Remote Condition Monitoring: Application to Railway Traction Systems

Rafael Mendes Borges

Thesis to obtain the Master of Science Degree in Aerospace Engineering

Supervisors:  Prof. Cláudia Alexandra Magalhães Soares
              Prof. Rodrigo Martins de Matos Ventura

Examination Committee
Chairperson: Prof. José Fernando Alves da Silva
Supervisor: Prof. Cláudia Alexandra Magalhães Soares
Member of the Committee: Prof. Maria da Conceição Esperança Amado

November 2018
To Sampaio
**Acknowledgments**

First of all, I would like to thank Professor Cláudia Soares and Professor Rodrigo Ventura for joining my desire to work in an area for which I have a special and long-standing interest, as well as their supervision throughout this work.

I am very grateful to Nomad Tech and Eng. Augusto Franco in particular for his interest from the beginning in collaborating with this work. A word of gratitude to my external advisors, Eng. Gonçalo Marques and Eng. Hélder Ribeiro, and also Eng. Luís Sequeira, who were available to help whenever necessary. I would like to extend my thanks to everyone I met at Amadora office, who shared with me their knowledge and experience.

My deepest thanks to my darling Beatriz, from whom I had an endless support along this journey. A special word of gratitude to my family, in particular Patrícia, Leonor and Alcina, who ever privileged me with their sweet care. Finally, to my friends for all the good moments, especially my longtime friend João.

A final word of appreciation to Eng. Dora Peralta and Eng. José Tomé who initially contributed to making this work possible.
Resumo

A manutenção é um tema crucial no setor dos transportes. A manutenção preditiva aumenta a disponibilidade das frotas, enquanto reduz custos. Esta estratégia tem como objetivo antecipar as falhas dos equipamentos, permitindo agendar atividades de manutenção em conformidade.

É apresentada uma abordagem baseada em ferramentas de ciência de dados para predição de falhas aplicada a sistemas de tração ferroviários, utilizando dados reais. Primeiramente, faz-se a análise exploratória dos dados e a redução de dimensionalidade. De seguida, o problema de engenharia é representado do ponto de vista da ciência de dados. Por fim, é desenhado um esquema para predição de falhas e são usados algoritmos de aprendizagem automática para aprender a partir dos dados.

Foram alcançados resultados importantes, porquanto todas as falhas foram corretamente previstas no conjunto de teste. Isto permite que o sistema seja reparado imediatamente antes da falha, evitando falhas durante a operação e as suas repercussões potencialmente graves. Todavia, a taxa de falsos positivos (0.40) é maior do que o desejado, conduzindo a operações de manutenção desnecessárias, mas é esperado que estas tenham um impacto menor do que não prever uma falha verdadeira. Do ponto de vista da operação, esta solução representa uma redução de custos se o gasto com uma reparação for pelo menos 2.22 vezes superior ao de uma manutenção, o que está claramente abaixo da estimativa para este custo relativo (3.7 – 6.1). São discutidas futuras orientações de trabalho, desde métodos alternativos de aprendizagem até soluções que foram desenhadas mas não realizadas por limitações de tempo.

Palavras-chave: predição de falhas, manutenção preditiva, abordagem baseada em dados, ciência de dados, sistemas de tração ferroviários, monitorização de condição remota
Abstract

Maintenance is a crucial aspect for any transportation operator. The predictive maintenance strategy improves fleet availability while reducing costs. This approach aims to anticipate equipment failures, allowing to schedule maintenance activities accordingly, preventing downtime and the severe consequences of breakdowns during operation.

This thesis presents a data-driven approach based on data science tools for failure prediction applied to railway traction systems, using real data collected by a remote condition monitoring system. The first steps include the exploratory data analysis and dimensionality reduction. Then, the engineering problem is represented from a data science point of view, for which the expert knowledge perspective is considered. Lastly, a scheme for failure prediction is designed and machine learning algorithms together with strategies for imbalanced datasets are used to learn from data.

Important results were achieved as all failures were correctly predicted in the test set, meaning that every fault is anticipated. This allows repairing the system just before its breakdown, avoiding failures during operation and their potentially serious repercussions. However, the false positive rate (0.40) is higher than desired, leading to unnecessary maintenance operations, but these are expected to have a lower impact than not predicting a true failure. From an operation’s perspective, this solution represents a cost reduction if a repair costs at least 2.22 times more than a maintenance activity, which is clearly below the estimated relative cost (3.7 – 6.1). Future work directions are discussed, from alternative learning methods to solutions which were designed but not performed due to time constraints.

Keywords: failure prediction, predictive maintenance, data-driven approach, data science, railway traction systems, remote condition monitoring
# Contents

Acknowledgments .................................................. v  
Resumo ................................................................. vii  
Abstract ............................................................. ix  
List of Tables ......................................................... xiii  
List of Figures ......................................................... xv  
Nomenclature ......................................................... xvii

1 Introduction ......................................................... 1  
1.1 Motivation ......................................................... 1  
1.2 Business Context .................................................. 1  
  1.2.1 Condition Monitoring of the Power Converters ................. 2  
  1.2.2 Remote Condition Monitoring (RCM) Data ...................... 2  
1.3 Contribution of this Work ......................................... 3  
1.4 Failure Prediction and the Aerospace Sector ...................... 3  
1.5 Thesis Outline ................................................... 4

2 Background ......................................................... 5  
2.1 Maintenance Strategies ........................................... 5  
2.2 Main Train Systems ............................................... 7  
  2.2.1 CP EMU 2300/2400 Series .................................. 7  
  2.2.2 Train Control Units ....................................... 7  
  2.2.3 Lusogate Interface ....................................... 8  
  2.2.4 Remote Condition Monitoring ............................. 8  
2.3 From Railway Engineering to Data Science ....................... 8  
2.4 Related Work .................................................... 9

3 Exploratory Data Analysis ........................................ 11  
3.1 Data Preparation ............................................... 11  
  3.1.1 Types of Data ........................................... 11  
  3.1.2 Extraction from Files .................................... 12  
  3.1.3 Data Cleaning .......................................... 13  
3.2 Exploratory Data Analysis ...................................... 13
List of Tables

5.1 Prediction scheme parameters. ........................................ 50
5.2 Oversampling method statistics. .................................... 51
5.3 Undersampling method statistics. .................................. 52

A.1 Statistics of the CCU variables before cleaning (M1). .......... 67
A.2 Statistics of the CCU variables after cleaning (M1). .......... 68
A.3 Statistics of a few TCU variables of context (M1). ........... 68
# List of Figures

2.1 Maintenance strategies comparison. ........................................... 6  
2.2 CP EMU 2300/2400 series train scheme. ................................. 7  
2.3 Problem formulation cycle. ....................................................... 9  

3.1 Temperature near one of the IGBT modules. ............................ 14  
3.2 Heat map of CCU GPS data (M1). .............................................. 15  
3.3 Statistics of CCU variables before cleaning (M1). ..................... 15  
3.4 Statistics of CCU variables after cleaning (M1). ....................... 16  
3.5 Correlation matrix of CCU variables. ....................................... 17  
3.6 Heat map zoom (M1). ............................................................. 17  
3.7 CCU variables missings (M1). ............................................... 18  
3.8 Histograms of CCU events. ..................................................... 19  
3.9 CCU events location by category. ......................................... 20  
3.10 TCU events histograms. ....................................................... 21  
3.11 TCU events location by category. ......................................... 21  
3.12 TCU events in time (M1). ..................................................... 22  
3.13 Detail of TCU important events occurring consecutively (M1). ....... 22  
3.14 Statistics of a few TCU variables of context (M1). .................. 23  
3.15 Division of data into blocks. ................................................. 24  
3.16 Example of data blocks for M1. ............................................ 25  
3.17 Severe and very severe TCU events. ...................................... 26  
3.18 Diagram of the final problem formulation. .............................. 27  

4.1 Cross-validation training errors using quadratic regularization. .... 35  
4.2 Cross-validation test errors using quadratic regularization. ......... 35  

5.1 SVM hyperplane. ............................................................... 38  
5.2 Confusion matrix for binary classification. ............................. 41  
5.3 The effect of undersampling on the separator. ......................... 44  
5.4 Original label histogram. ..................................................... 46  
5.5 Multiclass classification experiment results with imbalance methods. 47  
5.6 Binary classification results with imbalance methods. ................ 48
# Nomenclature

## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUC</td>
<td>Area Under the Curve.</td>
</tr>
<tr>
<td>CBM</td>
<td>Condition-Based Maintenance.</td>
</tr>
<tr>
<td>CCU</td>
<td>Central Control Unit.</td>
</tr>
<tr>
<td>CP</td>
<td>Portuguese Railways.</td>
</tr>
<tr>
<td>EDA</td>
<td>Exploratory Data Analysis.</td>
</tr>
<tr>
<td>EMU</td>
<td>Electrical Multiple Unit.</td>
</tr>
<tr>
<td>FN</td>
<td>False Negative.</td>
</tr>
<tr>
<td>FPR</td>
<td>False Positive Rate.</td>
</tr>
<tr>
<td>FP</td>
<td>False Positive.</td>
</tr>
<tr>
<td>GLRM</td>
<td>Generalized Low Rank Models.</td>
</tr>
<tr>
<td>GTO</td>
<td>Gate Turn-Off thyristor.</td>
</tr>
<tr>
<td>IGBT</td>
<td>Insulated-Gate Bipolar Transistor.</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis.</td>
</tr>
<tr>
<td>PdM</td>
<td>Predictive Maintenance.</td>
</tr>
<tr>
<td>PR</td>
<td>Precision-Recall.</td>
</tr>
<tr>
<td>QP</td>
<td>Quadratic Programming.</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function.</td>
</tr>
<tr>
<td>RCM</td>
<td>Remote Condition Monitoring.</td>
</tr>
<tr>
<td>ROC</td>
<td>Receiver Operating Characteristic.</td>
</tr>
<tr>
<td>SMOTE</td>
<td>Synthetic Minority Oversampling Technique.</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine.</td>
</tr>
</tbody>
</table>
TCU  Traction Control Unit.

TN   True Negative.

TPR  True Positive Rate.

TP   True Positive.
Chapter 1

Introduction

1.1 Motivation

Maintenance represents a crucial aspect for any transportation operator. As a first reason, one might think of safety, and it is surely right, but it is much more than that. Maintenance constitutes a significant fraction of operators’ expenses and everything made to optimize it can produce significant results. A third and no less important motive has to do with trains’ useful life, which depends on the quality of maintenance performed. Finally, fleet management is largely conditioned by maintenance scheduling. If the time needed for maintenance can be reduced, then availability will increase, thus making a better use of resources. On the other hand, to be able to predict failures and anticipate necessary corrections would be a significant improvement, reducing breakdowns during operation, which cause severe problems, and bring high financial losses.

This work arises from the recent focus of the rail sector to take advantage of all the developments in data science and machine learning, which have demonstrated their potential in different domains and applications, from disease diagnosis to aircraft maintenance. In this particular case, the goal is to make use of these tools to predict failures on the train traction systems.

The present dissertation project stems from a deep interest in working with the business world, and to learn more about the subject, by being closer to the problem. At the same time, this thesis allows me to contribute with possible solutions that can ideally benefit everyone. Thus, this project is a collaboration with a Portuguese company working on railways: Nomad Tech, Lda.

1.2 Business Context

This work addresses a real industry problem for which an engineering solution is being sought. Some trains from Portuguese Railways (CP) received a refurbishment on its traction converter, namely the fleet EMU (Electrical Multiple Unit) 2300/2400. The outdated gate turn-off thyristor (GTO) technology was substituted by the insulated-gate bipolar transistor (IGBT). In addition to this retraction, these trains were equipped with a Remote Condition Monitoring (RCM) system, allowing the extraction of relevant
data, in particular from the traction system.

Since this is the first time a refurbishment like this was done in these EMUs, there is an interest in deepening the knowledge on the new system, specially regarding its failures and modern approaches of maintenance.

1.2.1 Condition Monitoring of the Power Converters

Although these trains (from the early 90’s) were not originally equipped using IGBT, this technology is not so recent. Power converters on operation are exposed to varying environmental stress causes like thermal gradients, ambient temperature and electric power loss during operation. The power modules suffer thermomechanical stress, causing aging and failures; in fact, this thermomechanical stress is the major cause of failures [1]. The reliability of this equipment is an important aspect of this kind of systems since the semiconductor devices have high-power density and relatively high failure rates [1–3]. Real-time condition monitoring is crucial to increase the operation’s reliability as well as to reduce the maintenance costs. Power module failures can be divided into different categories: early, random and end-of-life failures. The early faults are difficult to anticipate using these methods because some of them are random. These methods are more adequate for monitoring the end-of-life failures, mainly due to wear out of the power module [1].

Different ways of monitoring this equipment are being used in laboratory experiments. Some measurements consist of voltage and resistance, gate signal and switch-time. Sensor-based methods are also used, as well as system identification techniques. These methods are in general hard to apply. Some of the disadvantages are related to the difficulty of measure small variations, high real-time requirements or hard algorithms. In addition to these technical problems, they are very expensive [4].

If these techniques are complicated to use in laboratory conditions, it is easy to understand why they are not being applied in operational conditions. Nevertheless, industry emphasizes the need of better condition monitoring methods for power electronic converters [2].

1.2.2 Remote Condition Monitoring (RCM) Data

The data collected from the traction system would ideally consist of the metrics referred above. But, as discussed, this is not easy to apply and is very expensive. For these reasons, the RCM system was designed to collect only the temperature of the power module ¹ and some flags indicating its status, at a sample rate of 1Hz. So, it is very far from the conditions used in laboratory experiments.

In addition to these serious constraints, the system that collects the data coming from the power modules is under development and some problems were detected when the data was analyzed. This beta stage implied a different approach for this work since arguably the most relevant portion of data could not be used. The testing stage of the RCM restricted this thesis to the data coming from two systems of the train: the central control unit (CCU) and the traction control unit (TCU). These sources collect information from the train main systems, being the former more generic information while the latter is dedicated to the traction system.

¹It is not the junction temperature but the temperature near the IGBT modules.
Considering the conditions mentioned above and the data that can be used, this work changed the initial idea of predicting failures in IGBTs to predict severe and very severe events related to the traction system, which can indicate problems with IGBT modules. These correspond to a total hardware block (severe event) and a short-circuit forced by the TCU (very severe event). However they are just registered failures, there is no continuous monitoring data coming from the modules as desired. The available data corresponds to some months of operation and just one train.

This deviation from the initial idea brought a more challenging problem.

1.3 Contribution of this Work

This work presents a predictive maintenance (PdM) approach using data science and machine learning methods. Data collected from two different train systems is used to predict failures on the traction system. The data is composed of data collected over time representing several variables monitoring the train and also events which represent failures. Some of these events have also several variables associated (variables of context), which are collected every time a failure is detected. The failures of the traction system are the focus of this work, particularly four important events that are the ones to be predicted.

The idea of this thesis is to use a black-box approach, as opposed to the traditional ones based on failure modes and rule engines [5]. This work follows the usual steps involved in a data science project, of course adapted to the circumstances that are particular to this case. The fact that this thesis follows a black-box approach does not mean that the way data is processed ignores the domain and application. This work involved an initial study on rail topics in general and then more focused on traction systems and the fleet used. This knowledge is very important throughout the thesis.

The main results of this work will be submitted to KDD'19 next February.

1.4 Failure Prediction and the Aerospace Sector

The concepts of predictive maintenance and failure prediction are very broad, being applied in a wide range of fields. Its use in rail was the motivation for this work, but it is just one example, and the same is true for aeronautics or space. The work developed has a minor dependence on the railway domain, while the major part is adjustable to many domains. The main tools used come from data science and machine learning.

So, this thesis emerges from this very broad concept of predictive maintenance and its main tools could also be applied to data coming from airplanes with the necessary adjustments. In fact, PdM has been used in aerospace, e.g. in aircraft engines [6]. These considerations show that this work is related to engineering in general and Aerospace Engineering in particular.
1.5 Thesis Outline

This document follows the structure described here. Chapter 2 introduces some background of maintenance methods and the trains and systems involved. It also includes an overview of related work. Chapter 3 presents the different types of data, describes the data analysis made and explains how the different types of data were used to build the dataset, together with the problem formulation. Chapter 4 presents the algorithms used for dimensionality reduction and the way they are applied to this particular case, including the results obtained. Chapter 5 describes all the topics related to the classification algorithms applied and strategies used to deal with imbalanced data. The results are shown and discussed. Finally, Chapter 6 presents the main conclusions and ideas for future work.
Chapter 2

Background

This Chapter introduces relevant topics for this work. Firstly, Section 2.1 describes the evolution of maintenance strategies. Secondly, Section 2.2 describes the main systems of the train, focusing on the object of this thesis. Then, Section 2.3 explains how the real railway engineering problem is interpreted and codified from a data science point of view, discussing the challenging interface between these two worlds. Lastly, an overview of related work is presented in Section 2.4.

2.1 Maintenance Strategies

Corrective maintenance can be considered the most basic way of doing maintenance. This is a reactive strategy: repair after the fault occurs. It is very expensive and inefficient. A different approach is preventive maintenance. It is a traditional strategy and contrary to corrective maintenance, it is based on prevention, as the name suggests. The idea is to make the necessary actions before the failure to avoid it [7].

The concept of preventive maintenance can be further specified. The first method, which is called planned maintenance, the most common, consists of replacing equipment based on its statistics of faults and other similar criteria. This implies that the equipment useful life is reduced and more maintenance is needed [8].

The concept of condition-based maintenance (CBM) emerges as an improvement over the previous approach. This strategy evaluates the real condition of equipment using monitoring practices, which require additional devices (e.g. sensors) and a control unit. It also needs a system able to collect and manage all data. Although it is more expensive in terms of implementation, generally achieves substantial cost reductions. Due to its philosophy, CBM implies a more adjustable maintenance scheduling [9].

The last method is called predictive maintenance, which is an improved CBM strategy. It tries to predict the time of failure, allowing maintenance to be planned accordingly and extending the equipment useful life. Compared to CBM, its implementation costs are higher but in general achieves better results. This technique requires intelligent systems capable of predicting failures based on condition monitoring [8].
The idea is not as recent as it seems. In fact, this kind of maintenance started before being conceptualized. A technician doing a visual inspection and using his knowledge to infer the right time to make a repair can be considered predictive maintenance. It is probably the most basic method [10]. Over time, maintenance started to use different instruments and make periodic inspections. These strategies allowed to have relevant information about the condition of the equipment. More advanced strategies include real-time condition monitoring, using sensors which are collecting data coming from several systems. The data can then be processed in several ways. One of the approaches, usually called a model-driven approach, consist of using rule engines based on failure modes. Another type of methods is known as a knowledge-based approach employing methods like fuzzy logic and expert systems [5, 11].

With time, more sophisticated methods of analyzing data were developed and the capacity for storing data significantly increased. Having more data allows better decisions since they are based on more information. It leads to enhanced reliability, reducing failures and optimizing availability, with lower costs. The current paradigm of PdM, generally referred to as a data-driven approach, takes advantage of data science, machine learning and big data tools. Conventional numerical algorithms and statistical approaches were the first methods to be used, followed by Bayesian networks and Hidden Markov Models, among others. The most recent include machine learning algorithms like neural networks, decision trees and support vector machines (SVMs) [12].

Figure 2.1 summarizes the ideas presented above. The evolution of maintenance strategies is reducing overall costs while improving availability. At the same time, equipment useful life is maximized, making repairs as close as possible to failure. Both points of view show PdM as the optimal maintenance strategy [6].

These techniques are usually part of a reliability-centered maintenance program, another important concept on maintenance. It is a more wide-ranging term, consisting of a process to define the optimal combination of maintenance strategies for a given system. It includes operational and financial practices, trying to manage the risk of equipment failure as efficiently as possible [13].

This engineering framework is used to define a complete maintenance regime. One example of maintenance methods integration inside reliability-centered programs could be using CBM or PdM as one primary failure management strategy [14].
2.2 Main Train Systems

This Section introduces some background related to the trains involved in this thesis. The train series is briefly presented as well as its main control and monitoring units, finishing with the RCM system that collects our data.

2.2.1 CP EMU 2300/2400 Series

The data used in this work comes from a passenger train operated by CP in Lisbon suburban area, being part of the fleet 2300/2400. This train from the early 90’s is an EMU, composed of four rail cars. The fleet is usually operated using two coupled trains (eight rail cars) but is also used with just one train. Even if coupled, the data comes only from that specific train and does not contain information about the other train.

To be consistent throughout the thesis, it is important to explain the meaning given to each of the following expressions: train or unit refers to four rail cars; vehicle stand for two rail cars (motor car + trailer car); M1 and M2 designate each vehicle that composes the train. M1 refers always to the same vehicle and the same is valid for M2. Figure 2.2 presents a scheme of the train, constituted by two vehicles: M1 and M2, e.g. M1 being the two rail cars on the left and M2 the remaining ones. The motor cars are on the edges and the trailers in the middle.

![Figure 2.2: Train with four rail cars: two motor cars in the edges and two trailers in the middle. Each motor car has two traction groups, one per bogie. CP EMU 2300/2400 series train scheme (extracted from [15]).](image)

The driver uses the cabin of one motor car, depending on the direction of travel. The traction motors are placed in the two motor cars. Each motor car has two traction groups (or traction circuits), one in each bogie \(^1\). So, the train has four traction circuits: M1 has two and M2 the remaining two. To simplify the notation, the traction group \(#j\) of vehicle Mi is referred to as Mi,Gj.

2.2.2 Train Control Units

The train has two main control units: the central control unit (CCU) and the traction control unit (TCU).

Central Control Unit (CCU)

The CCU is the core of the train control system. Its main functions are the supervision, control and command of the principal train parameters, including [16]:

- dynamics (traction, coasting and braking);
- other vehicle functions (e.g. doors, control of auxiliary converters, air conditioning);

\(^1\)A bogie is the undercarriage assembly of rolling stock incorporating the train wheels, suspension, brakes and, in powered units, the traction motors.
• train set configurations (single or multiple units).

**Traction Control Unit (TCU)**

The TCU is specially designed for the control and command of a traction converter, in accordance with the dynamic profile chosen by the driver. Its main functions include [16]:

• receiving the commands to be executed (coming from the CCU);
• generating the waveforms required for controlling the power converter;
• monitoring the power circuits, detecting any malfunctions and activate the required protections when necessary;
• transmitting to the CCU the information about the status of every traction group, including failures.

**2.2.3 Lusogate Interface**

A new electronic equipment was installed when the refurbishment on the traction converter was made, called Lusogate interface. It is the interface between the TCU and the driver units inside the new power converter. The driver is an electronic circuit used to establish the communication between the TCU and the IGBTs. This interface includes some sensors to extract monitoring data, such as temperatures.

**2.2.4 Remote Condition Monitoring**

The available data is collected by an RCM system developed by Nomad Tech and installed in the train used for this work. Several files containing the information were provided by them. These files are divided by system (CCU, TCU and Lusogate interface) and types of data. Details about the data and all the steps followed to extract it are explained in Section 3.1.

**2.3 From Railway Engineering to Data Science**

This Section is dedicated to a crucial part of this work: the interface between Railway Engineering and Data Science worlds. The process of achieving an appropriate problem formulation is discussed.

The problem addressed by this thesis was presented from a Railway Engineering point of view. As it is always necessary for real problems, a challenging process is followed to adapt the problem from the expert knowledge to the data science viewpoint.

Figure 2.3 summarizes a possible interpretation of the steps involved. In the beginning, an initial problem formulation was done with the information available from expert knowledge. Then, looking at the data, it was verified that the concept needed improvements. Then, deepen knowledge about the system was found and a new formulation was designed. Other difficulties appeared when changing again to the data science side. So, the loop represented in the box was made again and again until finding a final problem codification appropriated for both Railway Engineering and data science sides. Although it may seem simple, it requires several steps and careful analysis. Sometimes, small details force significant
changes and a complete new idealization of the problem. Once a suitable formulation is found, the initial problem can be adequately represented as a data science problem without compromising the real engineering context.

![Diagram of problem formulation process](image)

Figure 2.3: Several iterations are necessary to find an appropriate representation of the problem, which satisfies the requirements from both expert knowledge and data science domains. Problem formulation cycle.

### 2.4 Related Work

This Section describes some research works related to the subject of this thesis. PdM approaches using machine learning tools have been discussed in various studies, some of which address problems common to this work.

In [17], a machine learning approach for failure prediction is proposed. Huge volumes of data from different sources are used, e.g. detectors, maintenance operations and failures. The authors start by feature extraction, using aggregated statistics, such as quantiles, followed by removal of some noisy data. The next step is dimensionality reduction, for which principal component analysis (PCA) is used, reducing the number of features. Then, SVM algorithms are used to solve the classification problem. The goal is to predict if there will be an alarm in the next days. The dataset is imbalanced, which is why the authors apply some techniques, e.g. enforcing penalties for misclassification. They use the true positive rate (TPR) and false positive rate (FPR) as metrics. This work shows that this approach is able to predict failures on train bearings, achieving interesting results, with high TPR or low FPR rates, depending on whether the most relevant metric is TPR or FPR. As they stated, most of this work is not specific from rail domain and can be applied to several industries, from aircraft to chemical plant maintenance.

The authors of [18] propose an approach to predict train wheel faults by combining multiple classifiers using decision trees and Naïve Bayes algorithms. This work proposes a different evaluation method, joining the common metrics for imbalanced data with a reward function to reinforce early and penalize late alerts.
A log-based PdM approach for failure prediction in medical devices is proposed in [19]. The data comes from equipment event logs, labeled as normal/fault. The classifier intends to label both single events and events in an interval (bag), e.g. one week. Its label is binary, being set as failure whenever one instance reports a fault. The instances of a bag labeled as failure become negative examples; the instance from non-failure bags is transformed into one positive example by averaging all its instances. These examples together build the dataset for training, in which the authors apply classification algorithms based on SVMs. Finally, the prediction is based on the score of each individual instance from a bag, labeling the bag as a failure if any instance hits a given score, otherwise as negative.

More recent works are focusing their attention on big data, which is beyond the scope of this thesis. Although this work is not dealing with a considerable amount of data when compared to such applications, big data is a very important topic in this area. The next step of this work could be the integration of big data tools to make use of data coming from more trains, for example.

Recently, rail operators and infrastructure managers started to collect huge amounts of data, but these are not fully used due to the lack of adequate technologies to extract relevant information. Datasets having huge sizes can’t be analyzed using traditional approaches since they usually exceed the limits of a typical application. Recent studies are showing its potential, in particular on PdM applications [20]. The authors of [21] present an example by applying these tools to detect defects on rail tracks. The data collected is approximately 1 terabyte per day.
Chapter 3

Exploratory Data Analysis

This Chapter starts by introducing the data used in this work, in Section 3.1. The exploratory data analysis is presented in Section 3.2. Then, Section 3.3 describes the process of joining all the different types of data into a single dataset. Lastly, Section 3.4 describes the problem formulation.

3.1 Data Preparation

3.1.1 Types of Data

The information can be divided into three types: variables, events and variables of context. Both the CCU and the Lusogate interface have variables. For the case of CCU, the variables are mostly real-valued parameters of the train with an associated timestamp and GPS data, which are recorded whenever the train is working, at a certain sample rate. The variables of the Lusogate interface are similar but the parameters contain information about the power modules, e.g. temperatures and some flags indicating their status.

Events represent failures and, for that reason, they are stored whenever the train detects a failure. Each log has the fault description, the start and end timestamps, and the GPS coordinates relative to the location where the event started. Both CCU and TCU have specific events. Finally, the variables of context are exclusive from the TCU and they are collected every time a TCU event starts. They represent the values of several parameters and also flags indicating internal commands and states of the traction system. They could be integers or binary values (flags). Their values correspond to the instant when its associated event starts.

These types of data are stored in different ways. The CCU variables are stored in txt files, which correspond to less than 10 logs, in general. The same happens with the Lusogate data. Each line of these files has a timestamp and the associated variables. The logs from CCU have a frequency of $2\,Hz$ while the ones from Lusogate are registered at a rate of $1\,Hz$.

For the case of CCU events, the RCM system produces a comma-separated values (csv) file including all the events, where each line represents a different event. The TCU events and variables of context are stored in txt files. For each event, several lines contain all the information: the TCU event and
its variables of context. For events and variables of context, it doesn’t make sense to talk about the
temperature of reading; they happen whenever there is a failure.

The files containing events and variables of context from TCU (hereinafter TCU raw files) are more
difficult to read than the remaining files since they are not pre-processed like the CCU files. These txt
files contain reads of a stack of TCU events with their variables of context. The RCM system reads this
buffer with a certain cadence. This implies that the txt files contain repeated events because the buffer
can contain the same events in consecutive reads, until they are removed from the stack. Reading an
event which is not yet closed is also possible and the reason is that the buffer can be read while the event
is happening. These and other details are explored later in Section 3.1.3.

The volume of data used in this work corresponds to one train, as said before. The RCM system
began collecting data from CCU, TCU and Lusogate interface at different times. The data from CCU
covers five months and TCU almost four months. The Lusogate data covers a few weeks and no more
data was used since this data was discarded after its analysis.

As discussed later, this work uses the data collected when all the systems were recording.

3.1.2 Extraction from Files

As explained before in Section 2.2.1, the train is divided into two vehicles, which is why the data is
separated into two blocks, one per vehicle. In practice, this means that the data from M1 and M2 are
stored in independent files (except for CCU events). For that reason, the steps described below are made
for M1 and M2.

Considering that data from the systems is stored in different ways, the extraction of the data from
the files changes from type to type. The CCU variables consist of hundreds of thousands of txt files.
The approach to this is merging everything in a single txt file and then reading this big file containing
everything. This is possible because the files don’t contain duplicated data. The first step cleans the
headers and terminations of every single file before merging them. After this joining, the new file is used
to extract the data into a table. These steps are made using MATLAB. Since the Lusogate data is similar
to the CCU variables, the method of extraction is the same.

The CCU events (stored in a csv file) are easier and faster to read than the remaining data types. Easier
because the data is pre-processed and faster because the size is around one thousand lines. MATLAB is
used to convert the data in the file into a table. After this, the only step needed is to separate the data
of M1 and M2, since the same file stores all information, as said above.

The txt files containing the TCU data are the most complex to read. They are around thousands of
files. It is necessary to extract both events and variables of context from them. In this case, the first
step is merging the files into a single file and then reading the file line by line, since the structure is not
uniform. Some lines contain the information of the event and the lines below have all the correspondent
variables of context. These variables are dispersed in several lines which makes the reading harder.
3.1.3 Data Cleaning

In general, the files corresponding to posterior phases of processing by the RCM system are already clean and presenting all the information needed in a more convenient way. This is the case for the Lusogate data, CCU variables and events. On the contrary, the TCU raw files are more challenging.

As referred in Section 3.1.1, these files represent readings of the buffer containing TCU events, which cause the files to have repeated events or events not closed (not finished), depending on the time of reading. To solve this problem, while reading the large txt file, the code checks if the event was already read or not, ignoring it if necessary.

3.2 Exploratory Data Analysis

This Section presents the exploratory data analysis (EDA) of our dataset. This analysis was introduced by the author of [22] as a way of looking at the data, allowing a better comprehension of its main characteristics. The EDA can be done using multiple techniques, usually including graphical methods. This analysis is based on [23], where these methods are listed, as well as some important considerations.

The analysis is done for the various types of data separately. Firstly, some statistics of the data are presented. Then, histograms and figures are shown to better understand the data. Finally, the most important events are explored in detail.

Please note that due to the number of variables and events, some figures used in this Section contain only data from one vehicle. This is done in cases where the data from the other vehicle is similar and does not contain relevant information that justifies its detailed presentation.

3.2.1 Lusogate Data

The Lusogate data contains some variables, most of them parameters monitoring the power modules, e.g. temperatures near the IGBTs. There are also flag variables indicating their status. As the system is under development, many variables and flags are still not being used.

A quick analysis of the temperatures led to the discovery of faulty data. The temperature near the IGBTs has improbable values. There are temperatures of 6000°C or near 0°C and large variations in small time intervals, as shown in Figure 3.1.

At this point, it was found that the sensing system had problems. For this reason, the data from the Lusogate interface could not be used, since the system was not fixed on time. This is why the analysis of the Lusogate data was not pursued and this data was discarded from our work, as discussed previously in Section 1.2.2.

3.2.2 CCU Variables

As explained in Section 3.1, the CCU variables consist of several variables collected by the CCU of each vehicle. These can be divided into two categories: GPS data and train parameters.
GPS Data

The GPS data stored together with the remaining variables is composed of five parameters: latitude, longitude, speed, azimuth and a flag. The flag indicates if we have a GPS fix. The routes of the train and its operating frequency in each part of the route can be extracted from the GPS data. We use a heat map to visualize the train routes, as shown in Figure 3.2. A Python package (heatmap [24]) was used to obtain the heat map.

The heat map shows the expected results. This train operates on Sintra and Azambuja service lines. It is difficult to divide the map by service lines because they have coincident sections, which is why the map points out the ends of the routes. The zones having higher densities are common to different routes. The map shows the five routes operated by the train (same notation as in the heat map):

1. Lisboa - Oriente (O) ⇔ Sintra (SI)
2. Lisboa - Rossio (R) ⇔ Sintra (SI)
3. Alverca (AL) ⇔ Sintra (SI)
4. Lisboa - Rossio (R) ⇔ Mira Sintra - Meleças (M)
5. Lisboa - Santa Apolónia (SA) ⇔ Azambuja (AZ).

The heatmap evidences the higher density in the zones of the first four routes, which correspond to the Sintra line services. The lower density points are associated to the last route listed above, that is the Azambuja service line.

Train parameters

Different parameters of the train are collected: speeds, voltages, currents, mass, etc. Our first analysis is summarized in Figure 3.3, using a box plot for each variable (detailed statistics are shown in Appendix A).
Figure 3.2: The train travels most of the time in the routes of the Sintra line (routes 1-4). The Azambuja line (route 5) is less used. Heat map of CCU GPS data (M1).

Regarding the boxplots presented throughout this Chapter, the notation used is the following: on each box, the central mark represents the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively; the whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the red ‘+’ markers.

Two variables (ccu\_var\_18 and ccu\_var\_19) are constant, and thus not informative. In fact, we confirmed with domain experts that these variables are not in use.

Figure 3.3: Some variables have distinct outliers, which represent numerical overflows, and other variables are constant. Statistics of CCU variables before cleaning (M1).

There are some variables whose maximum values are not realistic or feasible. These maximum values
have also a significative value, around $2^{16}$. Therefore, it can be concluded that these represent numerical overflows. Another interesting aspect is revealed by looking at the values in time. These overflows occur only in the first days. It has an explanation: these values coincide with the days when the RCM system was being connected to the CCU and some adjustments were being made. This is why only these days have abnormal values and our solution was discarding the first days. For this reason, these days are counted as missing data, as discussed later. The main statistics are calculated again and the overflows disappear, as shown in Figure 3.4 (detailed statistics are shown in the Appendix A). Another statistical study is done by calculating the correlations between variables. As pointed out in Figure 3.5, some variables are highly correlated. Most of them are expected, e.g. instantaneous and reference speed.

Figure 3.4: The ranges are very diverse as the variables represent different parameters. Overflows and constant variables were removed. The high number of outliers may indicate some noise in the data. Statistics of CCU variables after cleaning (M1).

**Missing Data**

In general, the missing data in the variables can be divided into two categories: GPS missing and missing data of the train parameters, justified by their different patterns of missing data. Only one type of missing data is common to the train parameters and GPS. These missings are related to a characteristic of the system. The timestamp of each log is given by the GPS clock. So, every time the train starts working, the GPS takes some time to get the correct timestamp. This leads to some entries with wrong timestamps, easily found since they always have the year of 1970. These entries are residual and were discarded.

**GPS Data**

The system is always collecting GPS data. So, the missings in the GPS exist when the receiver cannot
Figure 3.5: Some variables are highly correlated, e.g. reference and measured speeds. Correlation matrix of CCU variables.

The missing rate of GPS data is 1.21%. The missings occur in two different situations: when the train starts working and when the location of the train does not allow GPS signal. The first reason happens everytime the train starts operating while the later depends on the train route. For example, Figure 3.6 shows a clear example. When the train goes to Lisboa - Rossio, it crosses a tunnel from Campolide to that station. There is GPS data at the station but no points are collected inside the tunnel.

Figure 3.6: No GPS data can be collected while the train crosses the tunnel from Campolide to Lisboa - Rossio. Heat map zoom (M1).

**Train Parameters**

The missing data in the train parameters has two justifications. The first is related to the system installation. Initially, the system was mounted in the train but not connected to the CCU, which is why only GPS data was collected. This happens because the GPS data is recorded by the RCM system itself, while the train parameters and events come from the CCU. Some days later, the system was connected to the CCU. However, as said before, they had to configure
the system and the data collected in the first days contain some errors, that is reason why it was discarded. Considering this, only by the end of January the train parameters started to be correctly collected. This explains all the initial missings in the train parameters which do not correspond to missings in the GPS data.

The second reason for missings in the train parameters explain the missings during operation. The system cannot read parameters and events at the same time. It can detect events when it is registering variables but then, at certain times, the system loads the buffer of events. During these reads, the train parameters are missing. The number of missing entries depends on the number of events to be loaded. For this reason, the missings in the train parameters occur in block. All the parameters have missings for the same logs. There is only one exception, which is the instantaneous speed recorded by the CCU (not the same as the GPS speed). It has some additional missings whose explanation was not given, but the number of missings is residual. The missing rate of the train parameters is 8.97%. This high value is explained by the reasons above.

In order to see these different missings, two plots were made, one representing the missings in the train parameters and the other in the GPS data. Figure 3.7 shows the differences, being the missing entries represented by ones and valid entries as zeros. Only the first weeks of available data are shown. It evidences the first days where only GPS data was collected and also the missings during operation, which are caused by different reasons in each case, as explained above. The gaps where neither missing nor valid entries exist represent the time intervals in which the train is not working, and of course no data is collected.

![Figure 3.7: The GPS data was collected from the beginning. Its missings are related to GPS coverage. Train parameters started to be registered later, what explains the first days full of missings. The subsequent missings are related to the system. The gaps without any entries in both plots correspond to when the train is not working. CCU variables missings (M1).](image)

3.2.3 CCU Events

This Section describes the analysis of the CCU events. Some statistics made with the CCU events are described below. The first analysis shows that M1 registered approximately 52% of the events and M2 the remaining 48%. Looking at the type of events, 33 different events were collected. Type of event means the same code, that is the same failure, which can occur in both vehicles, as explained previously. For example, a door failure in the motor car has the same code for M1 and M2. The distinction is made in the description with ‘M1’ or ‘M2’.
From those 33 codes, M1 had 31 different types of events and M2 had 27. Figure 3.8 shows the histogram containing the events for each vehicle, highlighting the number of occurrences for the most frequent events for each unit.

(a) Most frequent events of M1 are: internal event, low battery voltage and damaged door(s) in the motor car.

(b) Most frequent events of M2 are: non-critical air conditioning failure in the motor car, internal event and damaged door(s) in the motor car.

Figure 3.8: A few events occur several times, while most of them happen just sometimes, for both vehicles. Histograms of CCU events.

For M1, the most frequent failure is an internal event. It is also frequent for M2, as well as the problems with the doors in the motor car. The internal events are confidential information which is not accessible. For M2, the most common is the non-critical air conditioning failure in the motor car.

Figure 3.9 shows where the events happened. These were divided into categories. As shown, the CCU events are uniformly distributed along the routes, specially along the most used. The internal events are also spread along the lines in both cases. Failures related to air conditioning occur more in M2, as the Figure 3.8 points out. Finally, the remaining events do not show any particular pattern. As expected, events related to doors happen at stations.

Missing Data

Most missings in the CCU events are related to GPS coordinates. The explanation is the same as for the variables. If the GPS is not fixed, the event cannot store the position. The other type of missings
happens only a few times and these are events without end timestamp. They have the start timestamp but the end is empty. In terms of statistics, 11.7% of the events do not have GPS data and there are 0.36% (only three events) without end timestamp.

### 3.2.4 TCU Events and Variables of Context

#### TCU Events

In order to analyze the TCU events, some statistics are explored. M1 has approximately 54% of the events and M2 the remaining 46%. Looking at the types of events, there are 34 different codes registered. From these, M1 has all codes and M2 has 30 of them. Figure 3.10 shows the histograms for each vehicle. These give significant information, in particular about the most important events (described later in Section 3.4.1). The number of occurrences for these severe and very severe events are shown. The numbers reveal that 71% of these events happened in M1. They also show that the traction circuit #1 has more failures than the circuit #2 for both units. Moreover, the event code tcu-ev-AD (very severe event of circuit #2) was not registered.

In fact, the number of these important events is very small, considering that these are related to the state diagram (according to Section 3.4 presented later) which will be used to get the label for the classification algorithms. Later, in Chapter 5, this will be discussed in detail.

Figure 3.11 shows where these events happened. There is one location where both events tcu-ev-AA and tcu-ev-AC were registered for both units, which coincide with the unique tcu-ev-AC event of M2. This means that the event tcu-ev-AC also occurred isolated for M1. As expected from the explanation given below in Section 3.4, all tcu-ev-AC happen together with tcu-ev-AA, which is confirmed in the maps. Since no events tcu-ev-AD were registered, it means that the traction circuit #2 did not have very severe failures, for both vehicles.

Another way of seeing the events is by looking at their timestamps, as shown in Figure 3.12, for M1. It includes only the start timestamps; even if it included end times, they would coincide with the start,
(a) Occurrences of the most important events of M1: tcu.ev_AA (19), tcu.ev_AB (12), tcu.ev_AC (8), tcu.ev_AD (0). M2: tcu.ev_AA (10), tcu.ev_AB (5), tcu.ev_AC (1), tcu.ev_AD (0).

Figure 3.10: A total of 55 important events were registered. M1 registered much more of these events than M2, for both traction groups. None of the vehicles registered the very severe event of the traction group #2. TCU events histograms.

(b) M1: tcu.ev_AA, tcu.ev_AB, tcu.ev_AC, tcu.ev_AD
(b) M2: tcu.ev_AA, tcu.ev_AB, tcu.ev_AC

Figure 3.11: The locations are spread; almost all events occur in the most traveled lines. There are tcu.ev_AA isolated but all tcu.ev_AC occur together with a tcu.ev_AA. This crucial observation will make it possible to determine a state of the train, as presented in Section 3.4.1. TCU events location by category.
since the scale in Figure 3.12 is much bigger than the duration of events (a few seconds, in general). It shows that there are several events which happen in parallel, including the most important events (the three on top). Events are spread in time, without evident patterns. Figure 3.12 confirms in time what is shown in space in Figure 3.11: events tcu.ev.AC are always followed by events tcu.ev.AA. Figure 3.13 shows an example where it is possible to see these events occurring consecutively, now including start and end timestamps.

Figure 3.12: Several events happened in parallel. There are tcu.ev.AA isolated but all tcu.ev.AC occur together with a tcu.ev.AA (verifying the diagram of states). TCU events in time with the most important events on the top (M1).

Figure 3.13: Every tcu.ev.AC is followed by a tcu.ev.AA (example verifying the diagram of states). Detail of TCU important events occurring consecutively (M1).

The missings in the TCU events are exactly of the same type and with equal justification as those of CCU, as explained in Section 3.2.3.

Variables of Context

Every TCU event comes together with its variables of context. Even if the number of the TCU events is small when compared to the number of the CCU variables logs, the point is that every single TCU
event stores the value of hundreds of variables (939, to be exact). As explained in Section 3.1.1, these are binary and integer values.

Due to the considerable amount of variables, it is difficult to show the results of the analysis. The steps involved are similar to the ones followed to analyze the CCU variables. Contrary to what happens in the remaining types of data, in this case any errors (e.g. numerical overflows) were detected. Considering the huge number of variables, most of them without knowing their meaning, it is harder to detect problems.

Figure 3.14 shows the box plots of a few variables, for M1 (detailed statistics are shown in the Appendix A). It contains examples of integer and binary variables. As happened with the CCU variables, there are constant variables. In this case, they represent 52% (485 out of 939), which allows for a substantial reduction if they are removed. But they are not discarded at this stage and there is a reason for that, which is explained below in Section 3.3.

Figure 3.14: The ranges are diverse as the variables of context represent very different parameters; many of them are binary (flags). Several variables are constant. Statistics of a few TCU variables of context (M1).

Unlike what happens with the remaining types of data already discussed in this Chapter, the variables of context do not have any missings. Every registered TCU event has all its respective variables, even if the event itself has missing data on GPS coordinates or timestamps.

### 3.3 Dataset Preparation

This Section describes the steps needed to join all types of data into a single table, considering the specificity of each type. As discussed below when designing the problem formulation, the systems of interest are the traction groups and not the entire train or vehicles. This is why it makes sense to build four blocks of data, one per group.

For each vehicle, the different sources contain both general data which is associated with the whole vehicle but also data concerning each traction group. The CCU data (variables and events) is common to both traction groups of the vehicle, while the TCU data can be divided into data which is common to the whole vehicle and data specific to each traction group.
These considerations result into the four blocks of data shown in Figure 3.15. The left part of the blocks contain the data related to the whole vehicle while the right part is composed of the data specific to each traction group. It means that the information on the left part is duplicated, to assure that the common information exists in the rows of both traction groups for each vehicle. For example, the mass of the vehicle M1 is related to the entire vehicle and therefore belongs to the two left parts marked with M1; each traction group of that vehicle needs to have that information, which implies that it is duplicated. For the case of the voltage in the traction groups of M1, each right side of the blocks (marked with M1,G1 and M1,G2) has its own specific value and thus it is not repeated.

![Figure 3.15: The data can be divided into four blocks, each one corresponding to a specific traction group. The blocks of traction groups belonging the same vehicle share the data which is common to the whole vehicle. Division of data into blocks.](image)

The construction of the dataset is based on this division. It is necessary to define how the different types of data are encoded as features and also the way to get the label. Each row of the dataset (associated with a timestamp) is composed of all types of data by aggregating the features coming from each type. Every variable, event and variable of context is represented by a column in the dataset.

Given that the different types of data are not collected at coincident timestamps, the strategy to initialize each row of the dataset is the following: a new row is created with the timestamp held by the data to be added. The columns associated with CCU variables or TCU variables of context are filled with missings, and the columns of all events with zeros. Then, according to the type of data, the correspondent columns are filled with the data to be added, leaving the remaining columns as they were created.

Since the CCU variables are related to each vehicle, they belong to the left part of the blocks of Figure 3.15. The CCU variables are logs with an associated timestamp, thus they are easily converted to rows of the dataset where which variable will represent a feature. As this information belongs to the vehicle, each row is repeated for each traction group of that vehicle. For example, given a CCU log from M1, two new rows are created with the timestamp of the log and the CCU variables fill the respective columns. The remaining columns are not changed. Each of these two new rows belongs to the first and second top blocks, respectively.

The TCU data (events and variables of context) contains both information related to the whole vehicle or specific to each group. Therefore, the first task consists of dividing it into general and specific data. This was done by consulting the manual having the dictionary of types of data. Knowing this, the TCU events and variables of context related to the vehicle belong to the left part of the blocks (need to be duplicated), while the specific data will be placed on the right part of the blocks.

The right side of the blocks needs to be carefully built. In order to have a coherent dataset, its
columns for specific data must concern the same variable/event of each group. This explains why the TCU variables of context which were detected to be constant on the EDA were not removed at that point. For example, a specific flag related to one traction group could be constant. But the equivalent flag for another group could not. For this reason, one must not remove that variable as it is necessary to build the right side of the dataset blocks. These variables can be removed only when the equivalent variables are constant for all groups, i.e. if four variables representing the same for each group are constant, they can be removed. The TCU specific events need also to be processed. Similarly to the specific variables of context, its columns must concern the same event for each group. In some cases, equivalent events could not be registered anytime for one or some groups. In that case, the missing columns are added to that block(s). These two processes guarantee that the dataset has the same number of features for all traction groups, which is a requirement to have a single dataset containing all types of data.

The TCU variables of context have an associated timestamp, which is used to the create the new rows of the dataset. After dividing these variables into specific and general variables, the data is inserted into the new rows of the dataset, where the general variables are duplicated, similarly to happened before.

After adding all the rows, they are sorted by timestamp. At this point, all data was added except CCU and TCU events. The events are more challenging since they are not logs with an associated timestamp. They have start and end timestamps. The strategy to encode their information is done as follows: each event is a feature in the dataset. Then, for each event, the start and end timestamps are used to create new rows. The row for the start timestamp is created and its associated column set as one. The same is done for the end timestamp. These encode only the beginning and the end of the event. The time in the middle has to be also codified. This is done by checking the timestamps of all rows and setting the rows with timestamps in between the start and end of the event as ones. This strategy makes each column associated with events a binary feature which is one for the timestamps in which the event is happening and zero otherwise.

A final step is required. For each block, the rows having repeated timestamps are merged. Each resulting row contains the data separately added to those rows with the same timestamp. At this point, all the data was added to the dataset. The structure of the dataset shown in Figure 3.15 can be detailed to show the types of data, as illustrated in Figure 3.16 (only the top half relative to M1, the one for M2 is similar).

<table>
<thead>
<tr>
<th>CCU variables</th>
<th>CCU events</th>
<th>General TCU vars. context</th>
<th>General TCU events</th>
<th>Specific TCU vars. context</th>
<th>Specific TCU events</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCU variables</td>
<td>CCU events</td>
<td>General TCU vars. context</td>
<td>General TCU events</td>
<td>Specific TCU vars. context</td>
<td>Specific TCU events</td>
</tr>
</tbody>
</table>

Figure 3.16: The features of the dataset come from the blocks of data: the left blocks contain the general data associated with the vehicle while the blocks on the right contain the data specific to each traction group. The general data is duplicated. Example of data blocks for M1.
3.4 Problem Formulation

In order to correctly label each data point in the dataset, it is necessary to encode the state of the traction systems of each vehicle. As explained before, several steps were needed to achieve the final representation of the problem.

3.4.1 Problem Codification

The main details which were being figured out and included are presented below:

1. There are two severe and two very severe events which represent the failures to be predicted.

2. Each of the four traction circuits has two individual events (one severe and one very severe), as shown in Figure 3.17.

3. The very severe event is always followed by a severe event; a severe event can happen without any very severe event.

4. In some situations, a train reset done by the driver may recover from a severe event (perhaps preceded by a very severe event) to a normal condition.

5. There are two types of failure: permanent and non-permanent.

6. It is possible to identify a permanent failure reading its indication in the end timestamp, but only from raw files (non pre-processed by the RCM system).

7. The timestamps may contain errors or missings, in some cases not allowing to know when the event ended.

8. The repair of a permanent failure is only done by maintenance and is not visible from the data available; maintenance information is needed.

Figure 3.17: Each traction group has two important events: one severe (orange) and one very severe (red). The distinction of the same code for different circuits is made knowing the associated vehicle (M1 or M2). Severe and very severe TCU events.

The considerations above complemented with expert knowledge, lead to the diagram of states shown in Figure 3.18. These states can be inferred from the data and are later used as the label for the machine learning classification algorithms.
3.4.2 Adding the Label

Having defined the problem and the label on the state of the traction system, it is now possible to add the label to the data. At this point, the dataset is almost complete, only the label is missing. For each block, obtaining the label consists of finding the state of the correspondent traction group. This is done by looking at the columns associated with the important events, from which the states are inferred taking into account the diagram of states presented above. This is done as follows: the label is created as a column of zeros (state OK). Then, the failure states are added. F2 is the first to be considered, which corresponds to the timestamps having a very severe event happening. In order to distinguish the permanent failures, a binary column was also used, indicating if the severe event has or not an end timestamp indicating a permanent failure. Knowing the value in this column, the states F1/PF consist of timestamps having severe events associated with non-permanent/permanent failures, respectively.

The columns containing the important events and also the indication of permanent failure were used as auxiliary information to build the label. For this reason, these are removed after getting the label. At this stage, the dataset and its labels are complete.

3.4.3 Missing Entries

As discussed before, the CCU variables are collected continuously, while the remaining types are registered only when failures occur. This implies that most rows of the dataset contain only the CCU variables and these correspond to a small fraction of the features. Most features are related to the TCU variables of context and their frequency is not comparable. This implies that their columns are missing entries for most rows of the dataset. The case of events is different; they never have missing entries since the values of their columns are always known: if the event is happening at a given timestamp, the value is one, and zero otherwise (meaning that the event is not happening). Therefore, events do not originate missing entries.

\[1\text{We remind that a permanent failure is identified by a special end timestamp of the associated severe event.}\]
A quantitative analysis demonstrates that the number of columns of the blocks shown in Figure 3.16 are, from left to right: 17, 31, 351, 11, 58, 11. So, considering the relative size of the variables of context (85.4%), it is easily understood the reason why most entries of the dataset are missing. Counting the missings of the entire dataset, the conclusion is enlightening: 85.24% are missing entries. Finally, the dataset has 780852 rows and 479 columns.
Chapter 4

Dimensionality Reduction

This Chapter describes the process of reducing the dimensionality of the dataset and the imputation of its missing entries. Firstly, it is presented the background of techniques used to compress data and impute missing data, as well as strategies for model selection and performance evaluation, in Section 4.1. Then, the process of reducing the dimensionality of the dataset is explained in Section 4.2.

4.1 Background

4.1.1 Principal Component Analysis

Datasets often include a considerable number of features, which causes them to have big sizes. Most of the times, it is possible to reduce considerably this high number of features while still preserving much of the information in the original dataset. The principal component analysis (PCA), introduced by Pearson [25], is probably the most widely used technique for dimension reduction of datasets. PCA is also used for other purposes, e.g. outlier detection [26].

This technique analyzes a data table containing observations described by many dependent variables, generally inter-correlated. Its goal is to extract the relevant information from the data table in order to represent it as a set of new orthogonal variables known as principal components. PCA generates an approximation of a numeric data table, in terms of the product of two smaller matrices, which capture the fundamental patterns of the data table [27].

The data table is represented by the matrix $A \in \mathbb{R}^{m \times n}$, where $m$ is the number of observations and $n$ the number of features. In general, before performing the analysis, the matrix $A$ is pre-processed by adjusting its columns so that they have zero mean and unit norm. It is done for each variable by subtracting its mean and diving by its norm. This standardization rescales the data in order to compensate for unequal scaling in different features [28].

PCA finds a matrix $Z \in \mathbb{R}^{m \times n}$ of rank $k < \min(m, n)$ that best approximates $A$ in the least-squares sense. The constraint of the rank can be encoded implicitly be expressing $Z$ in a factored form as $Z = XY$, with $X \in \mathbb{R}^{m \times k}$ and $Y \in \mathbb{R}^{k \times n}$, whose factorization is not unique. The problem is reduced to choosing $X$ and $Y$ in order to minimize $\|A - XY\|_F^2$, where $\| \cdot \|_F$ is the Frobenius norm of a matrix,
i.e., the square root of the sum of the squares of the entries. \( x_i \in \mathbb{R}^{1 \times n} \) and \( y_j \in \mathbb{R}^m \) are defined to be the \( i \)th row of \( X \) and the \( j \)th column of \( Y \), respectively. Therefore, \( x_i y_j \in \mathbb{R} \) denotes the Euclidean inner product. PCA can be written as

\[
\minimize_{X,Y} \| A - XY \|_F^2 = \sum_{i=1}^m \sum_{j=1}^n (A_{ij} - x_i y_j)^2,
\]

with variables \( x_i \) and \( y_j \) [29].

### 4.1.2 Generalized Low Rank Models

PCA can be used for numeric matrices not containing missing entries. When only some entries are observed, this method cannot be used to find a matrix that approximates \( A \). PCA is also not suitable for data tables containing non-numeric features, such as Boolean or categorical data. Generalized low rank models (GLRM) is a method which performs an analogue process of PCA on an arbitrary dataset composed of different data types, e.g. numerical, Boolean, categorical and ordinal, allowing missing entries [30].

Consider that the matrix \( A \) has missing entries so that only the entries \( A_{ij} \) for \( (i,j) \in \Omega \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\} \) are observed. In this case, the problem of finding a low rank matrix to approximate \( A \) can be represented as

\[
\minimize \sum_{(i,j) \in \Omega} (A_{ij} - x_i y_j)^2,
\]

with variables \( x_i \) and \( y_j \). A solution of this problem gives an estimate \( \hat{A}_{uv} = x_u y_v \) for the value of the entries \( (u, v) \notin \Omega \) that were not observed. Considering a matrix where most entries are missing (the case of interest in this work), this method is used to guess all its entries, given just a few of them [29].

Suppose now that the matrix \( A \) is composed of entries drawn from a feature set \( F_j \). The entries of \( A \) can have different data types, e.g. real \( (F_j = \mathbb{R}) \) and Boolean \( (F_j = \{T,F\}) \). Only the entries \( A_{ij} \) for \( (i,j) \in \Omega \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\} \) are observed, such that the other entries are missing. For each data type, a loss function \( L_j : \mathbb{R} \times F_j \to \mathbb{R} \) is given, which describes the approximation error incurred when a feature value \( a \in F_j \) is represented by the number \( u \in \mathbb{R} \). In addition, one must provide regularizers \( r_i(x_i) : \mathbb{R}^{1 \times k} \to \mathbb{R} \) and \( \tilde{r}_j(y_j) : \mathbb{R}^{k \times 1} \to \mathbb{R} \) for \( i = 1, \ldots, n \) and \( j = 1, \ldots, m \), which are used to prevent the overfitting to the observations, encouraging the model to have a particular interpretable form, if needed. The GLRM problem can now be written as

\[
\minimize \sum_{(i,j) \in \Omega} L_{ij}(x_i y_j, A_{ij}) + \sum_{i=1}^m r_i(x_i) + \sum_{j=1}^n \tilde{r}_j(y_j),
\]

with variables \( x_i \) and \( y_j \), and with loss functions \( L_{ij} \) and regularizers \( r_i \) and \( \tilde{r}_j \) [30].

Different loss functions can be used, according to the type of data. In this work, two different loss functions were used: for real-valued data, the quadratic loss, \( L(u,a) = (u-a)^2 \), which is a simple and commonly used loss function; for Boolean data, the Hinge loss, \( L(u,a) = \max(1- au, 0) \), representing the value “true” with +1 and “false” with -1. The Hinge loss is chosen since it punishes misclassifications.
The regularizers used were \( r(x) = \gamma \|x\|_2^2 \) and \( \tilde{r}(y) = \tilde{\gamma} \|y\|_1 \), to prevent overfitting. For the case of \( \tilde{r}(y) \), a sparsity inducing regularizer was also tested, \( \tilde{r}(y) = \tilde{\gamma} \|y\|_1 \), to verify if inducing sparsity on matrix \( Y \) could reduce the problem to only some features while preserving significant information present in the original data. Note that each row of \( X \) represents an example by a vector in \( \mathbb{R}^k \) and the matrix \( Y \) maps these representations back into the original feature space. The columns of \( Y \) embed the features into \( \mathbb{R}^k \).

The solution to the problem \((X \text{ and } Y)\) gives a compressed and real-valued representation that may be used to efficiently store the information present in the original dataset [30].

Finally, this problem can be adapted to allow a similar approach to that commonly used in PCA, which is the standardization of the data. In this technique, instead of performing this on the data, the loss functions are rescaled, preserving the sparsity of the dataset. This approach leads to a final formulation of the GLRM problem [29],

\[
\text{minimize} \quad \sum_{(i,j) \in \Omega} L_{ij} (x_i y_j + \mu_j, A_{ij}) / \sigma_j^2 + \sum_{i=1}^m r_i (x_i) + \sum_{j=1}^n \tilde{r}_j (y_j), \tag{4.4}
\]

with variables \( x_i, y_i \) and \( \mu_j \), where, for each feature \( j \),

\[
\mu_j = \arg\min_{\mu} \sum_{i : (i,j) \in \Omega} L_{ij} (\mu, A_{ij}), \quad \sigma_j^2 = \frac{1}{n_j - 1} \sum_{i : (i,j) \in \Omega} L_{ij} (\mu_j, A_{ij}). \tag{4.5}
\]

The authors of [29] developed a package written in the programming language Julia which can be used to solve this problem. The approach to solve it uses alternating minimization, which iteratively keeps one of \( X, Y \) fixed and optimize over the other, then switch and repeat. The overall problem is non-convex due to bi-linearity, which is why in general it cannot be solved globally. Even though, it can be locally solved by alternating minimization, where each sub-problem is convex and can be solved efficiently. Despite the wide usage of bi-linear representation and alternating minimization, global optimality guarantees for such methods are still lacking [30, 31].

### 4.1.3 Optimization

Most linear models can be learned by solving an optimization problem,

\[
\hat{\theta} = \arg\min_{\theta} J(\theta), \tag{4.6}
\]

where \( \theta \) stands for the model parameters. In most cases, iterative numerical optimization algorithms are applied, from which several different methods can be chosen, depending on the function to be minimized. The goal is to find the global minimum, but the algorithms may not converge to it since they can stop in local minima. If the function is convex, it has no more than one minimum [32].

For the cases where \( \theta \) is a scalar, the derivative of \( J(\theta) \) gives information about the slope of the function. The function decreases when moving a small amount in the opposite direction of the derivative,

\[
\theta^{(t+1)} = \theta^{(t)} - \eta \frac{dJ}{d\theta} \left( \theta^{(t)} \right), \tag{4.7}
\]
where \( \eta \) is the step size which controls the displacement of the point \( \theta^{(t)} \). The process starts with an initial guess, \( \theta^{(0)} \). For the vector case, with \( \theta \in \mathbb{R}^p \) and \( J(\theta) \) a differentiable function in a neighborhood of \( \theta^{(t)} \), \( J(\theta) \) decreases fastest when moving along the opposite direction of the gradient. In this case, it can be written as

\[
\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} J \left( \theta^{(t)} \right).
\] (4.8)

This process is repeated until the function stops decreasing, which means that either it is in the vicinity of a local minimum or in a plateau. This algorithm is known as the gradient descent method [32].

It finds the global minimum of the problem under certain conditions. The objective function has to be convex and smooth, meaning that it is continuously differentiable with Lipschitz continuous gradient. The function has to be below bounded and attain the minimum. In this case, the process stops in a point which is characterized by

\[
\nabla_{\theta} J \left( \theta^{(t)} \right) = 0 \Leftrightarrow \theta^{(t)} = \arg \min_{\theta} J(\theta),
\] (4.9)

meaning that the global minimum was achieved [33].

Choosing the step size involves a trade-off, which is why it is usually obtained by trial-and-error. If it is too small, the update process becomes very slow. On the other hand, a too large step size may cause the algorithm to skip a local minimum or produce an update that increases \( J(\theta) \). A technique used to speed up convergence is the adaptive step size, which increases the step in the next iteration if the objective function decreased; if it increased, the step size is reduced and the previous step is undone, for which is necessary to keep the best solution so far [29, 32].

Generally, the algorithm uses some stopping criteria. For the case of GLRM, the algorithm stops in any of these circumstances: the decrease in the objective is less than a given rate; a maximum number of iterations is reached; the step size decreases below a certain limit [29].

### 4.1.4 Model Selection: k-fold cross-validation

When selecting a model, a way of testing the different values for each parameter is required in order to choose the most adequate ones. One of the most used techniques is the k-fold cross-validation, which is utilized in this work.

In order to choose a model, it is necessary to measure its ability in an independent set and not in the data used for training, a process which is known as generalization. An evaluation done in the training set is not reliable and is too optimistic. A common procedure consists of dividing the dataset into training, validation and test sets. For each model, the algorithm is trained using the training set and the validation set is used for evaluation. After selecting the model with the best performance, the training and validation sets are used together to train the algorithm and the final evaluation is done using the test set, whose data was never used before [32].

This approach of using a validation set requires large volumes of data. When not having a large amount of data, the k-fold cross-validation is a suitable method. The dataset is split into training and test set. Then, for model selection, the training set is split into k folds (subsets of equal size). The number of folds k is chosen to be equal to the number of models to test. For each model, k–1 folds are
used for training and one fold is used for test. The process is repeated k times such that each fold is used once per test set. After this, the model achieving the best performance is selected. Then, the entire training set is used to train the algorithm and the final evaluation is done using the test set, guaranteeing that the final model is tested using data that was never used before [34].

4.2 Dimensionality Reduction and Missing Data Imputation

This Section explains how the tools presented above are applied to the dataset built in Section 3.3. As discussed before, this dataset has two significant characteristics: the high number of features (479) and also the large amount of missing entries (85.24%). These two challenging properties motivate the use of GLRM, which is used to both dimension reduction and missing imputation. GLRM allows to reduce the size of the dataset while preserving as much information as possible, and also imputing the missing entries in order to apply the learning algorithms discussed later. In order to use GLRM, several inputs to build the model need to be considered.

The dataset is composed of two types of data: real-valued and Boolean data. The Boolean data consists of all binary features, including all events and part of the TCU variables of context (flags). These are binary features whose values 1/0 are used as “true”/“false”, respectively. The remaining features are real-valued data. The loss functions and regularizers used are the ones whose choice was explained before in Section 4.1.2. The loss functions are the quadratic loss for real-valued data and Hinge loss for Boolean data. Quadratic regularization is used for matrix \( X \) while two regularizations are tested for \( Y \), quadratic and sparsity-inducing. The true rank \( k \) of the dataset is unknown, which is why the value of \( k \) needs to be chosen.

The inputs to be selected are: the rank \( k \), the regularizer \( \tilde{r}(y) \) and the scales of both regularizers, \( \gamma \) and \( \tilde{\gamma} \). The k-fold cross-validation technique is applied for model selection. Once the model is chosen, the GLRM algorithm is applied to the entire dataset in order to find the matrices \( X \) and \( Y \) to be used later in the next steps of this work.

Ideally, the entire dataset would be used for cross-validation in order to find the necessary inputs to build the final model. However, GLRM has some restrictions in terms of time required for each fit, even using small amounts of data. Since the time for this work is limited, a solution has to be found. It consists of using only a small randomly chosen subset from the entire dataset and then applying the k-fold cross-validation technique to this small portion of the dataset. This has to be done since many models have to be tested, which requires several fitting processes and these are considerable slow.

In addition, another non-ideal solution was used. Since there are several possibilities to combine the parameters to choose, the number of folds would be enormous. The strategy used consist of refining gradually the parameters until finding the best model, by applying k-fold cross validation as many times as necessary. Of course, to a certain degree and not refining indefinitely. Utilizing a huge amount of folds at the beginning would require too much time and the solution would not be so accurate.
The performance metric to evaluate each model is the normalized reconstruction error,

\[
\frac{1}{|\Omega^S|} \sum_{(i,j) \in \Omega^S} L_{ij} (x_i y_j + \mu_j, A_{ij}) / \sigma_j^2,
\]

where \( \Omega^S \) is the set of observed entries in the data used for evaluation. For example, when evaluating a specific model using k-fold cross-validation, the training error is calculated using the set of observed entries in the data used for training; the test error is equivalent but using the observed entries of data for test.

It is also necessary to define the stopping criteria of the algorithm introduced before. For the case of cross-validation, the number of iterations was set to be high enough so that it does not stop (8000) the algorithm; the minimum step size was selected to be very small so that it is almost in the order of machine precision (10\(^{-15}\)); the decrease in the objective function per iteration was set to be small enough to guarantee an adequate approximation (10\(^{-4}\)). Using these criteria, the algorithm usually stops by the criterion of the decrease in the objective function, as desired. Finally, the initial step size was set to be small enough to not compromise achieving the solution but not too small to lead to a very slow fit (10\(^{-2}\)). It is just for starting the algorithm, since it uses an adaptive step size, as explained before. The authors of [29] suggest to use a step size on the order of one, but by doing some small tests, it was found that it is not adequate in some cases, which is why a smaller value is used.

Having defined all the necessary parameters to use the algorithm, the next step is to apply the k-fold cross-validation, which is done in consecutive steps by refining the model parameters, as explained above. It was explained before that the problem is not convex, which implies that the algorithm may not achieve the same solution or similar solutions given the same parameters, but different initializations. Depending on the random initialization, the algorithm can stop in different local minima. This means that it may find very different solutions to the same problem. Considering this, we decided to run the algorithm three times for each model and averaging the results. Three trials are, nevertheless, too few to assure a good approximation, as we experienced.

The first cross-validation tests were done for different values in a wide range for all parameters. These allowed to then focus the attention into smaller ranges. The first tests allowed to conclude that the quadratic regularization performs better than the sparsity-inducing, which is why the latter was discarded. Another conclusion is that scales of \( \gamma \) close to zero and of \( \tilde{\gamma} \) near one achieve the smallest test errors. Regarding the rank \( k \), the training error decreases when increasing the rank, as expected. However, the behavior of the test error is not as expected, that is decreasing at first but then increasing. For some tests, it slightly decreases in the beginning and then increases; in other cases, it is always increasing or oscillating.

This unexpected behavior can be caused by optimization errors or by the randomness of the samples in the test set. Possibly the data used for test in cross-validation is particularly bad in some cases but in other test sets it is especially good. This problem would be mitigated if more data could be used for cross-validation, which is not possible due to time constraints. Finally, the quadratic regularization leads to a well-conditioned problem, which is why it is preferred.
Figures 4.1 and 4.2 present the training and test errors (respectively) for the final refinement using quadratic regularization on both $X$ and $Y$. It shows what was discussed above. The training errors have an expected behavior while the test errors do not. From these results, the final model is selected to have rank $k = 25$ and quadratic regularization with scales $\gamma = 0$ and $\tilde{\gamma} = 2$. This represents a considerable reduction on the number of features when comparing to the original dataset (25 instead of 479 columns), meaning that the size of matrix $X$ is 5.22% of the original dataset size.

The value obtained for the regularization scale on $X$, $\gamma = 0$, implies that the regularization term on $X$ vanishes. We are aware that it can be polarizing our solution. Nevertheless, this value was obtained from cross-validation using the normalized reconstruction error as the metric for evaluation, which is why we consider it as an adequate solution.

After selecting the model, the final step consists of applying the algorithm with these inputs using the entire dataset to get the low rank approximation. Then, the matrix $X$ is used in the place of the original dataset for the classification algorithms in Chapter 5. This representation allows to have a much
more compact dataset without missing entries.

Again, computation time constraints motivated a relaxed stopping criteria. For this reason, the decrease in the objective function per iteration was set to be $10^{-1}$ instead of $10^{-4}$ as used for cross-validation, which causes the low rank approximation to be much less accurate. All the adjustments done were necessary considering the size of the data, the proportion of missing values, and the time available for this thesis.
Chapter 5

Classification and Failure Prediction

This Chapter starts by introducing some background on classification algorithms and strategies to deal with imbalanced datasets in Section 5.1. Secondly, Section 5.2 describes two classification experiments using the original dataset 1. Lastly, Section 5.3 explains the approaches used for failure prediction, from the creation of a new dataset to the results of the prediction algorithms. The classification experiments make use of some tools from the scikit-learn library [35] and also a specific package for imbalanced learn (imbalanced-learn [36]), both developed in Python.

5.1 Background

A common division of machine learning algorithms as to do with the type of learning. In this case, it is a supervised learning problem because the algorithm is being fed with input features together with the correspondent output. From these, the classifier is able to learn a model between the input and the output, which is then used to predict the output of new inputs. Unsupervised (unlabeled data) or semi-supervised learning (partially labeled data) are examples of other types.

The supervised learning algorithms can be divided into classification and regression algorithms. Classification consists of predicting a discrete label while regression predicts a continuous quantity. In this case, the task is to predict a state or a condition (normal/failure). Therefore, it is a classification problem, since the label consists of discrete classes.

5.1.1 Classification Algorithms

The basic idea beyond classification is to find a method to separate the data of two classes, for the binary case. Two different classifiers are used in this work: support vector machines (SVMs) and decision trees, whose introduction below is based on [32].

SVMs are appropriated to deal with not too large datasets, which is the case. They are fast and provide linear models. SVMs are not explainable, but it is not a requirement. Higher accuracy values can be obtained by using kernels, with a complexity trade-off in terms of both storage and processing.

1This name is used to avoid misunderstanding when further ahead creating a new dataset for failure prediction.
Decision trees are fast to train and explainable algorithms while achieving high accuracy values. Even though, decision trees have high variance since they are sensitive to the training data. Small changes in the training set can have a significant impact on the classifier and thus in the prediction quality. This variance can be reduced using adequate techniques, like bagging, which is described in Section 5.1.2.

**Support Vector Machines**

SVMs divide the data by defining a hyperplane in a high-dimensional feature space. Its objective consists of maximizing the separability of the two classes. The hyperplane is expressed in the input space using the support vectors. These are the closest points to the hyperplane and the only of interest. The optimal hyperplane is the one that maximizes the margin between samples from the two classes.

Usually, SVMs are split into three steps. The first is used when the data is linearly separable. The training data points can be expressed as

\[ \{(x_i, y_i), i = 1, ..., N\} \text{ with } x_i \in \mathbb{R}^p, \ y_i \in \{-1, 1\}, \tag{5.1} \]

where \( \mathbb{R}^p \) is the input space. The label of positive samples is \( y_i = 1 \), being \( y_i = -1 \) for the negative training vectors. The hyperplane can be defined as

\[ w^T x_i + b = 0, \tag{5.2} \]

where \( w \) is a normal vector, orthogonal to the hyperplane, and \( b \) is an offset, as shown in Figure 5.1. The optimal margin corresponds to the \( w \) and \( b \) that best divide the classes. This can be expressed as

\[ \begin{align*} w^T x_i + b & \geq +1 \quad \text{for } y_i = +1 \\ w^T x_i + b & \leq -1 \quad \text{for } y_i = -1 \end{align*} \implies y_i (w^T x_i + b) - 1 \geq 0, \ \forall i. \tag{5.3} \]

![Figure 5.1: The instances of the two classes are separated by the hyperplane. SVM hyperplane (extracted from [32]).](image)

This quadratic optimization problem with constraints is solved using the method of Lagrange multipliers. The steps needed consist of building the function and then maximizing it with respect to the
Lagrange multipliers $\alpha_i \geq 0$, $\forall i$, from which the dual formulation is obtained:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j, \quad s.t. \quad \sum_{i=1}^{N} \alpha_i y_i = 0. \quad (5.4)$$

The optimization problem can be written as follows:

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \alpha^T H \alpha, \quad s.t. \quad \alpha_i \geq 0 \quad \forall i, \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \quad (5.5)$$

where $\alpha = [\alpha_1 \ldots \alpha_N]$ and $H_{ij} = y_i x_i^T x_j y_j$. This is a convex quadratic programming (QP) problem that can be solved using standard QP algorithms, returning all multipliers. From these, support vectors, $w$, $b$ are obtained, as well as the classification function $f(x) = \text{sign}(x^T w + b)$.

In general, the datasets of interest are not linearly separable. However, SVMs can be extended to deal with non-separable data. A first approach consists of using a soft margin, allowing data points to be in the wrong side of the hyperplane. A soft margin penalty, $C \sum_{i=1}^{N} \xi_i$, is defined to account for points violating the margin. $C$ is a penalty parameter and $\xi_i$ is a slack variable assigned to each data point. This leads to a trade off between soft and hard margins, depending on the value of $C$. In this case, the Lagrangian function is

$$L_P = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left[ y_i (x_i^T w + b) - 1 + \xi_i \right] - \sum_{i=1}^{N} \mu_i \xi_i, \quad (5.6)$$

where $\alpha_i, \mu_i \geq 0$ are Lagrange multipliers. Finally, the optimization problem can be written as

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \alpha^T H \alpha, \quad s.t. \quad 0 \leq \alpha_i \leq C \quad \forall i, \quad \sum_{i=1}^{N} \alpha_i y_i = 0, \quad (5.7)$$

where $\alpha = [\alpha_1 \ldots \alpha_N]$ and $H_{ij} = y_i x_i^T x_j y_j$. This is again a QP problem whose solution provides all $\alpha_i$, from which the classifier parameters are obtained as before.

A second technique is a non-linear approach. Since the boundary between classes can be not linear, the problem is solved by mapping the data from the input space into a high dimension feature space, where the data can be separated by a hyperplane. The decision function is obtained using the kernel trick. The algorithm only needs to know the inner products between input vectors, which is why it is not necessary to compute the feature vectors but only their inner products. These can be calculated using a kernel function,

$$k(x_i, x_j) = \phi(x_i) \cdot \phi(x_j). \quad (5.8)$$

This non-linear SVM can be used with low dimensional data by replacing the inner products by the kernel. Two common kernels are the linear, equivalent to (5.5) or (5.7),

$$k(x_i, x_j) = x_i^T x_j, \quad (5.9)$$
and the radial basis function (RBF),

\[ k(x_i, x_j) = e^{-\frac{1}{2\sigma^2} \|x_i - x_j\|^2}. \]  

These are the two kernels used in this work.

### Decision Trees

Decision trees divide the training data into smaller subsets so that the label variables in each subset are as homogeneous as possible. The classifier is assumed to be constant in each subset and the class is chosen by majority vote. Usually, decision trees are explained using categorical features. But they can be used with numeric attributes by using thresholds.

The classification is done by traveling along the tree until reaching a leaf node (containing the label). For a given data point, the label is found by going through the tree and making decisions based on the feature values. There are noisy labels both on training and test sets, which consist of examples leading to misclassification because they have the same attributes but different labels. This motivates the use of measures of impurity, e.g. misclassification error or Gini index. All of them achieve a minimum if all data points associated with a leaf node belong to the same class. Gini index is the measure used in this work since it is smoother and usually preferred.

Training a tree consists of finding a tree \( T \) which minimizes the impurity, defined as

\[ I(T) = \sum_{m \in L} p(m)i(m), \]  

where \( p(m) \) is the portion of training patterns associated with node \( m \) and \( L \) is the set of all leaf nodes. \( I(T) \) is an average of the impurity of the leaf nodes.

Ideally, the learning task would consist of finding a tree which minimizes the impurity by generating all possible configurations and choosing the one having the smallest \( I(T) \). While this is not feasible, a suboptimal method is used.

It starts with a root node \( m \) by choosing the best feature for splitting the data. For each attribute, \( m \) is divided and children nodes \( s \in S \) are created. Each child node is associated with a different value of the feature. The impurity of each child is calculated as well as the impurity drop with respect to the impurity of node \( m \),

\[ \Delta I = i(m) - \sum_{s \in S} \frac{p(s)}{p(m)}i(s). \]  

Then, the attribute having the higher drop is chosen. This process is repeated for another leaf node until reaching a stopping criterion, that is until all leaves are pure or all features have been used. Other criteria could be used for early stopping, e.g. only split a node if its impurity is above a defined threshold otherwise, it is a leaf. No early stopping conditions were used in this work.

As referred before, numeric features are used by setting threshold values. These have to be estimated during the growing process, commonly using an exhaustive search. All thresholds are taken into account for each attribute and the best impurity drop is chosen.
5.1.2 Imbalanced Data

It is very common to have a data set whose classes are not equally distributed. Moreover, frequently the most important samples belong the classes having less points. This is exactly what happens in failure prediction, in general. Fraud detection is also a well-known example. The samples representing faults are expected to be much less but more important than the ones associated with a normal condition. It is often better failing to predict the normal examples than the abnormal because the latter are more important to identify. These complex characteristics of the datasets require special tools to enable an efficient knowledge extraction [37].

Regarding performance measures, these problems need also a different approach. As referred above, they require a high detection in the minority class rather than the majority class. For this reason, simple predictive accuracy is not a good measure. The metrics used (e.g. precision and recall) give more importance to the positive class.

Appropriate Metrics for Imbalanced Data

An important concept when defining performance metrics is the confusion matrix, as shown in Figure 5.2, for the binary case. It is a representation of the predicted label compared to the true class.

![Confusion Matrix](image)

Figure 5.2: The confusion matrix represents the combinations of the true and predict labels. Confusion matrix for binary classification.

The most common metrics for imbalanced data are derived from the interpretation of the confusion matrix and two examples are precision and recall. For the binary case, these are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (5.13)$$

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (5.14)$$

A similar scheme is used for multiclass cases. For a $N$ class problem, the confusion matrix, $CM$, is a $NxN$ matrix. Precision and recall are then computed as follows, for each class of interest $i$:

$$\text{Precision (class } i) = \frac{CM_{ii}}{\sum_j CM_{ij}}. \quad (5.15)$$
Recall (class $i$) = \[ \frac{CM_{ii}}{\sum_j CM_{ij}}. \] (5.16)

Precision can be interpreted as measuring how many samples predicted as class $i$ are truly of class $i$. On the other hand, recall measures how many true points of class $i$ are labeled as class $i$. Using an example from the context of failure prediction, precision measures what proportion of predicted failures was actually correct. Recall measures what proportion of real failures was identified correctly. Ideally, both metrics should be close to one. For each particular problem, one of these metrics can be considered as the more relevant. Some other metrics are also used for imbalanced data, e.g. F1-score, specificity and sensitivity. For this work, only precision and recall are considered.

Other tools are frequently used for classifier performance evaluation. A generic method, not specifically designed for imbalance data, uses the receiver operating characteristic (ROC) curve. It represents the classifier performance over combinations of true positive and false positive error rates. The true positive rate (TPR) is the same as the recall, and the false positive rate (FPR) is defined as:

\[ \text{FPR} = \frac{FP}{FP + TN}. \] (5.17)

The objective of the ROC curves is to be in the upper left hand of the ROC space. A traditional metric for the ROC curves is the area under the curve (AUC), that should be as close to one as possible. The AUC provides an aggregate measure of performance across all thresholds. It can be seen as the probability that the model ranks a random positive data point more highly than a random negative example.

The curve is obtained from the scores extracted from the classifier. For each instance of the test set, a score can be obtained. By varying the threshold to classify the instances as positive or negative, different confusion matrices are created, from which the points on the ROC space are generated. These points are then interpolated. For example, the score used for the SVM algorithm consists of the distance from each data point to the hyperplane. For the case of decision trees, the score used is the predicted class probability, that is the fraction of samples of the same class in a leaf. When using the bagging technique, as discussed later, the score is calculated as the mean predicted class probabilities of the decision trees in the ensemble [35].

This is a general method and it is not the most adequate for imbalance data, as the ROC curve may provide a too optimistic view of the algorithm performance. For this case, an alternative is commonly used, the precision-recall (PR) curve, which provides a more informative representation of the performance. It is obtained similarly to what is done for the ROC curve, but computing precision and recall. While the goal of the ROC curves is to be in the upper left of the ROC space, a dominant PR curve resides in the upper right of the PR space [38]. Contrary to what happens with the ROC curve, the AUC of the PR curve does not make a relevant contribution. Generally, the comparison can be made by considering that the curves closer to the upper right corner have a better performance.
Strategies for Imbalanced Data

Many techniques were developed to mitigate the problems with imbalanced datasets. There are various approaches, from which four different strategies were chosen to be used in this work. These are SVMs with weighted classes, undersampling, SMOTE and bagging trees. The first two methods are classic and simple strategies, which is why they are tested. Due to the high imbalance, two more advanced and complex techniques are also used.

SVMs with weighted classes

This simple strategy consists of an adjustment done to the soft margin of the SVM algorithm. The parameter $C$ used in the soft margin penalty is adjusted for each class, in a way that the weights become inversely proportional to the class frequencies. The value of $C$ is multiplied by the factor $\frac{N}{n \cdot N_i}$, where $N$ is the number of samples (of the entire dataset), $n$ the number of classes and $N_i$ the number of instances belonging to class $i$. This intends to increase the penalty for minority class misclassification to prevent the minority class samples from being overwhelmed by the majority class.

Undersampling

The sampling methods consist of the modification of an imbalanced dataset to get a balanced distribution. In general, a balanced dataset achieves better performances compared to imbalanced datasets, which justifies the use of sampling methods. For most imbalanced datasets, the application of sampling techniques improves the classifier performance [37].

The first sampling technique used in this work consists of undersampling the majority class by randomly removing samples from it until reaching the desired class size. The percentage of undersampling can be adjusted, for which cross-validation is used. This method is simpler to apply when having a two-class problem. In this case, the majority class can be reduced to have a size similar to the minority class. For the multiclass problems, it is not so simple, since it is possible to have different majority and minority classes with unequal sizes. This will be discussed later in Section 5.2.

SMOTE

The authors of [39] proposed a combination of undersampling the majority class and oversampling the minority class, called synthetic minority oversampling technique (SMOTE). The oversampling of the minority class is done creating synthetic data points. These are introduced along the line segments joining the any or all the $k$ minority class nearest neighbors. According to the amount of oversampling needed, the necessary neighbors from these $k$ neighbors are randomly chosen.

In order to generate the synthetic instances, the difference between the sample and its neighbor is used. This difference is then multiplied by a random number between zero and one and is added to the sample under consideration. The new data point is created along the line segment between two examples. Using this method, the decision region of the minority class becomes more general. The undersampling of the majority class is done by randomly removing samples from it until achieving the desired class size.

Although SMOTE can be used with multiclass problems, its authors were focused only on two-class problems. Finally, the selection of the parameters (undersampling and oversampling rates, and the number of neighbors) can be done using cross-validation.
Balanced Bagging

Undersampling the majority class is a good strategy but has a problem which is its high variance. This leads to different performances of classifiers over individual bootstrap examples. The solution proposed in [40] is to use the bagging (bootstrap aggregating) variance-reduction ensemble method.

The traditional methods like SMOTE have a common problem: the positive distribution (minority class) is under-represented compared to the negative distribution (majority class) and this implies a poorly characterized positive distribution. It leads to a separator that is skewed toward the minority class, being closer to the minority class points than it should be. This results in poor predictive performance over hold-out minority class instances. The problem of those methods is that generating more points inside the same region does not decrease the bias. Undersampling is effective despite its simplicity because it reduces the probability of the separator to be biased. Nevertheless, it is also a high-variance process because distinct re-samplings induce different separators, as shown in Figure 5.3. This is the problem that the bagging technique tries to minimize.

The balanced bagging method can be summarized as follows: an ensemble composed of a given number of models is built, each induced over an undersampled bootstrapped sample of the training data. The bootstrap samples are generated by sampling randomly with replacement and then the undersampling of the majority class is done removing instances randomly until the classes are balanced. When a new instance is to be classified, each model makes a prediction, and the final prediction is obtained as the majority vote. In general, this approach will improve classifier performance when the models of the ensemble are high-variance, which is the case.

Figure 5.3: $\hat{w}$ is the biased separator induced using the full dataset. $w^*$ is the optimal plane (w.r.t. the distributions). The grey lines represent the separators induced over independently drawn balanced bootstrap samples using the training data. All of these are less biased than $\hat{w}$ and closer to $w^*$. The effect of undersampling on the separator (extracted from [40]).

This method is usually applied with decision trees as the base classifiers, which will also be the case in this thesis. Finally, cross-validation can be used to select the parameters needed: the number of models and the size of the bootstrap samples. The authors of [40] chose the number of models arbitrarily to be 11.
5.1.3 Binary and Multiclass Classification

The classification problems are commonly divided into two categories: binary or multiclass. Several tasks can be labeled using only two labels. In other cases, three or more classes are needed. One example could be predicting the state of the traction group based on the diagram of states (four possible states) presented before.

Binary problems are easier to solve, in general. Multiclass labels bring some complications, namely on how to use the algorithms designed for binary problems, which is the case of SVMs. On the contrary, decision trees can naturally handle multiclass problems.

Multiclass Techniques

In order to use the SVM algorithms (among others) in multiclass problems, a common strategy consists of dividing it into binary problems, and then using the classifiers as in the binary case. Two frequent approaches are one-vs-all and one-vs-one.

The first technique trains one classifier per class, that is \( N \) classifiers, using the points of that class as positives and all other points as negative samples. The classification is made by picking up the class for which the corresponding classifier has the highest score.

The second method trains one classifier per each pair of classes, for what \( N(N-1)/2 \) classifiers are necessary. The prediction is based on majority vote and therefore the class with more votes is chosen, considering all classifiers. The one-vs-one approach is the one used in this work since it is generally much less sensitive to the problems of imbalanced datasets when compared to one-vs-all. Nevertheless, it is much more computationally expensive [41].

5.2 Classification Experiments

This Section describes two classification experiments which allowed to test the methods previously introduced using the original data set. Firstly, a multiclass approach is tried, in which the label corresponds to the states of the diagram presented in Section 3.4.1, for each traction group. A second method is also tested, which consists of using binary classification by converting the label to OK/fault, that is aggregating all failure states.

5.2.1 Multiclass Experiment

The data set is highly imbalanced, as shown in the label histogram of Figure 5.4. Class ‘0’ (state OK) represents 99.89% of the size, while classes ‘1’ (state F1), ‘2’ (state F2) and ‘3’ (state PF) contribute with 0.0159%, 0.0018% and 0.0904%, respectively. For this reason, the methods to deal with imbalance introduced before are applied.

SVMs without any method for imbalance were also tested, just to validate the considerations done before on imbalanced data and its consequences. The results using a linear SVM achieved a very high accuracy (0.996) but poor values for the most important parameters, precision and recall, which are zero.
except for class ‘3’ (0.89 and 0.23, respectively). It verifies the problems discussed before. The classifier labels almost everything as ‘0’, ending up with a great but fruitless accuracy value. The results for the class ‘3’ can be explained considering its size, which is small compared to the majority class but still much larger than the other minority classes. As said above, these results intend only to show the effects of the enormous imbalance.

![Label Histogram](image)

Figure 5.4: The majority class is much bigger than the others; also minority classes are imbalanced. Class ‘2’ has a few instances. Original label histogram (logarithmic scale).

Regarding the methods for imbalanced data, the undersampling technique was not used for this multiclass case. This is a multiclass problem with imbalance between all classes, which is why it is hard to define a rate for undersampling the majority class since the problem would still be imbalanced. The weighted SVM is not affected by this particular problem. SMOTE does undersampling of the majority class but can generate instances for minority classes, balancing the dataset. SMOTE is applied in a way that the undersampling rate is defined and then instances for the minority classes are created until balancing all classes. Finally, bagging is affected when balancing the classes in each bootstrap sample, since most instances are removed. Even though, building several models can slightly reduce this effect. Nevertheless, it is expected that this affects the results, and in fact, it does.

Figure 5.5 presents the final results when testing the methods for imbalanced data. The results show that none of the methods achieves adequate performance values of precision and recall since both metrics have poor values for most minority classes, which makes the classifiers incapable of making reasonable predictions.

As expected, the huge imbalance severely conditioned the results. There are two challenging problems: firstly, the majority class is much bigger than all the minority classes together. Secondly, there is also a big imbalance between minority classes. This combination makes the problem very hard to solve. Considering this, it is desirable to find an adequate alternative. In fact, it is possible to transform this into a binary classification problem, aggregating the classes ‘1’, ‘2’ and ‘3’ as failure and class ‘0’ as OK. The notion of a label as a representation of the diagram of states is lost, but an important characteristic is preserved: the distinction between a failure and the normal condition (OK).
5.2.2 Binary Experiment

Similarly to what is presented above, this Section shows the results for the same tests but using the binary classification (OK/fault) approach. Even using two classes, the problem still highly imbalanced: the minority class (failure) represents only 0.1081% of the data.

As done before, the dataset was tested without imbalanced methods and the conclusion is the same: those methods are needed. Using a linear SVM, the accuracy achieves 0.996 and precision and recall are 0.89 and 0.22, respectively. These are not so poor as for the multiclass experiment, but imbalanced methods are expected to achieve better performances.

This time, the problems with imbalance between various classes disappeared, since there are only two classes. Therefore, all the methods for imbalanced data can be applied without the complications observed in the multiclass approach. Figure 5.6 presents the final results when testing these methods. SMOTE achieves the better results for both precision and recall. The remaining methods have recall values similar to SMOTE but lower precision. As expected, all these methods achieved higher values of recall than the simple SVM without any method for imbalance. Nevertheless, none of them hit its precision value (0.89). In this case, there are two alternatives to consider, depending on whether precision or recall would be considered as the more adequate metric. Ideally, both values would be close to one.
These results show that this binary approach is more appropriate to the problem. It achieves better performances and continues to preserve the most important classification characteristic, which is distinguishing a failure state from a normal condition. Desirably, it would differentiate every state but it is not a crucial requirement, which is why this approach can be considered as adequate taking into account all the limitations.

Figure 5.6: SMOTE performs better than the remaining methods for both metrics. All approaches achieve suitable values of recall but low precision. Binary classification results with imbalance methods.

5.3 Failure Prediction

The experiments which were done before tried just to identify data points as corresponding or not to a failure or normal condition. This Section addresses the final goal of this work, which is failure prediction. This challenge requires a new data set, that is built from the original data set. The necessary steps are described below.

The scheme for failure prediction is shown in Figure 5.7, representing how to extract the data from the original dataset. The green block represents the data points which are used to predict if there will be a failure afterwards. The label of the upcoming data is the blue block. The orange square is the label obtained for prediction, which is one if there is any failure on the blue block and zero otherwise. The data points from the green block are then vectorized to be a row on the new dataset for prediction whose corresponding label is the value of the orange block. As Figure 5.8 shows, these steps are repeated by sliding on time, an operation that is represented by the dashed blocks. A new row results from these dashed blocks. The dataset for prediction is built moving forward on the original dataset until reaching the end, that is when the blue block reaches the last data point. Note that this operation is done separately for each traction group, using the correspondent data.

As pointed out in Figure 5.8, there are some parameters to be defined. Their meanings and values are described in Table 5.1. The ideal time window for prediction would be around one month, which is more or less the interval between maintenance operations. However, the available data covers only a few months, which is why it is set to be one week. The horizon to be predicted is chosen to be equal to the time window for prediction. Then, the sliding window step used is one day, allowing to have more weeks
to train the algorithm. Finally, no delay is used, meaning that the time windows are consecutive: data from one week is used to predict if there is a fault in the next week.

Exemplifying, this means that using the data collected in the past week, the algorithm predicts if there will be a failure in the next week. For synchronism reasons (discussed below), these blocks are defined to begin always at midnight, so this tool can be used whenever a new day begins. This start time is chosen to coincide with the beginning of the day and in a way that it is a time of the day generally associated with a low intensity operation of the train.

As expected, the weeks do not have the same number of data points. The size of each week varies significantly, from 6,000 to 18,000 data points, approximately. When building the dataset for prediction, the ideal is to have synchronous data points for every week. This makes the features of the prediction dataset corresponding to equivalent times of the day, in order to preserve the time characteristics. However, it does not happen having weeks with so different sizes. These big differences are explained by the
rail operation itself. There are some weeks in which the train is some days stopped for maintenance or
days being less used. The fact of choosing one week for the time window is an attempt to minimize these
synchronization problems. As expected, e.g. during weekends, sometimes the train is generally more
time stopped, and this time window allows to capture more regularity.

Even using these small tricks, the regularity is far from the desired. Figure 5.9 represents the data
points for some weeks over time. There are big gaps inside several weeks, and these are propagated along
some days because of the sliding window of one day.

![Figure 5.9: Several weeks have big gaps without data points; these gaps are propagated because of the
sliding window of one day. Even the other weeks have many irregular gaps. Distribution of data from
weeks of the traction group M1,G1.](image)

### 5.3.1 Data Synchronization: Oversampling Method

A solution was designed to cope with this problem, consisting of an oversampling procedure. The week
is divided into small slots, being the size of the slot a parameter to define. It can be one minute, which is
close to the cadence of data points or a bit larger, e.g. five minutes, to avoid ending up with a huge data
set, as explained below. For each slot, the number of points in each week is counted. Then, new data
points are inserted in the weeks not having the maximum number of points until reaching the maximum
number of points. These data points are inserted to be in that slot and filled with missings. Doing these
steps for all slots of the week, in the end all weeks have the same number of points in each slot, assuring
synchronism. And every week has the same number of data points, which meets another requirement,
having vectors with equal sizes when performing vectorization. The resulting statistics of this solution
are shown in Table 5.2.

Table 5.2: The oversampling method sets the size of every week to be the same. Example using a slot of 5 min. to avoid a huge dataset but preserving synchronism. Oversampling method statistics (example).

<table>
<thead>
<tr>
<th>Slot Duration</th>
<th>Week Size</th>
<th>Original Dataset Size</th>
<th>New Dataset Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 min.</td>
<td>24493 data points</td>
<td>780852 x 479</td>
<td>392 x 11732147</td>
</tr>
</tbody>
</table>

The next steps would be common to those followed before in Chapter 4 (dimensionality reduction) but this time using the new data set. Firstly, tuning the parameters and choosing regularizers using cross-validation and then applying it to the entire data set. Although this solution would overcome the problems discussed above, it is not feasible, at least within the time of this work. As discussed in Chapter 4, GLRM has some limitations, namely on the time needed. If the time spent on the original dataset was very considerable, in this case it would require much more.

An alternative could be using the product of GLRM for the original data set, that is using the data points coming from the dimensionality reduction instead of applying GLRM from the beginning again. This is not the desired solution but would be more feasible. Even so, the problem of imputing the missings added when synchronizing weeks remains and needs to be solved. This requires using GLRM to fit this new and much bigger data set. Again, time limitations make this impossible to be done within the available time for this thesis.

5.3.2 Data Synchronization: Undersampling Method

An alternative has to be designed to work around all the limitations. The solution found and presented below has several unavoidable drawbacks, which are consequences of everything that was discussed above.

The limitations related to GLRM imply that no solution involving oversampling can be used, which would be the desired way to deal with the problem of synchronization. The solution found to cope with this problem is undersampling. It represents a radical change from the previous approach, but it is needed taking into account the restrictions.

The week is again divided into slots, whose size can be selected. This time, the operation consists of removing data points. For each slot, it is calculated the minimum number of points in all weeks. Then, for the weeks to have the same number of data points, it is necessary to remove points in all weeks until all of them having the same number of points. The minimum can be zero, implying that all points are removed from that slot.

In order to minimize the damage of removing data, the algorithm chooses the points using a semi-random criteria. Firstly, it removes randomly points originally coming from the CCU data. Then, if more points need to be removed, it randomly removes points originally coming from the TCU data. This allows to keep as much TCU data as possible, as it is the more relevant one (traction specific data).

As said above, it is possible to eliminate all points from a slot. Figure 5.9 shows the distribution of data from where several big holes are visible. Even using big slots (order of hours), at least one week would not have any point in each slot. Again, a solution needs to be designed to overcome the obstacle. It can be done into two parts. Firstly, the weeks containing big holes are removed and not used. For
this, a minimum number of data points is used as the criterion. Most weeks having big holes coincide with longer maintenance periods. Also, when sliding through the data, a few days affect several weeks because of the sliding window of one day, as explained before.

The second part of the solution consists of choosing a longer slot duration. Removing the weeks containing big holes doesn’t solve the problem by itself since one week not having data in a slot is enough to force all the data in that slot to be removed. Therefore, the solution consists of using a slot of two hours, which is much bigger than desired but, considering the context, it is not so long that the synchronization is severely affected. Two hours represent only 1.2% of a week. Figure 5.10 shows the distribution of data after applying this undersampling approach.

![Distribution of data using the undersampling method.](image)

Figure 5.10: The undersampling approach allows to have weeks with the same number of data points. The distribution in time becomes generally uniform. Distribution of data using the undersampling method.

Table 5.3 presents the statistics after applying this undersampling solution. All weeks having less than 15,500 data points were removed before applying undersampling, to avoid the big gaps. After performing undersampling, the week size is much smaller than the mean week size before removing data. It represents a reduction of more than 50%. The fact that so much data is gone has certainly a significant impact in the final results.

<table>
<thead>
<tr>
<th>Slot Duration</th>
<th>Week Size after Undersampling</th>
<th>Original Dataset Size</th>
<th>Prediction Dataset Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 hours</td>
<td>6512</td>
<td>780852 x 479</td>
<td>96 x 162800</td>
</tr>
</tbody>
</table>

Table 5.3: The undersampling method sets the size of every week to be the same, but implying a reduction of more than 50% of the data. Undersampling method statistics.

### 5.3.3 Undersampling Method: Classification

At this point, the dataset obtained by applying undersampling is ready for classification. Firstly, it is important to verify if it is balanced or not. Figure 5.11 shows a histogram of the label. In this case, the dataset is more or less balanced (57%/43%). For this reason, it is expected that SVMs without
imbalanced methods can deal with this imbalance.

![Histogram of the prediction dataset label.](image)

**Figure 5.11:** The prediction dataset is more or less balanced. Histogram of the prediction dataset label.

The division of this dataset into training and test was not done by randomly dividing it into training and test set as done before for the original data set. The approach employed was using the most recent weeks of each traction group as the test set. This has two purposes: firstly, not using weeks for training and test containing data points in common, which could compromise the confidence of the results. This happens because of the sliding window, as explained before, which makes the same data to be in several weeks. Even using the most recent weeks as the test set, some weeks have to be removed to prevent the test set to contain data in common to the training data, again because of the sliding window. Secondly, this way simulates what would be the real application of the algorithm: receiving unseen data from the train and trying to make a prediction for the future based on historical data.

SVMs were tested without imbalance methods, for which a precision and recall of 0.33 and 0.40 were obtained using a linear SVM, respectively. These values are very far from the desired but are somehow expected. All the considerations about the approach which had to be taken were presaging this outcome. Nevertheless, even expecting these results, it makes sense to try one last approach before further analyzing the results. Even though the prediction dataset is more or less balanced, the imbalanced methods used before were tested.

Figure 5.12 presents the results of the imbalance methods. These results represent an improvement when compared to the previous ones not using strategies for imbalanced data. The accuracy seems to not be significantly affected, but the same doesn’t apply for precision and recall, which can be considered as the most important parameters. The bagging technique achieves the highest values for both precision and recall, and also accuracy (0.70). The precision still low but the recall is the maximum. Although bagging cannot be considered as an optimal solution due to its low precision, its recall value is very important. It means that all weeks having failures are correctly labeled, which is a relevant feature of this prediction algorithm. If recall would be considered as the most important metric, this method could be a viable solution.

Another study was made for the classifiers presented above, consisting of a further evaluation using two other performance metrics. Figure 5.13 shows both the ROC and PR curves for the imbalance methods presented above. These curves confirm bagging as the best solution among the strategies tested, achieving the highest AUC in the ROC curve and being the closest curve to the upper right corner in the PR curve.
Figure 5.12: Bagging outperforms the remaining techniques for both metrics, achieving a maximum recall, even so with a low precision. Classification results using methods for imbalance.

(a) Bagging has the highest AUC (0.83), outperforming the remaining methods: weighted SVM (0.57), undersampling (0.53) and SMOTE (0.57). ROC curves.

(b) The bagging curve is the closest to the upper right corner, outperforming the other strategies. PR curves.

Figure 5.13: Both ROC and PR curves verify bagging as the best solution. ROC and PR curves for imbalanced methods (markers representing the observed values).

Finally, the confusion matrix for the best solution is presented in Figure 5.14. It shows what was discussed before. The maximum recall allows to not have false negatives (all weeks with failures are correctly predicted), while the low precision induces some false positives (some weeks without failures are incorrectly predicted).

Figure 5.14: All weeks with failures are correctly predicted (FN=0) but some weeks without failures are incorrectly predicted (FP≠0). Bagging confusion matrix.

The tests which were done using these imbalance techniques showed improvements over the first results. Ideally, the results would have high values of accuracy, precision and recall, which is not the case. Nevertheless, some values achieved can be considered as being positives, in particular, the value of recall,
as explained above. On the other hand, precision still low.

One must conclude that the results are far from the expected for this type of application. Even so, the analysis of the results must take into account all the limitations. It is important to examine those limitations and their causes:

- It is not known how noisy the data is. As happened with the discarded temperatures, other sources of data can be introducing relevant noise which was not detected due to the huge amount of variables and, in most cases, their unknown meaning.

- The fact that the missing imputation was done using linear models is also a source of noise, specially considering that most entries are missing and therefore imputed.

- The steps followed are not taking into account temporal correlations, which would be a relevant improvement by taking advantage of that information in the data.

- The fitting done by GLRM is not as accurate as needed. The stopping criteria of the algorithm had to be relaxed in order to have results on time, conditioning significantly the dataset used.

- Again because of time constraints, the approach of oversampling was discarded. The undersampling method had severe repercussions, removing more than 50% of the data.

It is impossible to know what would be the results if the limitations above were mitigated. Of course, one would expect better results. Stays open the possibility of this work to be continued, allowing these ideas to be tested and evaluated, which was not achievable within the time for this work.

5.3.4 Results from an Operation’s Perspective

The results presented above can be discussed from a different perspective, closer to the expert knowledge point of view. It is important to consider the relative cost of repairs and maintenance activities, for both the baseline and the proposed solutions.

The analysis is made assuming some simplifications. Firstly, the baseline solution is considered as just planned maintenance. This means that the failures will correspond to repairs. Secondly, the proposed solution assumes that the all predicted failures lead to maintenance activities, avoiding repairs for the case of correct predictions. Finally, there are periodic maintenance activities for both solutions.

Considering these assumptions, one can calculate the number of repairs ($\#REP$) and maintenance activities ($\#MAN$) considering the confusion matrix. For the baseline solution, $\#REP = FN + TP$ and $\#MAN = PER$, where $PER$ is the number of periodic maintenance activities. For the proposed solution, $\#REP = FN$ and $\#MAN = PER + TP + FP$.

The total cost for each case is the calculated using $\#REP$ and $\#MAN$, and the relative cost between repairs and maintenance activities. Consider that $C_{REP}$ and $C_{MAN}$ represent the costs of repairs and maintenance activities, respectively. The relative cost between them can be expressed as $C_{REP} = K \cdot C_{MAN}$. Therefore the total cost for each solution is $\#REP \cdot C_{REP} + \#MAN \cdot C_{MAN}$.
Finally, the total cost for the baseline solution is

\[ C_{\text{BASELINE}} = (FN + TP) \cdot C_{\text{REP}} + \text{PER} \cdot C_{\text{MAN}} = ((FN + TP) \cdot K + \text{PER}) \cdot C_{\text{MAN}}, \quad (5.18) \]

and, for the proposed solution (designated as SOLIST),

\[ C_{\text{SOLIST}} = FN \cdot C_{\text{REP}} + (\text{PER} + TP + FP) \cdot C_{\text{MAN}} = (FN \cdot K + \text{PER} + TP + FP) \cdot C_{\text{MAN}}. \quad (5.19) \]

The costs for both solutions can be compared using the equations above. From these equations, it is possible to find a relative cost \( K \) for which the costs for the proposed solution are lower than the baseline, by solving \( C_{\text{SOLIST}} < C_{\text{BASELINE}} \). The solution for this inequality is

\[ C_{\text{SOLIST}} < C_{\text{BASELINE}} \iff \frac{TP + FP}{TP} \iff K > \frac{1}{\text{Precision}}. \quad (5.20) \]

Note that the solution does not depend on the number of periodic maintenance activities \( \text{PER} \). Considering the value obtained for precision (0.45), the proposed solution is cheaper than the baseline if \( K > 2.22 \), that is, this solution represents a cost reduction if a repair costs at least 2.22 times more than a maintenance activity. From domain knowledge, we know an estimate for the relative cost: \( 3.7 < K < 6.1 \). It is clearly above the relative cost for which our solution is cheaper than the baseline. Therefore, considering the results on the test set, our solution represents a cost reduction when compared to the baseline.

Figure 5.15 shows an example of the comparison between these two solutions, for hypotheticals \( K = 4 \) and \( \text{PER} = 2 \). The markers represent the two solutions. These points are obtained from the test set using the equations above. The lines represent constant costs (‘isocost’ lines) and the cost increases when moving the line upwards. The blue line is associated with the baseline solution while the green corresponds to the proposed solution (SOLIST). The orange line points out that the cost of one repair is equal to the cost of four maintenance activities.

![Figure 5.15: This example for \( K = 4 \) and \( \text{PER} = 2 \) shows that the proposed solution is cheaper than the baseline, as the ‘isocost’ line for SOLIST is below when compared to the baseline. The objective is to be as much as possible below the blue line. The markers represent the baseline and SOLIST solutions obtained from the test set. Cost comparison of baseline and proposed solutions.](image-url)
This means that, in terms of costs, the most relevant metric is the precision, which is lower than desired. Nevertheless, a deeper analysis cannot consider only these costs. As discussed before, the value of recall is very important for different reasons, for example the fact that the maximum recall obtained allows to avoid breakdowns during operation, whose financial losses go far beyond the cost of a repair, as simplistically assumed here.
Chapter 6

Discussion and Perspectives

This thesis addressed the problem of failure prediction on railways traction systems, for which a data-driven approach was used to design a solution, making use of data science and machine learning tools. This work resulted from the desire of finding an innovative solution for a real engineering problem. For this reason, it involved a challenging process of interpretation. It was necessary to design an adequate representation from a data science point of view but not forgetting all the expert knowledge considerations.

Naturally, this kind of processes with real data give rise to several difficulties. The first challenge had to do with the fact that the data started to be collected during the time of this work due to some unexpected delays, which caused it to be gradually available. Secondly, the analysis of the data was complicated because of the numerous and very different types of data which were used. For this reason, the EDA was very important. It allowed to discard irrelevant variables and, at the same time, previously unknown problems were detected, e.g. the overflows in the first days or the errors with the temperatures, with the latter motivating a change of direction in the approach to the problem. Even so, working with real data has also a positive side, allowing to face many different challenges emerging throughout the process of exploring the data, which would not happen using synthetic or processed data.

The process of joining all the types of data into a single dataset end up with a considerable number of features (although many constant variables were removed) and a huge percentage of missing entries. These characteristics motivate the use of GLRM, which generalizes PCA, allowing different types of data and missing entries. It was used both for dimensionality reduction and missing imputation. Despite being a powerful tool to perform these steps, GLRM has a computational cost, growing linearly with the size of data, that is relevant on our dataset. Our time constraints, derived from real data acquisition, EDA, and problem formulation, stringent to a one semester thesis, motivated some trade-offs to have results in a reasonable time in order to proceed to the next phases of the work. In particular, it was necessary to use a reduced portion of data when performing cross-validation for model selection and then, when fitting the entire dataset, the stopping criteria of the algorithm had to be considerably relaxed, implying a much less accurate approximation. Nevertheless, GLRM allowed to find a low rank approximation with a much lower number of features (25 instead of 479) and, at the same time, performing the imputation of the huge number of missing entries (85.24%).
The classification experiments using the original dataset demonstrated the consequences of having a highly imbalanced dataset, not only between the majority and minority classes, but also between minority classes. These tests evidenced the improvements when using strategies for imbalanced data. Reducing the problem to the binary case allowed to achieve better performances but still far from desired, since either we achieved a low precision and an adequate recall or the opposite. However, these experiments were the analysis process of the data, and not the main goal of this thesis, which is failure prediction.

When designing the solution for failure prediction, an oversampling approach was considered, allowing to maintain the synchronism of the data. For this, a new dataset was created from the original dataset. Even so, this strategy could not be adopted, again because of dimensionality reduction and data imputation complexity constraints. Ideally, the same steps would be followed for dimensionality reduction of this new dataset. However, it would not be practicable due to its size, considerably larger than the original dataset. Even a simpler but not so accurate strategy had to be abandoned. It consisted of making use of the low rank approximation of the original dataset, instead of repeating the entire process of dimension reduction. Nevertheless, it would be necessary to use GLRM for missing imputation as new missing entries were generated by oversampling.

Since no solutions for failure prediction using GLRM could be utilized, an alternative approach consisting of undersampling was used, in which the new dataset was built from the approximation of the original dataset. Many consequences resulted from this solution. Firstly, several weeks were not used due to the irregularity of the data in time, with some weeks having much less data than the others. Even discarding those weeks, more than 50% of the data was removed when synchronizing the data, despite using a relaxed synchronism criterion, without which all data would be easily removed, making the solution unrealistic.

Even though the dataset used for failure prediction was relatively balanced, the use of methods for imbalanced data considerably improved the results. The precision and recall for the best-performing technique (balanced bagging) achieved 45% and 100%, respectively. This represents a lower precision than desired but a maximum value for recall, which is a very important feature for a failure prediction algorithm. The optimal recall means that all weeks having failures are correctly predicted. This guarantees that all failures are anticipated, allowing to repair the system before its breakdown. Moreover, it avoids the serious consequences of a failure occurring during operation, which can force the train to stop, causing high financial losses and comprising all the operation schedule, which is even worse since it is a passenger train. On the other hand, the precision value below what one might expect implies that the algorithm incorrectly labels some weeks as having failures. This leads to unnecessary maintenance operations, negatively affecting the train availability. Nevertheless, these negative consequences are expected to have a lower impact than not predicting a true failure.

Considering a simplistic comparison between the proposed solution and a planned maintenance strategy, our solution represents a cost reduction if a repair costs at least 2.22 times more than a maintenance activity. From domain knowledge, we know an estimate for this relative cost, which is between 3.7 and 6.1. Therefore, considering the results on the test set, the proposed solution represents a cost reduction when compared to the planned maintenance strategy.
Despite having achieved significant results, one would expect to obtain better results if the preferred alternatives could be used. All the code developed is prepared to perform those alternatives, but the time restrictions did not allow to go ahead with those approaches. Having more time would certainly contribute to attain a better solution, but the data itself plays also an important role. Firstly, we believe that using more data would improve the model. This work was restricted to the data available by that time. In addition, the quality of the data is also a crucial aspect as noisy data can severely influence the results. The errors which were detected with the temperatures may indicate the existence of other problems which cannot be discovered due to the huge number of variables and its unknown meaning in most cases. One must conclude that important results were achieved, especially considering all the limitations.

The results obtained in this work are in accordance with the recent studies which consider data science and machine learning as powerful tools to optimize maintenance strategies. This demonstrates the potential benefits of this kind of predictive maintenance approaches. The near future in this area will certainly make use of these tools. The data-driven models do not need the deep study of failure modes, which makes the models too specific and difficult to obtain, especially in complex systems. In addition, they are able to learn intricate patterns and behaviors which probably would not be accessed using traditional methodologies. Even if they are sometimes less interpretable, they are generally more capable of predicting failures than the classic methods. Finally, most of this work is not specific for railways, as discussed before. These tools can be applied in a wide range of domains and applications.

6.1 Future Work

Different limitations have arisen throughout this work, motivating the use of undesired alternative solutions. These challenges should be considered in future work endeavors.

As mentioned above, some steps were designed but could not be done because of time constraints. For this reason, the model selection should be repeated using more data and then the final approximation should be done using that model in order to find a better approximation, this time without relaxing the stopping criteria of the GLRM algorithm. The same is valid for the oversampling method, which can be tested with more time available. It would consist of performing the same steps but for the prediction dataset already obtained using oversampling. As the train continues to collect data, a larger dataset could be used, allowing to improve the models. In addition, possibly the temperatures which had errors could now be used, if the data is now being correctly measured. It would require a new data analysis to verify if any errors persist.

There are various other methods that can be used for different phases of this work. Another approach could be used instead of linear models for dimension reduction and missing imputation. Further improvements could be achieved by considering the temporal correlations. Regarding the strategies for failure prediction, other approaches could be tested, such as outlier detection or time series forecasting.

The interpretation of the problem and the data preparation represented a large portion of this work. Having these steps done, it would be the time to test different methods for failure prediction.
Bibliography


Appendix A

EDA Detailed Statistics

This Appendix presents the detailed statistics from the EDA. Tables A.1 and A.2 show the statistics of the CCU variables, before and after cleaning, respectively. Table A.3 presents the statistics of only a few TCU variables of context.

Table A.1: The maximum of various parameters is close to $2^{16}$, representing numerical overflows; the last two parameters are constant; the missing rate is considerable, except for the GPS variables. Statistics of the CCU variables before cleaning (M1).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Mode</th>
<th>Std. Deviation</th>
<th>Missing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccu_var_1</td>
<td>0</td>
<td>124</td>
<td>23.35</td>
<td>0</td>
<td>0</td>
<td>30.64</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_2</td>
<td>0</td>
<td>125</td>
<td>23.36</td>
<td>0</td>
<td>0</td>
<td>30.65</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_3</td>
<td>100</td>
<td>134</td>
<td>104.28</td>
<td>102</td>
<td>101</td>
<td>4.62</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_4</td>
<td>0</td>
<td>67</td>
<td>20.53</td>
<td>26</td>
<td>0</td>
<td>21.71</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_5</td>
<td>0</td>
<td>39</td>
<td>11.83</td>
<td>10</td>
<td>0</td>
<td>12.74</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_6</td>
<td>0</td>
<td>140</td>
<td>98.47</td>
<td>113</td>
<td>113</td>
<td>40.64</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_7</td>
<td>-107</td>
<td>65535</td>
<td>496.26</td>
<td>0</td>
<td>0</td>
<td>5797.62</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_8</td>
<td>-104</td>
<td>65535</td>
<td>101.97</td>
<td>0</td>
<td>0</td>
<td>2554.92</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_9</td>
<td>0</td>
<td>270</td>
<td>242.32</td>
<td>244</td>
<td>243</td>
<td>18.18</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_10</td>
<td>0</td>
<td>63</td>
<td>6.67</td>
<td>1</td>
<td>0</td>
<td>11.41</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_11</td>
<td>-855</td>
<td>65535</td>
<td>163.35</td>
<td>6</td>
<td>6</td>
<td>2975.32</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_12</td>
<td>-863</td>
<td>65535</td>
<td>151.86</td>
<td>5</td>
<td>5</td>
<td>2763.76</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_13</td>
<td>-992</td>
<td>65482</td>
<td>837.88</td>
<td>992</td>
<td>992</td>
<td>1843.48</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_14</td>
<td>0</td>
<td>1560</td>
<td>161.65</td>
<td>24</td>
<td>0</td>
<td>276.17</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_15</td>
<td>-1662</td>
<td>65529</td>
<td>132.24</td>
<td>0</td>
<td>0</td>
<td>2520.54</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_16</td>
<td>0</td>
<td>2456</td>
<td>1906.31</td>
<td>2030</td>
<td>2097</td>
<td>271.10</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_17</td>
<td>0</td>
<td>2507</td>
<td>1899.79</td>
<td>2032</td>
<td>2097</td>
<td>282.37</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_var_19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18.25%</td>
</tr>
<tr>
<td>ccu_gps_lat</td>
<td>38.71</td>
<td>39.07</td>
<td>38.77</td>
<td>38.76</td>
<td>38.80</td>
<td>0.04</td>
<td>1.20%</td>
</tr>
<tr>
<td>ccu_gps_long</td>
<td>-9.39</td>
<td>-8.87</td>
<td>-9.23</td>
<td>-9.22</td>
<td>-9.39</td>
<td>0.10</td>
<td>1.20%</td>
</tr>
<tr>
<td>ccu_gps_speed</td>
<td>0</td>
<td>125</td>
<td>23.19</td>
<td>0</td>
<td>0</td>
<td>30.81</td>
<td>1.20%</td>
</tr>
<tr>
<td>ccu_gps_az</td>
<td>0</td>
<td>360</td>
<td>85.49</td>
<td>0</td>
<td>0</td>
<td>114.61</td>
<td>1.20%</td>
</tr>
</tbody>
</table>

67
Table A.2: The missing rate of the parameters was reduced after cleaning the first days with numerical overflows; the constant parameters were removed. Statistics of the CCU variables after cleaning (M1).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Mode</th>
<th>Std. Deviation</th>
<th>Missing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccu_var_1</td>
<td>0</td>
<td>124</td>
<td>23.42</td>
<td>0</td>
<td>0</td>
<td>30.67</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_2</td>
<td>0</td>
<td>125</td>
<td>23.44</td>
<td>0</td>
<td>0</td>
<td>30.67</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_3</td>
<td>100</td>
<td>134</td>
<td>104.31</td>
<td>102</td>
<td>101</td>
<td>4.63</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_4</td>
<td>0</td>
<td>67</td>
<td>20.43</td>
<td>26</td>
<td>0</td>
<td>21.65</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_5</td>
<td>0</td>
<td>39</td>
<td>11.77</td>
<td>10</td>
<td>0</td>
<td>12.71</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_6</td>
<td>0</td>
<td>140</td>
<td>98.20</td>
<td>113</td>
<td>113</td>
<td>40.71</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_7</td>
<td>-107</td>
<td>136</td>
<td>-21.17</td>
<td>-3</td>
<td>0</td>
<td>48.32</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_8</td>
<td>-104</td>
<td>141</td>
<td>2.11</td>
<td>0</td>
<td>0</td>
<td>29.21</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_9</td>
<td>0</td>
<td>270</td>
<td>242.31</td>
<td>244</td>
<td>243</td>
<td>18.33</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_10</td>
<td>0</td>
<td>63</td>
<td>6.69</td>
<td>1</td>
<td>0</td>
<td>11.43</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_11</td>
<td>-855</td>
<td>1047</td>
<td>27.84</td>
<td>6</td>
<td>6</td>
<td>239.13</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_12</td>
<td>-863</td>
<td>1062</td>
<td>35.05</td>
<td>5</td>
<td>5</td>
<td>238.26</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_13</td>
<td>-992</td>
<td>1000</td>
<td>787.39</td>
<td>992</td>
<td>992</td>
<td>566.58</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_14</td>
<td>0</td>
<td>1560</td>
<td>162.26</td>
<td>24</td>
<td>0</td>
<td>276.64</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_15</td>
<td>-1662</td>
<td>1875</td>
<td>36.38</td>
<td>0</td>
<td>0</td>
<td>413.29</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_16</td>
<td>0</td>
<td>2456</td>
<td>1906.55</td>
<td>2034</td>
<td>2097</td>
<td>272.49</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_var_17</td>
<td>0</td>
<td>2507</td>
<td>1899.99</td>
<td>2035</td>
<td>2097</td>
<td>283.86</td>
<td>8.97%</td>
</tr>
<tr>
<td>ccu_gps_lat</td>
<td>38.71</td>
<td>39.07</td>
<td>38.77</td>
<td>38.76</td>
<td>38.80</td>
<td>0.04</td>
<td>1.21%</td>
</tr>
<tr>
<td>ccu_gps_long</td>
<td>-9.39</td>
<td>-8.87</td>
<td>-9.23</td>
<td>-9.22</td>
<td>-9.39</td>
<td>0.10</td>
<td>1.21%</td>
</tr>
<tr>
<td>ccu_gps_speed</td>
<td>0</td>
<td>125</td>
<td>23.24</td>
<td>0</td>
<td>0</td>
<td>30.85</td>
<td>1.21%</td>
</tr>
<tr>
<td>ccu_gps_az</td>
<td>0</td>
<td>360</td>
<td>85.48</td>
<td>0</td>
<td>0</td>
<td>114.57</td>
<td>1.21%</td>
</tr>
</tbody>
</table>

Table A.3: The variables of context do not have missings; some are binary (flags); several variables are constant. Statistics of a few TCU variables of context (M1).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Mode</th>
<th>Std. Deviation</th>
<th>Missing Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcu_var_1</td>
<td>0</td>
<td>1</td>
<td>0.06</td>
<td>0</td>
<td>0</td>
<td>0.24</td>
<td>0%</td>
</tr>
<tr>
<td>tcu_var_3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>tcu_var_35</td>
<td>0</td>
<td>4839</td>
<td>994.96</td>
<td>0</td>
<td>0</td>
<td>1375.19</td>
<td>0%</td>
</tr>
<tr>
<td>tcu_var_553</td>
<td>-368</td>
<td>369</td>
<td>-104.37</td>
<td>-150</td>
<td>0</td>
<td>115.42</td>
<td>0%</td>
</tr>
<tr>
<td>tcu_var_616</td>
<td>0</td>
<td>1</td>
<td>0.99</td>
<td>1</td>
<td>1</td>
<td>0.10</td>
<td>0%</td>
</tr>
<tr>
<td>tcu_var_619</td>
<td>0</td>
<td>1</td>
<td>0.56</td>
<td>1</td>
<td>1</td>
<td>0.50</td>
<td>0%</td>
</tr>
<tr>
<td>tcu_var_900</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>