/article/pii/S2352914818302387.

[17] D. Wilimitis. “The Kernel Trick in Support Vector Classification.” In: 2018. url: https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f.

[18] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt. “Support Vector Method for Novelty Detection.” In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. url: https://proceedings.neurips.cc/paper/1999/file/8725fb777f25776ffa9076e44fcfd776-Paper.pdf.

[19] Garima. “One Class SVM(OC-SVM).” In: 2020. url: https://medium.com/@mail.garima7/one-class-svm-oc-svm-9ade87da6b10.

[20] K. Yang, S. Kpotufe, and N. Feamster. *An Efficient One-Class SVM for Anomaly Detection in the Internet of Things*. 2021. doi: 10.48550/ARXIV.2104.11146. url: https://arxiv.org/abs/2104.11146.

[21] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau. “Autoencoder-based network anomaly detection.” In: *2018 Wireless Telecommunications Symposium (WTS)*. 2018, pp. 1–5. doi: 10.1109/WTS.2018.8363930.

[22] S. M. Tripathy, A. Chouhan, M. Dix, A. Kotriwala, B. Klöpper, and A. Prabhune. "Explaining Anomalies in Industrial Multivariate Time-series Data with the help of eXplainable AI." In: *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*. 2022, pp. 226–233. doi: 10.1109/BigComp54360.2022.00051.

[23] I. Shafkat. *Intuitively Understanding Variational Autoencoders*. 2018. url: https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf.

[24] J. Rocca. "Understanding Variational Autoencoders (VAEs)." In: 2019. url: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73.

[25] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. 2013. doi: 10.48550/ARXIV.1312.6114. url: https://arxiv.org/abs/1312.6114.

[26] A. Agmon. "Hands-on Anomaly Detection with Variational Autoencoders." In: 2021. url: https://towardsdatascience.com/hands-on-anomaly-detection-with-variational-autoencoders-d4044672acd5.

[27] H. Hadipour, C. Liu, R. Davis, S. T. Cardona, and P. Hu. "Deep clustering of small molecules at large-scale via variational autoencoder embedding and K-means." In: 2022. url: https://doi.org/10.1186/s12859-022-04667-1.

[28] R. Agarwal. *The 5 Classification Evaluation metrics every Data Scientist must know*. 2019. url: https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226.

[29] Scikit-learn Developers. *Metrics and scoring: quantifying the quality of predictions*. 2022. url: https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics.

[30] Google Developers. *Classificação: curva ROC e AUC*. 2022. url: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[31] Y. Liu, S. Hu, and T.-Y. Ho. "Vulnerability assessment and defense technology for smart home cybersecurity considering pricing cyberattacks." In: *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2014, pp. 183–190. doi: 10.1109/ICCAD.2014.7001350.

[32] Y. Liu, S. Hu, and T.-Y. Ho. "Leveraging Strategic Detection Techniques for Smart Home Pricing Cyberattacks." In: *IEEE Transactions on Dependable and Secure Computing* 13.2 (2016), pp. 220–235. doi: 10.1109/TDSC.2015.2427841.

[33] M. E. Kabir, M. Ghafouri, B. Moussa, and C. Assi. “A Two-Stage Protection Method for Detection and Mitigation of Coordinated EVSE Switching Attacks.” In: *IEEE Transactions on Smart Grid* 12.5 (2021), pp. 4377–4388. doi: `10.1109/TSG.2021.3083696`.

[34] N. Hotz. *What is CRISP DM?* 2022. url: `https://www.datascience-pm.com/crisp-dm-2/`.

[35] M. L. Waskom. “seaborn: statistical data visualization.” In: *Journal of Open Source Software* 6.60 (2021), p. 3021. doi: `10.21105/joss.03021`. url: `https://doi.org/10.21105/joss.03021`.

[36] J. D. Hunter. “Matplotlib: A 2D graphics environment.” In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. doi: `10.1109/MCSE.2007.55`.

[37] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. “Array programming with NumPy.” In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. doi: `10.1038/s41586-020-2649-2`. url: `https://doi.org/10.1038/s41586-020-2649-2`.

[38] J. Reback, jbrockmendel, W. McKinney, J. V. den Bossche, M. Roeschke, T. Augspurger, S. Hawkins, P. Cloud, gfyoung, Sinhrks, P. Hoefler, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, J. Darbyshire, R. Shadrach, M. Garcia, J. Schendel, A. Hayden, D. Saxton, M. E. Gorelli, F. Li, T. Wörtwein, M. Zeitlin, V. Jancauskas, A. McMaster, and T. Li. *pandas-dev/pandas: Pandas 1.4.3*. Version v1.4.3. June 2022. doi: `10.5281/zenodo.6702671`. url: `https://doi.org/10.5281/zenodo.6702671`.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python.” In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[40] T. Developers. *TensorFlow*. Version v2.6.0. Aug. 2021. doi: `10.5281/zenodo.5181671`. url: `https://doi.org/10.5281/zenodo.5181671`.

[41] F. Chollet et al. *Keras*. `https://keras.io`. 2015.

[42] M. Tan, S. Yuan, S. Li, Y. Su, H. Li, and F. He. “Ultra-Short-Term Industrial Power Demand Forecasting Using LSTM Based Hybrid Ensemble Learning.” In: *IEEE Transactions on Power Systems* 35.4 (2020), pp. 2937–2948. doi: `10.1109/TPWRS.2019.2963109`.

[43] M. Shepero, J. Munkhammar, J. Widén, J. Bishop, and T. Boström. “Modeling of photovoltaic production and electric vehicles charging on city scale: A review.” In: *Renewable and Sustainable Energy Reviews* 89 (Mar. 2018). doi: `10.1016/j.rser.2018.02.034`.

[44] J. Pereira and M. Silveira. “Unsupervised Anomaly Detection in Energy Time Series Data Using Variational Recurrent Autoencoders with Attention.” In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018, pp. 1275–1282. doi: 10.1109/ICMLA.2018.00207.

[45] A. Gensler, J. Henze, B. Sick, and N. Raabe. “Deep Learning for solar power forecasting — An approach using AutoEncoder and LSTM Neural Networks.” In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2016, pp. 002858–002865. doi: 10.1109/SMC.2016.7844673.

Appendix A

Detailed Status Forecaster Results

Table 13 demonstrates the detailed results for the first status forecaster, the OSF and the WOSF models, showing the train and test results for the MSE, RMSE and MAE loss.

Table 13: LSTM detailed status forecaster results

Model	**MSE**		**RMSE**		**MAE**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
First Status Forecaster	1.4495	1.1207	1.2039	1.0586	0.5437	0.4129
Optimised Status Forecaster (OSF)	1.4544	1.0973	1.2060	1.0475	0.5297	0.3994
Weighted Optimised Status Forecaster (WOSF)	1.7266	1.3040	1.3140	1.1419	0.6369	0.4888

Appendix B

Detailed Status Ensemble Results

Table 14 and figures 45 and 46, demonstrate the status ensemble classification metrics results and the confusion matrix for the training and test sets, respectively. The tests were performed with a weight of 0.8 for the OSF model, and 1.8 for the WOSF model.

Table 14: Detailed classification metrics of the status ensemble

Dataset	**Precision**		**Recall**		**F1-score**		**Accuracy**
	Negative Class	**Positive Class**	**Negative Class**	**Positive Class**	**Negative Class**	**Positive Class**	
Training Set	0.97	0.66	0.93	0.82	0.95	0.73	0.91
Test Set	0.98	0.69	0.95	0.84	0.97	0.76	0.94

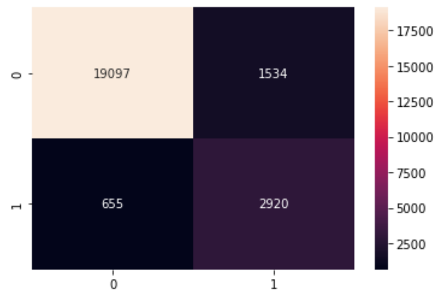


Figure 45: Confusion matrix of the status ensemble over the training set

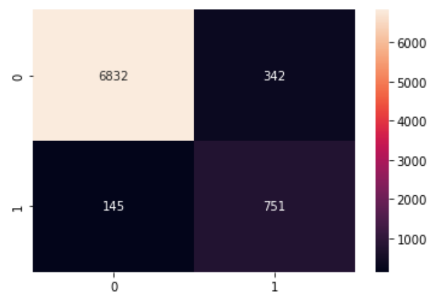


Figure 46: Confusion matrix of the status ensemble over the test set

Appendix C

Detailed Error Counts Forecaster Results

Table 15 shows the detailed results for the three tests in the order they were presented in section 4.2.2, showing the train and test results for the MSE, RMSE and MAE loss. Additionally, for the first test, we can verify the real versus predicted results in figures 47 and 48. The same can also be verified for the second test in figures 49 and 50. For the third test the real versus predicted results can be consulted in section 4.2.2.

Table 15: LSTM detailed error forecaster tests

Model	**MSE**		**RMSE**		**MAE**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
First Test	1.3337	1.0063	1.1548	1.0031	0.9957	0.9976
Second Test	0.5507	0.0215	0.7421	0.1467	0.0577	0.0065
Third Test	0.4565	0.0216	0.6756	0.1469	0.0613	0.0063

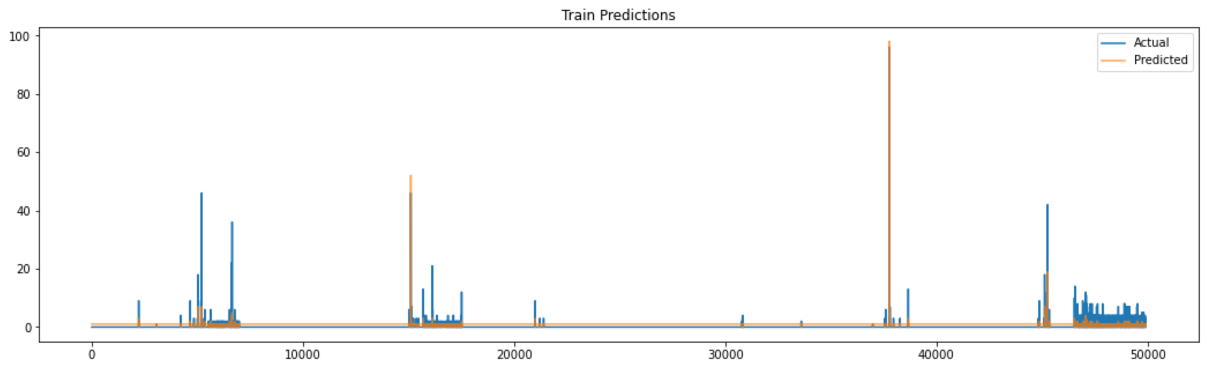


Figure 47: Real and predicted results of the first test in the training set

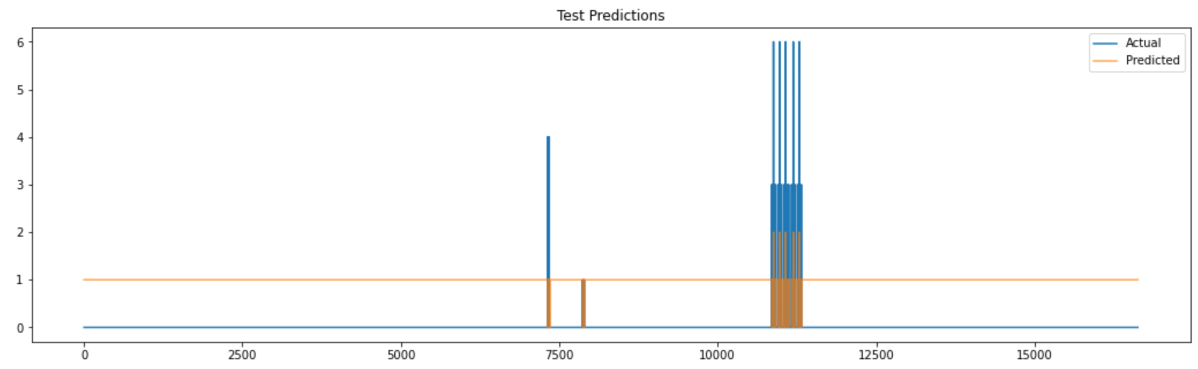


Figure 48: Real and predicted results of the first test in the test set

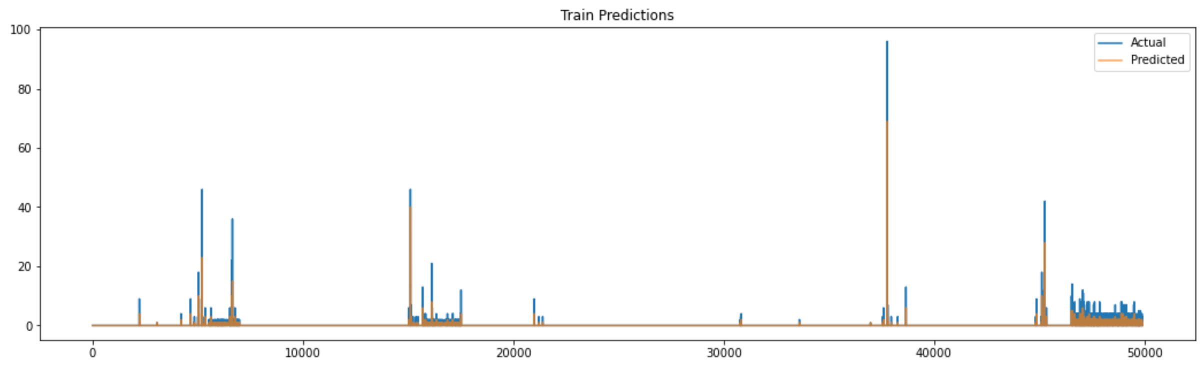


Figure 49: Real and predicted results of the second test in the training set

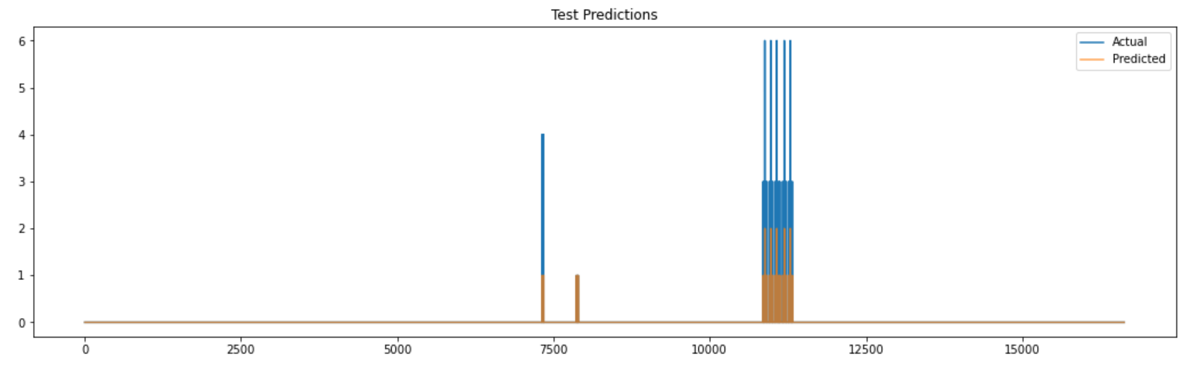


Figure 50: Real and predicted results of the second test in the test set

Annex I

Publications

I.1 Anomaly detection for preventive and predictive maintenance systems

Authors: Eduardo Eiras, Bruno Fernandes and César Analide.

Title: Anomaly detection for preventive and predictive maintenance systems.

Conference: ICAART, 16th International Conference on Agents and Artificial Intelligence

Submission Date: 2023, October.

Abstract: The rapid expansion of electric vehicle charging infrastructure and associated networks presents a growing challenge in managing any anomalies that may arise. The Open Charge Point Protocol (OCPP) is one of the standards used in the communication between the components of these charging networks. In the messages sent between the elements of these networks, we can find valuable information to detect anomalies and verify the state of those components. This study aims to utilise data gathered from the charging stations, applying machine learning techniques to conceive models capable of anomaly detection and forecasting. In order to accomplish this objective, we developed a real-time detection model, obtaining the best result for the Isolation Forest algorithm and two LSTM forecasters, one for the connector status and another for the number of errors reported by the charging connectors. The forecasters and real-time model were combined into a single pipeline, conceiving the Connectors

Forecasting Networks, which allowed us to forecast the occurrence of new anomalies.

Keywords: Anomaly Detection, Electric Vehicles, Machine Learning, Open Charge Point Protocol.

State of Publication: Waiting for approval.