Predictive Maintenance for Air Production Unit in EuroTram Vehicles

Mariana Chaves de Barros

Master Thesis in Modelling, Data Analysis and Decision Support Systems

Supervised by:
João Gama
Rita Ribeiro

Faculdade de Economia
Universidade do Porto
2020

Acknowledgements

First, I would like to express my gratitude to my thesis supervisor, professor João Gama. Thank you for guiding me and my work and being always available to help me when needed. I want to thank my co-supervisor, professor Rita Ribeiro, for being present during the whole journey and for helping me with the writing of this dissertation. I also want to thank Bruno Veloso, for giving me the technical support I needed during the whole journey.

I would like to thank the project Failstopper, for providing the data and the support necessary to achieve the goals proposed in this dissertation.

I want to thank my family, for the support they gave me during this important step of my life, and for always understanding my effort.

To my boyfriend, thank you for being my true companionship in life and for supporting me during this whole journey. I thank my friends, for their patience and support, even when I have to skip our meetings.

Finally, I thank my co-workers for supporting me achieving my personal goals and giving me opportunity to evolve.

Abstract

The transformation of industrial manufacturing with computers and automation with smart systems leads us to monitor and log of industrial equipment events. It is possible to apply analytic approaches, and to find interpretive results for strategic decision making, providing advantages such as failure detection and predictive maintenance. Over the last years, many researchers have been studying the application of machine learning techniques to improve such tasks.

In this context, we developed a system capable to distinguish the normal and abnormal working of the Air Production Unit (APU). The study started with the analysis of the sensors installed on the APU to define its normal behavior and its failure mode.

In a first approach, we considered the peak frequency of each sensor to characterize the normal and abnormal work of the APU, and we have defined rules that together, were able to predict 100% of the known failures.

In a second approach, we applied K-Means clustering to define and characterize the different working modes of the compressor. Also, the outlier detection based on DBSCAN clustering was able to predict 50% of the know failures, at least one hour before they happen.

After defining the normal and abnormal working of the Air Production Unit, the third approach consists on predict failures with Machine Learning. We have compared the performance of Autoencoders and Long Short Term Memory Autoencoders on failure prediction. The use of simple Autoencoders provided more accurate results with less computational effort and was capable of predict failures 5 minutes before they happen.

Keywords: Predictive Maintenance, Anomaly Detection, Frequency Analysis, Outlier Detection, Prediction

Resumo

A transformação industrial com computadores e a automação com sistemas inteligentes permite-nos monitorizar e registar eventos de equipamentos em tempo real. Estes registos permitem-nos fazer uma análise interpretativa dos resultados, de forma a melhorar o sistema através da deteção de falhas. Recentemente, várias técnicas de *Machine Learning* foram aplicadas de forma a detetar anomalies em sistemas industriais. Neste contexto, desenvolvemos um sistema capaz de distinguir o funcionamento normal e anormal do sistema da Unidade de Produção de Ar (APU) e prever falhas no sistema. Começamos por analisar os sensores instalados na APU, definindo o seu comportamento normal e caracterizando as falhas. Inicialmente, consideramos a frequência de picos de cada sensor e definimos regras para monitorizar a APU em tempo real. Em conjunto, estas regras permitiram-nos prever 100% das falhas conhecidas.

Numa segunda abordagem, utilizamos *K-Means clustering* para definir e caracterizar os diferentes tipos de falha do compressor. A deteção de *outliers* atrave *DBSCAN clustering* mostrou-se ainda capaz de detetar 50% das falhas.

Depois da caracterização do comportamento normal e anormal da APU, a terceira abordagem está relacionada com a previsão de falhas através de *Machine Learning*. Para isso, comparamos a utilização de métodos como *Autoencoders* e *Long Short Term Memory Autoencoders* na previsão de falhas. O uso de *Autoencoders* obteve melhores resultados com menos esforço computational e foi capaz de prever falhas 5 minutos antes delas acontecerem.

Palavras-chave: Manutenção preditiva, Deteção de anomalias, Análise de frequências, Deteção de outlier, Previsão

Contents

List of Figures

List of Tables

Acronyms

The following acronyms are mentioned in this document:

ADF Augmented Dickey Fuller

ANN Artificial Neural Network

APU Air Production Unit

AUC Area Under Curve

BN Bayesian Networks

CSV Comma Separated Values

DBSCAN Density-Based Spatial Clustering of Applications with Noise

DL Deep Learning

GBT Gradient Boosted Tree

GPS Global Positioning System

IoT Internet of Things

KNN K-Nearest Neighbor

FMEA Failure Mode and Effects Analysis

FMECA Failure Mode, Effects and Critical Analysis

FP False Positive

LSTM Long Short Term Memory

ML Machine Learning

MSE Mean Squared Error

OCC One Class Classifier

PCA Principal Components Analysis

PCCAD Principal Component Classifier based on Anomaly Detection

PdM Predictive Maintenance

RCM Reliability-Centered Maintenance

RF Random Forest

RNN Recurrent Neural Network

ROC Receiver Operating Characteristic

RUL Remaining Useful Life

SVM Support Vector Machine

SVR Support Vector Regression

TP True Positive

Abbreviations

The following abbreviations are mentioned in this document to identify the components of the equipment being studied:

TP2 Pressure on the compressor.

TP3 Pressure on the pneumatic panel.

H1 Pressure generated due to pressure drop when the discharge of the cyclonic separator filter occurs.

DV Pressure Pressure is exerted due to pressure drop generated when towers are discharged.

COMP Sensor that represents the electrical signal of the air intake valve on the compressor.

DV Electric Sensor that represents the electrical signal of the solenoid valve.

TOWERS Sensor that defines which tower is drying the air and which tower is draining the humidity.

MPG Sensor that activates the COMP intake valve to start the compressor under load.

LPS Sensor that is activated and the pressure is lower than 7 bar and the equipment stops working.

Chapter 1

Introduction

The project FailStopper comes from a partnership between Metro do Porto S.A. and INESC TEC. The project was approved in the Scientific Research and Technological Development Project Competition for Data Science and Artificial Intelligence in Public Administration, funded by the Foundation for Science and Technology (FCT).

The goal of the project is to predict failure on Eurotram vehicles in an operational context. The vehicles have a compressed air system with an Air Production Unit (APU) that decreases the cabin vibrations and levels the vehicle so that it is always at the same height as the track. This system is vital to the work of the vehicle and if it breaks down, the vehicle should be removed from service immediately, without any previous warning. The data that supports the project is collected from sensors that are installed on the APU of the train. The monitoring and logging of industrial equipment events, like temporal behavior and fault events is obtained from records generated by sensors in most different parties of an industrial plant. The monitoring of sensors values allows to predict possible failures and enhance the reliability of trains, signals, and tracks (Li and He, 2015). They monitor equipment to generate alerts about the need for attention on critical elements of the train. They establish a fully functioning, accurate, real-time, and efficient transportation management system, by combining information systems and technologies such as wireless networks, sensors, computing/networking devices and Global Positioning System (GPS).

This work was developed at the first stage of the project. The objective is to analyze the data received from the sensors installed on the APU, understand how the system operates in different contexts and characterize the

abnormal behavior of the APU. The ultimate goal is to predict failures.

The main contributions of this dissertation to the project is the characterization of the abnormal behavior of the system before failure happens. Our work is divided in three main steps. The first step includes the characterization of normal and abnormal behavior of the APU based on peak frequency and peak duration, and the definition of rules that alert about possible failures. The second step is about the characterization of the different normality modes of the system based on clustering. Also, the clustering approach allows to characterize the abnormal behavior and predict failures based on outlier detection. The last step compares Autoencoders and Long Short Term Memory Autoencoders to predict failures. The goal is to verify if it is possible to predict failure at this stage of the project, and the algorithm that has a better performance on this task.

Summing up, this work contributes to identify the possible ways to apply predictive maintenance on the APU system. This will avoid the sudden train stop, clients dissatisfaction and increasing maintenance costs.

This dissertation led to a publication in an international conference, ECML-PKDD: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.

Dissertation outline

This thesis is organized as follows: In Chapter 2, an extensive literature review about Reliability Centered Maintenance and Predictive Maintenance applied to railway industry is presented. Chapter 3 is about the data understanding. It consists on the data description, data preprocessing and exploratory analysis of the data. In Chapter 4, an approach of data analysis based on peak frequency is presented. In Chapter 5, we characterized the dataset using clustering methods, and tested clustering as an anomaly detection approach. In Chapter 6, the Machine Learning models that can be applied to this predictive maintenance problem are explored. It starts with the data preprocessing and compares different approaches to predict the failure of the APU, based on the historical data. Chapter 7 presents the conclusions of this study, the main limitations and some suggestions for future work.

Chapter 2

Related Work

The transformation of industrial manufacturing with computers and automation with smart systems lead us to use sensors to monitor industrial equipment events.

In this chapter, we will present some studies developed in the context of predictive maintenance and anomaly detection, focusing on the railway industry.

2.1 Reliability Centered Maintenance

Since 1940, maintenance techniques are evolving and new goals are being achieved, with the implementation of Reliability Centered Maintenance (RCM) (Brauer and Brauer, 1987) policies. RCM is an elaborate seven-step structured process that regulates equipment maintenance strategies and may be inclusive of condition-based, predictive, and planned maintenance (Kennedy, 2006). Also, the RCM aims to determine the maintenance processes required to ensure the expected equipment performance.

There are several maintenance techniques used by RCM models: Preventive Maintenance, Run to Failure, and Condition-based Maintenance. The taxonomy of these techniques is presented in Figure 2.1 and explained next.

Preventive maintenance is executed through periodical inspections where equipment parts are replaced based on a fixed timetable (Bengtsson, 2004).

Run to failure also referred to as Corrective Maintenance, waits for the failure and then proceeds to repair it (Li, Gao, Tang, and Li, 2016).

Condition-based maintenance is a maintenance strategy, which conducts

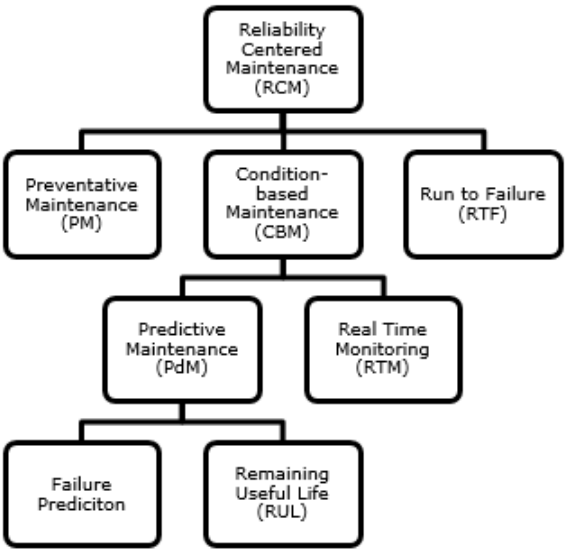


Figure 2.1: Reliability Centered Maintenance (RCM) techniques

the maintenance process using the information gathered via condition monitoring. This strategy is responsible for reducing the maintenance cost through maintenance actions when the machine failure process is approaching (Koonsstapf, 2015). A Real-time Monitor is employed to schedule tasks with random execution times in a real-time computing system. Predictive maintenance (PdM) measures the condition of the equipment, determines when it will fail in a specified future period, and then proceeds to take action to avoid system failure.

According to the literature (Li et al., 2016), researchers have formulated PdM problems from different perspectives. They are Failure Prediction, Remaining Useful Life, and Fault Explanation. To solve those problems, Machine Learning methods, and more recently Deep Learning have been applied. Those methods improve the prediction task with Artificial Intelligence, which provides the system with the ability to automatically learn and improve from experience.

Failure Prediction predicts the probability of equipment to fail. One of the biggest issues found was that faults are hypothesized according to a given (static) behavior. However, some of these faults may be not faithfully representing the operating behavior of a component and may not represent

a system failure (Manco, 2017).

Remaining Useful Life (RUL) estimates the remaining useful lifetime of the equipment and is usually aligned with the condition-based maintenance approach. In Fumeo, Oneto, and Anguita (2015) they trained an Online Support Vector Regression (SVR) algorithm to predict the RUL of the axle bearing of the train. The propose was to maximize its RUL, based on sensors data collection. They imputed data on the Online SVR algorithm, trained the model, and applied a meta-heuristic optimization approach to reduce the time needed to perform the task.

In Wang, He, Cui, and Li (2018) the authors have predicted the RUL of the train wheel and the type of failure that will occur by applying a multitask learning method. They have extracted features from RUL data and Failure Type data and used the common features to model the SVR to predict the RUL of the equipment.

Fault Explanation looks for causes and explanations about what could have happened when a failure occurs. Wang et al. (2018) used SVM for failure type classification. The faults can also be explained by providing a snapshot of the features of the device that exhibit abnormal values, such as Pearl (2019). Recently, Structural Causal Models have been used to classify failure, which deploys three parts: graphical models, counterfactual logic, and structural equations.

2.2 Anomaly Detection

An anomaly is an occurrence that deviates from what is standard, normal, or expected. Detecting the presence of anomalies can provide valuable insights into the industry to prevent unexpected equipment faults and applying predictive maintenance. Anomaly detection techniques follow different approaches according to three main scenarios (Jiawei Han and Pei, 2012):

- the data set has no temporal characteristics and instances are independent;
- the data set follows a sequential order and it is assumed that the data is generated in streams;
- the data set is produced as time series.

Most of the works identify outliers and replace or remove them, to have a cleaner data set. But, anomalies can be associated with errors and faults

that are worth to be identified and investigated. Chandola, Banerjee, and Kumar (2009) has classified anomalies into three different types.

Point Anomalies: represents an individual data instance that can be considered anomalous for the rest of the data.

Contextual Anomalies: represents a data instance that is only considered anomalous in a certain context.

Collective Anomalies: represents a subgroup of data instances that are anomalous considering the entire data set. Collective anomalies have been most commonly explored in time-series data, such as data that is collected from sensors in short periods.

Over the last years, several studies have been carried out regarding the application of Machine Learning methods for predictive maintenance and anomaly detection. The aim is to find the best methodologies to detect failure and anomalies on equipment by analyzing the data received from sensors. Those approaches are applied considering if the Machine Learning problem is supervised, semi-supervised, or unsupervised. Supervised tasks usually are about a labeled data set, and the problem to be solved is often predicting the labels for data points without a label. Unsupervised learning is when the algorithm does not have labeled data to train, such as in the clustering approach. Semi-supervised is when labeled and unlabeled data are used. Some layers are learning the structure of the data (unsupervised) and one layer is used to make the classification (trained with supervised data).

Rabatel, Bringay, and Poncelet (2011) focused on the field of train maintenance, using sensors positioned on the main train components.They have extracted normal behavior using a set of Sequential Patterns applying these patterns in normal and abnormal data to evaluate the conformity score.

Pereira, Ribeiro, and Gama (2014b) developed a failure detection system to predict train door breakdowns before they happen using data from the logging system. The authors developed a system for classifying anomalous open/close cycles within trains, based on the difference between the inlet and outlet pressure in specific intervals of the cycle. Cycle classification has been tested under three different approaches: outlier detection based on boxplot; novelty detection with One-Class Classification; and, supervised learning with Support Vector Machines (SVM).

Manco (2017) developed an application to predict and explain door failures on metro trains. The authors used outlier detection to predict the failure, considering the data that does not share common patterns as failures. Results show that high-degree outliers are effective indicators of failures.

Lee (2017) applied Linear Regression to model two different classes of compressor behavior for each train separately.

Benedetti, Leonardi, Messina, Santoro, and Vasilakos (2018) designed a learning approach to detect possible anomalies and created a predictive maintenance model in photovoltaic systems. The anomaly detection algorithm is based on the comparison between the measured and the predicted values of the alternate current power production. The system also recognizes degradation trends-

Kang, Sristi, Karachiwala, and Hu (2018) detected anomalies on train speed for intelligent railway systems. The authors adopted a Bayesian statistical learning model to represent normal behavior of train speed changes and detect the anomaly based on them.

All of these works were developed with the goal of extract the normal behavior of the system, considering as anomalies the data that does not share common patterns. To achieved this goal, different Machine Learning methods were applied, depending on the context and characteristics of the work. However, adaptive detection of anomalies in deployment environments is an important challenge for assuring the quality of sensor measurements. These changes increase the false alarm rates and therefore affect the effectiveness of detection models.

2.3 Predictive Maintenance

Section 2.1 of this work defined predictive maintenance as a maintenance technique that measures the condition of the equipment, determines when it will fail, and then proceeds to take action to avoid system failure.

The integration of digital information from many different sources and locations drives the physical act of maintenance, manufacturing, and distribution, in an ongoing cycle. First, information is captured from the physical world and digitized (physical to digital). Second, the focus is to share and analyze data to generate meaningful insights. Finally, the loop is closed with a digital-to-physical transformation of those insights into real-world actions (Coleman, Damodaran, Chandramouli, and Deuel, 2017). This is usually

achieved using Machine Learning algorithms.

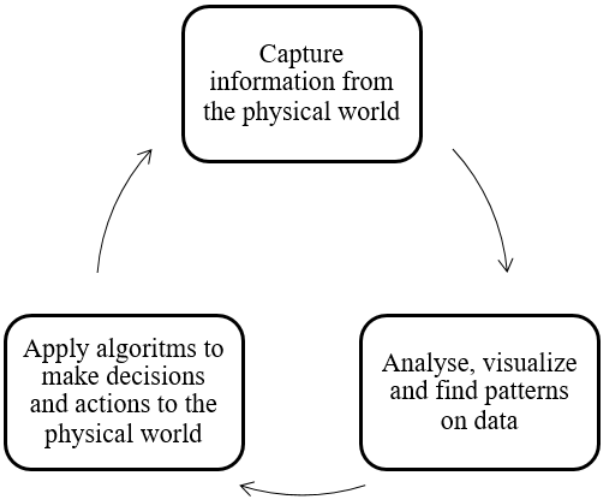


Figure 2.2: Predictive maintenance cycle

Like any knowledge discovery process, diagnostic data processing for fault detection purposes involves five steps: acquisition, preprocessing, model selection, evaluation, and model maintenance. Historical data acquisition is aimed at importing into the prediction system the history of both events and failures. Also, data from sensors is usually considered very noisy, can contain errors because of the multi-source information, and contain incomplete readings (Rabatel et al., 2011). Predicting future maintenance needs of equipment can be approached in many different ways.

PdM methods are mainly divided into two following groups: Model-based and Data-driven based (Liao and Kottig, 2014).

Model-based consists of modeling the system degradation evolution in discrete or continuous time, and its performance depends on the prior knowledge of system degradation processes.

Data-driven based is considered the most effective approach to be adopted in PdM solutions, because of its robustness. It uses sufficient data without knowing the physical nature of the degradation mechanism and the performance only depends on signal processing and feature engineering techniques (Ribeiro, Singh, and Guestrin, 2016). The historical data of equipment, statistical models, reliable functions, and artificial intelligence methods, es-

pecially Machine Learning, are widely used for both RUL estimation and failure prediction (Yan, Wan, Zhang, Tang, Hua, and Wang, 2018). Data-driven diagnostics methods generally require more data than model-based prognostics (Yuan and Liu, 2013). In the literature (Ge, Fu, Syed, Baig, Teo, and Robles-Kelly, 2019a), the number of data-driven approaches is compared to model-based approaches to predict when a piece of equipment will be at risk of failure in the future, and the results suggest that data-driven approaches outperform model-based approaches.

2.3.1 Traditional Machine Learning Approaches

Traditional Machine Learning (ML) methods, such as Decision Trees, Random Forests, Gradient Boosting Trees, and Support Vector Machines, have been used in the last years in many areas of study.

Fernández-Delgado, Cernadas, Barro, and Amorim (2014) proved that tree-based classifiers provide optimal prediction results for the structured data set in academic literature and many industry challenges.

Sharma, Cui, He, Mohammadi, and Li (2018) developed a data-driven condition-based policy for the inspection and maintenance of track geometry and applied Random Forest, Support Vector Machines, and Logistic Regression.

A tree-based classification was proposed by Allah Bukhsh, Saeed, Stipanovic, and Doree (2019) to estimate the performance state of an asset and predict the maintenance need. Predictive models based on Decision Trees, Random forest, and Gradient Boosted Trees were developed. For predictive maintenance, the GBT model performed most optimally as compared to other methods with 86% accuracy.

Badnjević (2019) studied the performance of medical devices on patient diagnosis and they have used five different ML algorithms that are Decision Tree, Random Forest, K-nearest Neighbor, Support Vector Machines, and Naïve Bayes algorithm. The systems based on the Random Forest classifier with Genetic Algorithm feature selection yielded the highest accuracy among other Machine Learning systems.

In a review of supervised ML algorithms (Alom, 2019), it was possible to conclude that a rule-based system (e.g. decision tree) has a better performance when dealing with discrete or categorical features, and Support Vector Machine and Neural Network are better for the prediction of continuous attributes.

Online Learning

Online Learning (Rosasco and Poggio, 2015) is a field of Machine Learning that embraces the fact that learning environments can change every second.

One-Class Classifiers are also widely used in Online Outlier Detection (OOD) (Amer, Goldstein, and Abdennadher, 2013; Zhang, Meratnia, and Havinga, 2009). One-Class SVM for anomaly detection was analyzed by Amer et al. (2013) and revealed an accuracy of almost 99%.

An approach based on One-Class Quarter-Sphere Support Vector Machine was proposed by Havinga (2010), with a model that represents the normal behavior of the sensed data, and that identifies an outlier as a sensor measurement that does not conform to this model.A quarter-sphere centered at the origin is fitted to incorporate the majority of the data vectors in the higher dimensional space. The vectors that fall inside the quarter-sphere are normal and those falling outside the quarter-sphere indicate anomalous data (Zhang, Meratnia, and Havinga, 2008). This approach was firstly introduced by Laskov, Schäfer, and Kotenko (2004) and obtained good results in the literature (Rassam, Maarof, and Zainal, 2014; Zhang et al., 2008).

One-Class Principal Components Analysis was used by Rassam et al. (2014) for online anomaly detection, where the authors created a Principal Component Classifier based Anomaly Detection model, which was trained for the first time with offline data and detects anomalies with online data, based on the normal reference model built in the training phase. An Adaptive PCCAD (APCCAD) was proposed to incorporate an incremental learning method in the design of the PCCAD model to detect anomalous measurements in real-time, adding a classification criterion to the online detection phase that further checks the abnormality of the measurement based on the statistical characteristics of the environment. The Online Support Vector Regression approach (Wang et al., 2018) incrementally increased or decreased the regression parameters every time a new sample is added, and the trained SVR function is updated.

After the analysis of these approaches, we can state that the most common Online Learning techniques are based on the One-Class Classification approach. The goal of this approach is to identify objects of a specific class among the other objects, by primarily learning from a training set containing only the objects of that class (Oliveri, 2017).

2.3.2 Deep Learning Approaches

Deep Learning (DL) is a new field of Machine Learning, has been growing rapidly, and has been applied to most traditional application domains, as well as some new areas that present more opportunities. Some of the most used DL approaches are Deep Neural Network, Convolution Neural Network, Recurrent Neural Network, Long Short Term Memory, or Gated Recurrent Unit. The big difference between traditional ML and DL is in how the features are extracted. ML approaches use handcrafted engineering features by applying several feature extraction algorithms and then apply the learning algorithms. In the case of DL, the features are learned automatically and are represented hierarchically at multiple levels. DL algorithms are applied in cases where the solution of the problem changes over time, such as tracking, predictions, stock, preferences, and others.

A systematic review of PdM with an emphasis on DL techniques was presented where the authors refer to DL benefits and limitations for fault diagnosis and prognostics, in addition to future opportunities (Khan and Yairi, 2018).

Jiawei Han and Pei (2012) used the Deep Belief Network in the context of fault diagnosis systems of vehicle on-board equipment and the method obtained better results to ANN and KNN.

Deep Belief Network was also implemented to predict failure with good results compared to other methods (Tamilselvan and Wang, 2013).

Ge, Fu, Syed, Baig, Teo, and Robles-Kelly (2019b) used for intrusion detection for IoT with 95,04% of accuracy.

Focusing on the infrastructure of the railway, Lopes Gerum, Altay, and Baykal-Gürsoy (2019) proposes a new approach that relies on easy-to-obtain data and integrates defect prediction with inspection and maintenance scheduling activities. K-Means were used to perform feature selection and after that, they compared methods to predict the number of defects: Random Forests and Recurrent Neural Networks. To find the optimum inspection policies they used a Markov decision process to integrate the stochastic nature of defect occurrence into scheduling. The performance of RF and RNN was similar.

A new Dynamic Predictive Maintenance framework using DL for failure prognostics was proposed, based on Long Short Term Memory (LSTM) (Nguyen and Medjaher, 2019).

Maya, Ueno, and Nishikawa (2019) proposed a delayed Long Short-Term

Memory (dLSTM) for time-series data and obtained good results. Deep Reinforcement Learning (DRL) allows the online re-adaptation of the predictive critic network following the new information and accounting for the human expert advice (Koprinkova-Hristova, 2013).

Borghesi, Bartolini, Lombardi, Milano, and Benini (2019) propose a semi-supervised anomaly detection approach based on a Neural Network called Autoencoder that learns the behavior of a cluster in the normal state, monitoring a period when the supercomputer is operating without faults. The Autoencoder-based approach improved accuracy by 12% compared with other algorithms coming from the semi-supervised area. Autoencoder reveals to be, in some cases, more efficient than using the LSTM method, RF, and Xgboost (Yeh, Hsu, and Yeh, 2019).

2.4 Predictive Maintenance in the Railway Industry

Table 2.1 summarizes the methodologies and results of the works that have been collected, related to anomaly detection and failure prediction in the railway context. Some of them use anomaly detection methods based on the characterization of normal and abnormal behavior of the compressor. Others use outlier detection based on clustering to detect anomalies. Most of them use classification methods to detect anomalies and failures. However, the classification methods obtain better results when the data set is labeled.

The contributions of our work in this context are the inclusion of the peak frequency as an anomaly detection method and the exploration of Deep Learning as a failure prediction approach in the context of the railway industry with an unlabeled data set.

Table 2.1: Methodologies applied and results obtained on the literature

Work	**Methodologies**	**Results**
Rabatel et al. (2011)	Sequential Patterns Mining	The model obtain recall and precision above 90%
Pereira et al. (2014b)	Unsupervised learning based on boxplot; One Class Classification (OCC); One Class Classification Suppor Vector Machine (OCCSVM)	OCC do not guarantee an adequate level of reliability and SVM have good performance with some weaknesses

Goodman, Hofmeister, and Wagoner (2015)	Microelectromechanical Sensor System	The detection system is efficient enough to identify, locate, and characterize such types of anomalies.
Fumeo et al. (2015)	Online Support Vector Regression (OL-SVR); K-Fold Cross Validation (KCV)	Prove the generality of the method
Wang, Xu, Tang, Yuan, and Wang (2017)	Bayesian Networks (BN) with Monte Carlo	BN model with Monte Carlo simulations was proved more accurate than ANN and SVM to model the time-based pattern
Wang et al. (2018)	Support Vector Regression and (SVR) and Support Vector Machine (SVM)	Multi-task feature selection outperforms both the single-task (3%) and Random Forest (1%)
Sharma et al. (2018)	Random Forest (RF), Support Vector Machines (SVM) and Logistic Regression	RF performed most optimally as compared to other methods with 75,2% accuracy
Badnjević (2019)	Decision Tree, Random Forest (RF), K-Nearest Neighbor (KNN); Support Vector Machines (SVM).	Random Forest classifier with Genetic Algorithm feature selection yielded highest accuracy among others
Allah Bukhsh et al. (2019)	Tree-based classification techniques (Decision Trees, Random Forest and Gradient Boosted Tree)	GBT model performed most optimally as compared to other methods with 86% accuracy
Lopes Gerum et al. (2019)	RF and RNN and K-means	RF and RNN have similar results with accuracy around 82% for red defects and 62% for yellow defects
Borghesi et al. (2019)	Autoencoder	Autoencoder outperforms algorithms used for anomaly detection in 12%

Chapter 3

Data Understanding

The present study was developed at an early stage of the project, when one of the priorities was to explore the data set, analyze each feature, and understand the feature extraction process.

In this chapter, we will describe each feature of the data set, do a exploratory analysis, and analyze the correlations between them.

3.1 Data Description

In this section, we will describe the sensors that were installed on the Air Production Unit (APU). The APU is part of a compressed air system and converts the power from an electric motor into kinetic energy by compressing and pressurizing air.

When the system fails, the equipment that is receiving its energy will consequently fail. Applying predictive maintenance to predict the equipment failure before it happens, decreases costs and optimizes the service, by lowering the number of equipment taken out of service for inspection and maintenance.

A study made by Pinto (2019) has defined the sensors that should be installed on the APU, to create a system for the acquisition of data. The system failures have been analyzed using Reliability Centered Maintenance Methods (RCM), Failure Mode and Effects Analysis (FMEA), and Critically Analysis (FMECA). FMEA analyzes the system failures to decrease damages on equipment. It consists on a table that evaluates the system faults according to its severity. The most common system failures from the previ-

ous three years and their root causes were identified to select the components to monitor. Thus, the most common failures on the APU identified were:

1. valve A4, that is triggered when the system pressure drops below 8.2 bar, leading to excessive air loss and low pressure;
2. the cyclonic separator filter discharge valve that causes air to be continually drained outwards, lowering pressure from the APU;
3. oil leaks that cause friction to increase compressor speed, decrease air-cooling, and affect engine efficiency;
4. motor failure;
5. pressure switches drying towers with pressure reading failures;
6. failure of the drying towers, that affect the humidity of the air but does not decrease its pressure.

The sensors and the places to install them were strategically defined, considering the most common failures of the system. The installation of eight analogical sensors and eight digital sensors was defined as the best way to monitor the APU behavior. The Figure 3.1 represents the components that compose the structure of the APU and the analogical sensors that were installed on it. The digital sensors are represented bu the Eletrinic Control Unit.

The analogical sensors present continuous variables and digital sensors present categorical values. They only return two different values: 0 when they are inactive or 1 when they are triggered by a certain event.

The sensors installed on the trains are described in Table 3.1.

The sensors communicate with a server that receives the data from the sensors with one second of sampling frequency. Every second, the sixteen sensors data, the timestamp, the GPS location, and speed are logged in a data file. Every five minutes, the file is sent to the server using the TCP/IP [1] protocol application.

The sensors were installed in two different trains named as APU01 and APU02. It is important to refer that on APU01, the sensor that measures the Flowmeter was not installed and the sensor that monitors the Oil Level was installed in reverse. Also, on APU02 the sensors that measure the pressure on Reservoirs and the Pressure switch drying towers present a defect since its installation. They will not be considered on some of the analysis, because they can negatively impact them.

[1] a protocol suite used to transfer data on the internet.

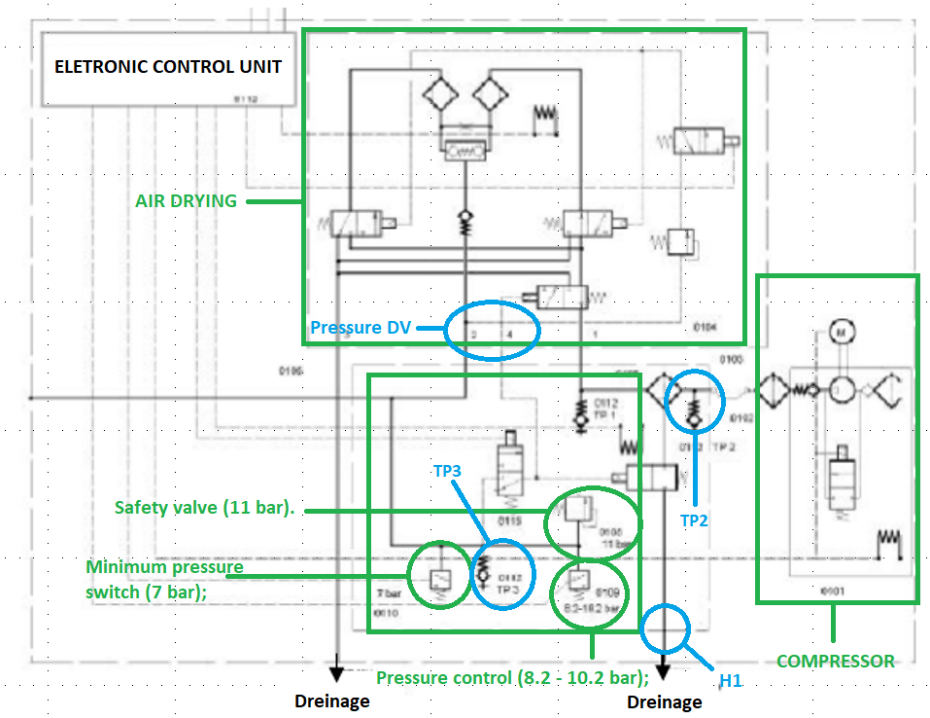


Figure 3.1: Illustration of the APU system and its components (Pinto, 2019)

3.1.1 Failure Reports

In this section, we will introduce the known failures that will be used to evaluate the predictive maintenance techniques applied to our data set.

The first part of this work, which consists on the data analysis, is based on the data that was collected over two months, from 1st of March until 30th of April. It includes the data analysis and exploration, the identification of patterns, and the detection of anomalies on those patterns The experimental evaluation, which includes the evaluation of anomaly detection approaches will be based on the data collected over five months, from 1st of March until 31st of July.

The data set is not labeled, and the only information that we have about the failures is the monthly report received by the company, represented in Table 3.2. Apart from those, Table A.1 contains the moments when both APU01 and APU02 reported anomalies and went to maintenance. Some

of those were reported as failures on the APU and others were reported as failures on other components of the train.

3.1.2 Data Preprocessing

The data from the APUs is collected every second, with 3600 samples of data per hour and 86400 samples of data per day. It means that the techniques applied will deal with a large amount of data, requiring high computational work. Large amounts of data collected from IoT systems face some challenges, according to D'mello (2019):

- data preparation still accounts for up to 80% of the time and resources involved in any data project, and the more data you add, the more time-intensive that process will become;
- preparing timestamp or GPS data and combine it with more structured sources, such as CSV files is a complex process;
- business computer systems — both hardware and software — are not made to exchange or process the vast amounts of complex information pulled from sensors and connected devices.

In addition to the challenges of a large amount of data, the data that is directly received from the analogical sensors should be re-transformed. The sensors transform the quantities that they are subject to an electric current, between 4 and 20mA. Then, we have to re-transform the electric current received from the sensors into the correct units of measure of each. On TP2, TP3, H1, DV Pressure, Reservoirs were transformed using the following equation:

$$x = 16 * (1 - (20 - x)/16) \tag{3.1}$$

The Oil Temperature and Motor Current were transformed using the following equation:

$$x = (x * 12.5) - 50 \tag{3.2}$$

The Flowmeter was transformed using the following equation:

$$x = ((x - 4) * 74.75)/16 + 0.25 \tag{3.3}$$

The high amount of data is one of the biggest challenges of this project. Since we are at an early stage of the project, every sample can be meaningful

and should not be ignored. Process all the information is time consuming and increases the time of the experimental evaluation of the models.

Before applying ML techniques is important to standardize or normalize the data (Hale, 2019), since we have different scales on our data set. MinMaxScaler() is a normalization function available in *sklearn* (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot, and Duchesnay, 2011) library from Python. It subtracts the minimum value in the feature and then divides by the range. The range is the difference between the original maximum and the original minimum. Also, normalization preserves the shape of the original distribution, which is an advantage when collection data from sensors. Based on this, MinMaxScaler() will be used to perform normalization on our data set when implementing ML techniques.

3.2 Data Exploration

In this section, a descriptive analysis of the variables is presented. This analysis is based on the data collected from 1st of March until 30th of April.

For the analogical sensors that assume numerical values, this analysis consisted of studying the dispersion and shape of the variable distribution. To measure the dispersion of the data, a proper metric is the Coefficient of Variation (CV). This metric is independent of the scale of the variable and it is calculated by dividing the standard deviation by the variable mean.

To study the shape of a variable distribution, the skewness and the kurtosis were considered. A variable has a positively skewed distribution if Fisher's skewness coefficient is positive. Otherwise, it has a negatively skewed distribution. To study the tails of the variable distribution compared with the Normal distribution, the kurtosis coefficient was considered. If it has a positive value, the tails of the variable distribution are heavier than those of the Normal distribution (Joanes and Gill, 1998). Otherwise, the tails of the variable distribution are lighter than those of the Normal distribution.

The Kolmogorov-Smirnov is usually used to test if the variable follows a normal distribution. However, our data set is too large and the test returned a p-value equal to 0. The visual analysis of the histograms is the more reliable way to test the normality of large data sets. The histograms of each variable are represented in Appendix A.1.

The results of the descriptive analysis of the numerical variables of the

APU01 are presented in Table 3.3 and Table 3.5 and the results of the APU02 are presented in Table 3.4 and Table 3.6.

For the categorical variables. we analyzed the frequency of each variable activation and the results are presented in Table 3.7. The goal is to characterize its behavior and compare the frequency between APU01 and APU02.

3.3 Data Analysis

In this section, we will analyze the data received from the two compressors, APU01 and APU02, and compare them. Also, we will verify if the compressor's behavior is the expected on both. This analysis is based on the data collected from 1st of March until 30th of April.

Appendix A.2 contains the time series representation of the work of APU01 and APU02 on the 1st of April.

Regarding APU01, the variable Flowmeter does not change over time, because this sensor was not installed on APU01. The Oil Level is always equal to 1, meaning that the level of the oil is different from the expected. This is explained by the fact that this sensor was installed in reverse, as referred on Section 3.3. The behavior of the TP3 variable is similar in Reservoirs, as expected. The sensor TP3 is expected to measure the pressure on the pneumatic panel and the Reservoirs sensor is suppose to read the pressure in Reservoirs, explaining the evolution of the pressure downstream on the reservoirs. This sensor was strategically placed to check if there is some consumption downstream of the valve installed before the reservoirs. The fact that the values are close to each other, means that the air is received in the Reservoirs with pressure very similar to that coming out of TP3, without loss.

On APU02, the values of Reservoirs are different from the values on TP3, because this sensor presents a defect on this APU. The same happens with Pressure switch drying towers on APU02. Visually, it is easy to identify that APU02 presents more peaks on the analogical sensors, meaning that the working mode of the APU02 changes more often.

The comparison between the two APUs on the digital sensors is represented in Table 3.7. Comparing to APU01, the APU02 is working off loaded more time and under load during less time. The values of COMP and MPG are closer because those two sensors are expected to assume the same val-

ues. When COMP is equal to 0, DV Electric should be equal to 1, and the opposite. This is the behavior described by APU02, but it presents some incompatibilities on APU01. The behavior of COMP, DV Electric, MPG, and LPS is similar on the two APUs. The other sensors present anomalies on one of the APUs and that is why they assume different values.

3.3.1 Correlation Analysis

The correlation between variables is also a good measure to understand how the APU system works. The heat maps that represent the correlations between variables on APU01 (cf. Figure A.3), APU02 (cf. Figure A.4), and on the whole data set (cf. Figure A.5, representing both APUs) are represented in Appendix A.1. Based on them, it is possible to conclude that:

- the sensors COMP, MPG, DV Electric, TP2, and H1 are highly correlated;
- the sensor Oil Temperature has a significant correlation with the sensor TP3;
- the sensors Pressure switch drying towers, Oil Level, and Caudal Impulses are not highly correlated with any other sensor;
- the sensor LPS presents its higher correlation with the sensor TP3. However, the correlation is higher on APU02;
- on APU01, the sensor TP3 is extremely correlated with the sensor Reservoirs;
- on APU02, the sensor Flowmeter is extremely correlated with the sensor TP2 and the other sensors that were referred on the first point.

3.3.2 Time Series Analysis

In this section, we will analyze some of the time series properties and verify if they fit on our data set.

Stationarity

A time series is said to be stationary if its corresponding statistical properties like mean, standard deviation, and autocorrelation remain constant throughout the time. We have performed the Augmented Dickey Fuller (ADF) test (Dickey and Fuller, 1979) to check if our data set is stationary. The null hypothesis of the ADF test is that the time series is non-stationary. Since the p-value of the ADF test is equal to 0.00001, less than the significance level of 0.05, we reject the null hypothesis and conclude that our data set is stationary.

Periodicity

A time series is periodic if it repeats itself at equally spaced intervals. The Figures 3.2 and 3.3 shows the periodicity on our data set, on both APU01 and APU02, in a period of one hour. We can confirm the periodicity on both APU01 and APU02. However, they present different repetition periods.

Autocorrelation

Autocorrelation is a technique for analyzing seasonality. It plots the correlation of the time series with itself at a different time lag. The plots in Figures 3.4 and 3.5 represent the autocorrelation on both APU01 and APU02. The x-axis represents the lag and the y-axis represents how correlated the time series is with itself at that lag. On APU01, the higher peak in correlation is located at the 1200 lag. It means the time series repeats every 1200 seconds, equivalent to 20 minutes. On APU02, the higher peak in correlation is located at the 700 lag. It means the time series repeats every 700 seconds, equivalent to 12 minutes. It confirms that APU01 and APU02 present a different periodicity.

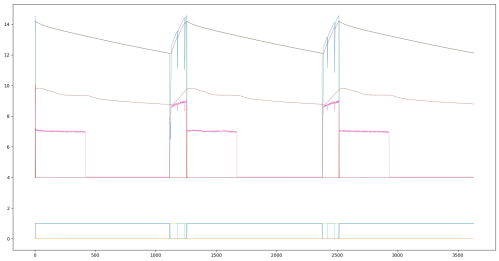


Figure 3.2: Periodicity APU01

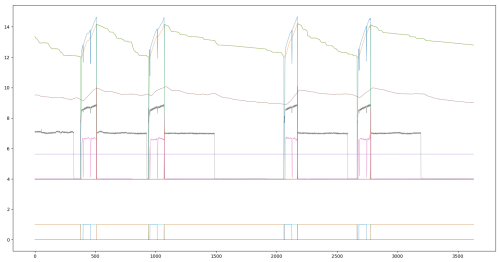


Figure 3.3: Periodicity APU02

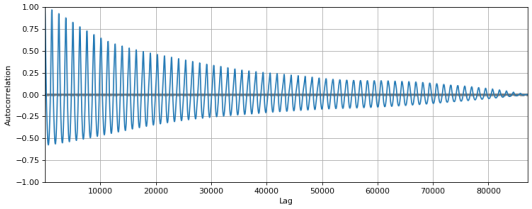


Figure 3.4: Autocorrelation APU01

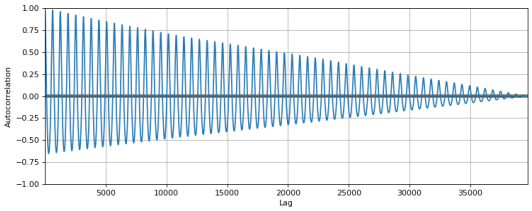


Figure 3.5: Autocorrelation APU02

Table 3.1: Sensors installed on the train's Air Production Unit

Sensor	Type	Unit	Behavior
TP2	Analogical	bar	Measures the pressure on the compressor generated due to loss of loads.
TP3	Analogical	bar	Measures the pressure generated in the pneumatic panel.
H1	Analogical	bar	Measures the pressure generated due to pressure drop when the discharge of the cyclonic separator filter occurs.
DV Pressure	Analogical	bar	Measures the pressure exerted due to pressure drop generated when towers are discharged.
Reservoirs	Analogical	bar	Measures the pressure that exists downstream the reservoirs, that should be close to the pressure on the pneumatic panel (TP3).
Motor Current	Analogical	A	Measures the current of one phase of the three-phase motor that should present values close to 4 amps when the compressor is turned off, close to 7 amps when the compressor is working off loaded and close to 8.8 amps when the compressor is working under load.
Oil Temperature	Analogical	ºC	Measures the temperature of the oil present on the compressor
Flowmeter	Analogical	m^3/h	Measures the air flow that leaves the APU for Reservoirs
COMP	Digital	-	Electrical signal of air intake valve on compressor.
DV Electric	Digital	-	The electrical signal of the solenoid valve A4 is active when the compressor is working under load.
TOWERS	Digital	-	Defines which tower is drying the air and which tower is draining the humidity removed from the air.
MPG	Digital	-	Is responsible for activating the COMP intake valve to start the compressor under load when the pressure in the APU is below 8.2 bar.
LPS	Digital	-	Is activated when the pressure is lower than 7 bars.
Pressure switch drying towers	Digital	-	Detects the discharge in the air drying towers.
Oil Level	Digital	-	Detects the level of the oil on the compressor and is active (equal to one) when the oil is below the expected values.
Caudal Impulses	Digital	-	Pulse output counter of the absolute amount of air passing from the APU to the reservoirs.

Table 3.2: Failures reported during the experimental period

APU	Date	Time	System	Problem
APU02	11-03-2020	20:00	-	-
APU01	12-03-2020	08:25	Vehicle	Strange noise
APU01	27-03-2020	-	-	-
APU01	18-04-2020	07:05	Pneumatic panel	Air leak
APU01	20-04-2020	17:20	Weight sensor	Failure
APU01	30-05-2020	01:10	Pneumatic panel	Air leak
APU01	05-06-2020	18:00	Pneumatic panel	Air leak
APU02	28-06-2020	14:58	Pneumatic panel	Pressure lower than 7 bar
APU01	15-07-2020	18:25	Pneumatic panel	Pressure lower than 7 bar
APU01	17-07-2020	05:46	Pneumatic panel	Air leak
APU02	28-07-2020	06:30	Weight sensor	Failure

Table 3.3: APU01 - Descriptive statistics of the numerical variables.

Variable	Min	Max	Mean	SD	CV(%)	Skewness	Kurtosis
TP2	-4	10.68	1.39	3.32	2.38	1.90	1.82
TP3	-4	10.31	8.39	1.15	0.13	-8.1	87.51
H1	-4	10.3	7.49	3.47	0.46	-1.72	1.22
DV Pressure	-4	10.28	0.05	0.48	10.04	-1.72	36.86
Oil Temperature	0	83.2	60.34	10.9	0.18	-5.72	53.05
Reservoirs	-4	10.31	8.39	1.15	-8.1	-0.67	87.51
Motor Current	0	11.62	5.55	1.98	0.34	0.4	-1.1

Table 3.4: APU02 - Descriptive statistics of the numerical variables.

Variable	Min	Max	Mean	SD	CV(%)	Skewness	Kurtosis
TP2	-4	10.92	0.97	2.96	3.04	2.46	4.53
TP3	-4	10.47	8.84	1.53	0.17	-6.35	4.80
H1	-4	10.45	7.45	3.55	0.48	-1.72	1.22
DV Pressure	-4	10.36	0.32	1.85	5.81	4.18	1.74
Oil Temperature	0	78.49	64.13	13.35	0.21	-5.83	4.55
Flowmeter	-4	15.41	0.23	0.89	3.85	1.09	7.54
Motor Current	0	11.68	5.8	1.83	0.32	-0.12	-0.04

Table 3.5: APU01 - Numerical variables dispersion and shape

Variable	Dispersion	Skewness	Tails of Distribution	Normal Distribution
TP2	Small	Positively	Tails heavier	No
TP3	Small	Negatively	Tails heavier	No
H1	Small	Negatively	Tails heavier	No
DV Pressure	Small	Negatively	Tails heavier	No
Oil Temperature	Small	Negatively	Tails lighter	Yes
Reservoirs	Small	Negatively	Tails heavier	No
Motor Current	Small	Positively	Tails lighter	No

Table 3.6: APU02 - Numerical variables dispersion and shape

Variable	Dispersion	Skewness	Tails of Distribution	Normal Distribution
TP2	Small	Positively	Tails heavier	No
TP3	Small	Negatively	Tails heavier	No
H1	Small	Negatively	Tails heavier	No
DV Pressure	Small	Positively	Tails heavier	No
Oil Temperature	Small	Negatively	Tails heavier	Yes
Flowmeter	Small	Positively	Tails heavier	No
Motor Current	Small	Negatively	Tails lighter	No

Table 3.7: Digital sensors frequency

Sensor	Value	APU01	APU02
COMP	1	83.75%	87.88%
	0	16.25%	12.12%
DV Electric	1	17.42%	12.12%
	0	82.58%	87.88%
TOWERS	1	91.87%	93.93%
	0	8.13%	6.07%
MPG	1	82.58%	87.88%
	0	17.42%	12.12%
LPS	1	0.14%	0.89%
	0	99.86%	0.2%
Pressure Switch Drying Towers	1	99.74%	0.0%
	0	0.26%	100%
Oil Level	1	98.63%	0.0%
	0	1.37%	100%
Caudal Impulses	1	79.8%	0.22%
	0	20.2%	99.78%

Chapter 4

Anomaly Detection

In this chapter, we will define rules for anomaly detection, considering the most common failures that were identified in the first phase of the project (Pinto, 2019) and the failures reported in Table 3.2.

The goal is to define a system to monitor the APU behavior and send notifications when the rules are triggered.

4.1 Anomaly Detection based on Sensors

Before the definition of rules, is important to distinguish the different states of the APU during the day. They are classified using GPS sensors. The analysis of anomaly detection is based on the moments when the train is in operation. Anomalies detected during the time the train is parked are not related to its operation and might follow different patterns, as described next.

Parked: the GPS coordinates and the speed of the train indicate that it is stopped at one of the parking stations.

In operation: the train is in operation, with GPS speed higher than 0 and GPS coordinates different from the parking station.

Under maintenance: the GPS coordinates of the train indicate that it is stopped at one of the maintenance stations.

Considering the previous knowledge about the compressor system, the pressure in TP3 should not be outside of the range of 7.5 bar and 10.5 bar,

and it can be an indicator of an anomaly. Figures B.3 and B.4 represent the number of samples of pressure TP3 outside the range in both APU01 and APU02 through different days. It is possible to verify that the days close to failure are highlighted. At APU02, we can highlight other days, beyond the days of failure.

When LPS is triggered, it means that the system is stopping, because the pressure is lower than 7 bar. Figures B.1 and B.2 represent the number of samples when the LPS is active, when the train is operating. It is possible to verify that the days close to failure are highlighted. As in the previous analysis, we can highlight other days, beyond the days of failure.

When the values of pressure in TP3 present a higher difference to the values of pressure in Reservoirs, it might be caused by an air leak on the compressor. However, this sensor is not working as expected on both trains, so we will not apply the rule on our data set.

In addition to those rules, the frequency of maximum peaks on the values registered by each sensor can be considered an effective way to verify the normal behavior of the compressor and detect anomalies. Comparing the figures in Appendix B, which represent the evolution of the sensors during "normal" days and failure days, some differences in the frequency of peaks can be reported. Those differences will be explored in the next section.

4.2 Peak Frequency Definition

In this section we will explore the evolution of the frequency of peaks over our data set. This analysis is based on the data collected from 1st of March until 30th of April. Define the normal number of peaks in a period, and monitor them in real-time might be a solution to detect anomalies. By the visual analysis of the values returned by each sensor through one day, we verified that all of them reach maximum and minimum peaks, as in Figure 4.1.

For the analogical sensors, this means that their values increase over some time until they reach a maximum peak. From there, they start to decrease until they reach a minimum peak, and so on. For the digital sensors, which only return two values, 0 or 1, the maximum peaks are reached when the sensor is active, and they last for a certain period until they are inactive again.

The frequency of the peaks can vary from day to day and during the day. Some of those variations are minimal and might be caused by small

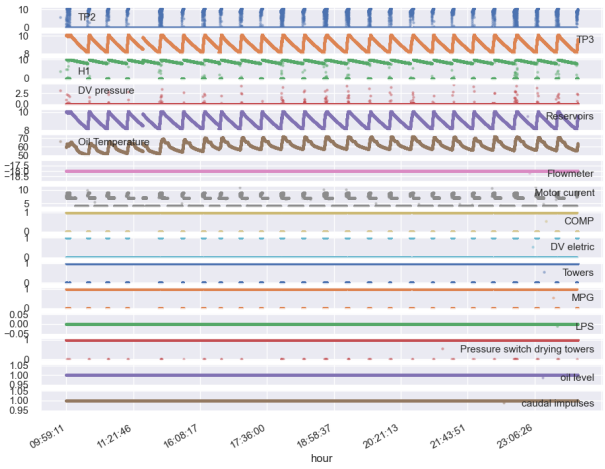


Figure 4.1: Evolution of the peaks during one day on APU01

perturbations on the environment where the machine is working. However, some of the variations are higher and are visible during the moments previous to failure.

To define the normal behavior of both APU01 and APU02, we will monitor periods of two hours. A period of two hours is enough, considering that the peak frequency changes at least on the day before the failure, as in Figures B.5 and B.6. The data is received from the sensors every five minutes, and the frequency of the previous two hours should be calculated every time it is received. This process allows us to analyze the peak frequency every time the data is received and predict failure as soon as possible.

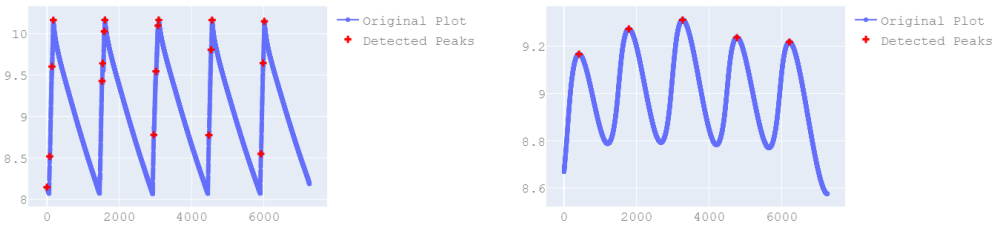
4.2.1 Low-pass Filtering

The sensor readings are very sensitive to environmental noise and perturbations, and the returned decimal values can also be noisy. The sensor data have an extensive range of values, and consecutive readings can have insignificant differences between them. The only peaks that should be considered on analogical sensors are the maximum values that they present in the specified period. We have tried different values of δ to ignore the noisy

values and to fit with our goal. However, we have noticed that this technique discards some significant peaks.

Low-pass filtering is a solution that is often applied to eliminate the effects of environmental noise. This approach eliminates the perturbation of the sensors reading. Thus, we have applied a low-pass filter on our data set, to attenuate the peaks and only consider the maximum peaks that are relevant to our study. In previous studies (Fridolfsson, Börjesson, Buck, Ekblom, Ekblom-Bak, Hunsberger, Lissner, and Arvidsson, 2019; Hook, 2014; Pereira, Ribeiro, and Gama, 2014a) low-pass filters have been applied before the analysis of the signals, to remove perturbation. This filter passes signals with a frequency lower than the selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The function *butter()* of the library *scipy* (Jones, Oliphant, Peterson, et al., 01) on Python, with $filter\_order = 1$ and $cutoff\_frequency = 0.001$ was used to apply the low pass filter to our data set. Those parameters were defined after a set of experiences and satisfy the needs of the data set.

The analogical peaks are counted as shown in Figure 4.2 for the TP3 sensor in a time window of two hours, with 7200 samples. Without the low pass filter, more than 10 peaks were identified. With the low pass filter, the real number of 5 peaks was identified, considering a peak when the sensor reaches a maximum value.



(a) without low-pass filter **(b)** with low-pass filter

Figure 4.2: Comparison of number of peaks in TP3

4.2.2 Peak Frequency Analysis

The analogical sensors are used to obtain the number of peaks in two hours. Every time the algorithm detects a peak, it means that the compressor turns off. The goal is to monitor how many times the compressor has turned off on a specific period. The developed algorithm for peak detection updates the maximum value when the difference between the value and the current local maximum is higher than δ, with $\delta = 0.01$. The algorithm developed to calculate the peak frequency on analogical sensors is described in Algorithm 1.

Algorithm 1 Algorithm to detect peaks on analogical sensors

```
sensor_readings = low_pass_filter(sensor_readings)
δ = 0.01
currentMax = -1
for value_t in sensor_readings do
  if value_t > currentMax - δ then
    currentMax = value_t
  end if
end for
```

The digital sensors will be used to obtain the peak duration, and determine how long the compressor was under load or off loaded. It happens when the sensor reading switch from 0 to 1 and 1 to 0. The algorithm is described in Algorithm 2.

Algorithm 2 Algorithm to detect the peak duration on digital sensors

```
for value_t in sensor_readings do
  if value_t == 0 and value_{t+1} == 1 then
    peak_start
  else if value_t == 0 and value_{t+1} == 1 then
    peak_end
  end if
end for
```

Next, we will analyze the results obtained when we calculate the peaks on our data set. We have analyzed our data set from the 1$^{\text{st}}$ of March until 30$^{\text{th}}$ of April. Then, we have removed the days of failure and days previous to

failure from the original data set, and calculated the mean number of peaks in two hours. The days before failure were analyzed individually. The results and the comparison is described on Tables B.1 and Tables B.2.

Normal peak frequency on analogical sensors

The normal number of peaks on a period of two hours can be characterized as in Table 4.1. The pressure on TP2, TP3, H1 and Reservoirs have a close number of peaks during two hours, on all the observed days. It means that those sensors are correlated, as expected.

Table 4.1: APU normal behavior based on peak frequency

APU	TP2	TP3	H1	DV Pressure	Oil Temperature	Motor Current
APU01	4	5	5	2	4	4
APU02	7	7	7	2	6	6

To understand how the distribution of peak frequency ranges, we have decided to analyze the peaks on TP3, because in Figure A.6, we can relate that TP3 reflects the behavior of most of the sensors. Also, in Table 3.2, most of the failures were reported on the pneumatic panel. From the analysis of both APUs, we can state that the mean number of peaks is higher on APU02 than on APU01. Figure 4.3 represents the mean number of samples of each value of peak frequency on APU01 and APU02 on a period of two hours. The x-axis represents the peak frequency on the TP3 sensor, and the y-axis represents the number of periods of two hours when that number of peaks was verified.

The number of peaks detected on APU01 varies between 2 and 9 and the number of peaks detected on APU02 varies between 2 and 12. On APU01, the mean number of 5 peaks have a higher frequency, comparing to the others. However, on APU02, the frequency of each number of is peaks more distributed, but it is possible to distinguish 6 and 7 as the mean number of peaks with higher frequency. This analysis reveals a different behavior on APU01 and APU02 operation modes.

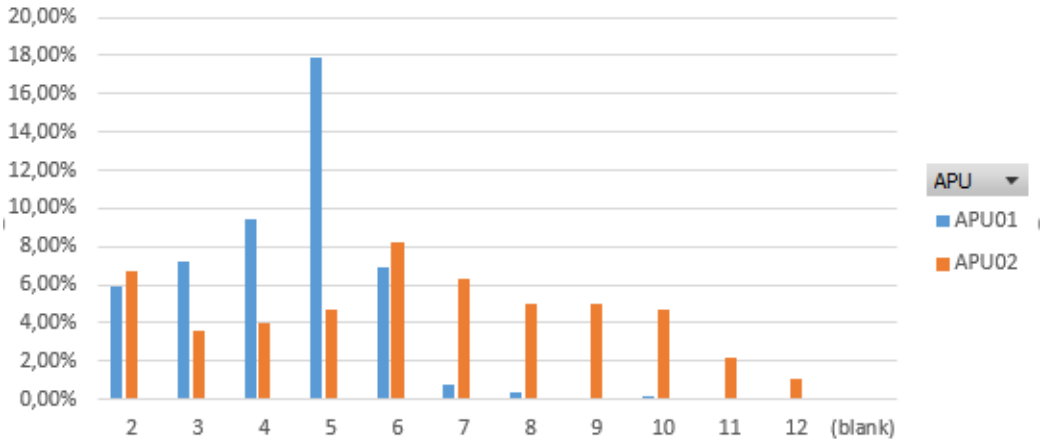
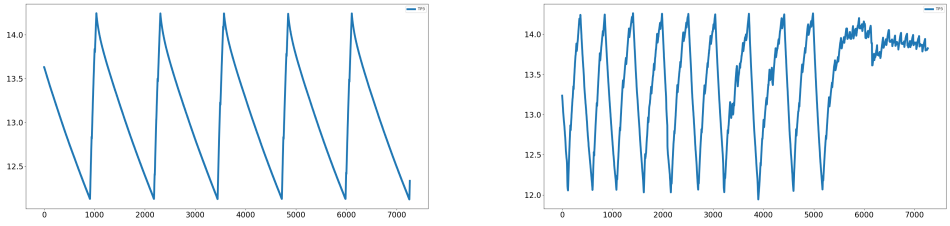


Figure 4.3: Frequency of the mean number of peaks on APU01 and APU02

Abnormal peak frequency on analogical sensors

On the days when a failure was reported, the differences in the peak frequencies are notable. Figures B.5, B.6 and B.7 represent evolution of the sensor measures on days of failure and days before failure. In some of those periods, a higher frequency of peaks is noticed. The visual comparison between the normal and abnormal work of the sensor TP3 on a period of two hours in Figure 4.4 is enough to conclude that TP3 presents a different behavior when the APU is about to fail. The plot that represents the normal period on APU01 presents 5 peaks, and the plot that represents the abnormal period on APU02 presents more than 10 peaks.



(a) Normal work **(b)** Abnormal work

Figure 4.4: TP3 sensor reading comparison

Normal peak duration on digital sensors

A different approach was used to analyze the behavior of the digital sensors of the train. The digital sensors can only assume two values, and they can be used to understand for how long a peak was reached. Also, using the digital sensors, it is possible to calculate the time the compressor works under load or off loaded and its normal duration for each APU.

Table 4.2 represent the mean duration of the peaks on digital sensors, for APU01 and APU02, respectively.

Table 4.2: Peak duration on the digital sensors

	COMP/MPG	DV Electric	TOWERS
APU01	22 minutes	2 minutes	23 minutes
APU02	12 minutes	2 minutes	13 minutes

The analysis of the duration of peaks has reinforced the idea that APU01 working mode is different from APU02 working mode. On APU02, the compressor is off or working off loaded for a shorter period, which means that the pressure rises more often and, consequently, more peaks are reached.

On COMP and MPG sensors, the peak is reached when there is no air on the compressors. The peaks on DV Electric are reached when the compressor is working under load and lasts for 2 minutes. The digital signal TOWERS gives us information about the duration of the peak when the compressor is off or working off loaded.

Table 4.3 summarizes the conclusions taken about the working modes of the two compressors installed on different trains. It contains the average time each APU is working under load and off loaded or turned off. Comparing the two compressors, it is possible to relate that most of the sensor readings on APU01 have a defined behavior when the compressors are off loaded for a period close to 20 minutes, and under load for a period close to 2 minutes. The APU02 components have a defined behavior, where the compressors are off loaded for a period close to 13 minutes and under load for a period close to 2 minutes. This analysis is according to the autocorrelation analysis in Section 3.3, where we concluded that on APU01 the time series repeats every 1200 seconds (20 minutes), and on APU02 the time series repeats every 700 seconds (12 minutes).

Table 4.3: APU normal behavior based on peak duration

APU	Working under load (DV = 1)	Working off loaded or turned off (COMP=1)
APU01	≈2 minutes	≈22 minutes
APU02	≈2 minutes	≈13 minutes

Abnormal peak duration on digital sensors

To understand how the peak duration changes when failure happens, we will analyze the previous days of the known failures. The digital sensors are highly correlated, namely, by their peak duration. We will analyze the sensor COMP, which represents the working modes of the compressor, and that is positively correlated with MPG and negatively correlated with DV Electric, as we concluded in Section 3.3. When it is active, it means that the compressor is turned off and when it is inactive, it means that the compressor is under load.

The results of the duration of the peaks when the compressor is under load on APU01 and APU02 are represented in Table B.4 and Table B.6, respectively. It is possible to conclude that the time when the compressor is working under load increases when the failure is close.

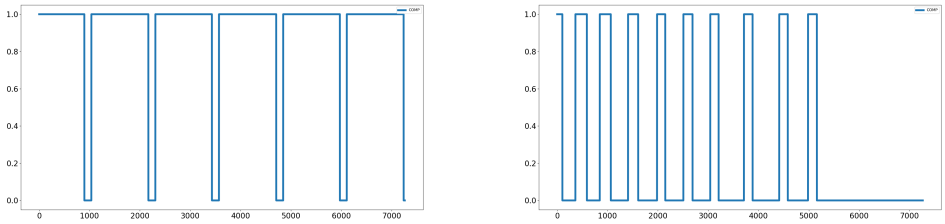
The results of the duration of the peaks when the compressor is off loaded on APU01 and APU02 are represented in Table B.3 and Table B.5, respectively. The time when the compressor is working off loaded decreases when a failure is close. This means that the sensor COMP can reach more peaks with an higher duration when the failure is close.

Figure 4.5 presents the comparison between the normal and abnormal work of the sensor COMP on a period of two hours. From the analysis of this figure is possible to conclude that the sensor COMP presents a different behavior when the APU is about to fail. On the plot that represents the normal period, the sensor is active for large periods and inactive for small periods. On the plot that represents the abnormal period, the sensor is active for short periods and inactive for long periods.

By the analysis of the duration of the peaks on the COMP sensor on failure days, we can distinguish some common patterns:

- the compressor can be off for shorter periods, and consequently the compressor is turned on with more frequency than the usual;

- the compressor can be off for more time than expected (more than ≈22 minutes on APU01 and more than ≈13 minutes on APU02);
- the compressor works under load for higher periods than expected (more than ≈2 minutes).



(a) Normal work **(b)** Abnormal work

Figure 4.5: COMP sensor reading comparison

4.2.3 Rules for Anomaly Detection

Based on the knowledge acquired in the previous section, about peak frequency and peak duration, we can now define the rules for anomaly detection. The analysis of the sensors behavior on the days when a failure occurred, and the previous days, was important on the definition of those rules.

On the days previous to a failure, the mean number of peaks has suffered some deviations and some extreme values such as 15 peaks per period or no peaks per period. The period when the compressor was working under load has also increased until periods of 5 minutes.

Considering the analysis of the peak frequency, the rules that trigger an anomaly are the following.

Rule 1 - Period of 2 hours with less than 2 peaks in TP3.

Rule 2 - Periods of 2 hours with more than 10 peaks detected in TP3.

Rule 3 - Periods when the compressor is working under load for more than 5 minutes with COMP = 0.

Rule 4 - Periods when the compressor is working off loaded for more than 40 minutes with COMP = 1.

Considering the knowledge about the compressor system, the rules that should trigger an anomaly are the following.

Rule 5 - LPS activated, meaning that the pressure on the APU is lower than 7 bar.

Rule 6 - Pressure TP3 outside of the range of 7.5 and 10.5 bar.

The definition of rules for anomaly detection consists on the first phase of this work, helping to understand how the system works in normal conditions and how it fails, based on the frequency of peaks.

As we described in Section 3.1, the data is sent to the server every five minutes using the TCP/IP [1] protocol. Every time the data is received, the server runs a script that verifies if the rules were triggered and sends an email with the description of the behavior of the APU in the previous two hours. This is a useful way of preventing failures on the system because the team who is in charge of the working system will be alerted when an anomaly is detected.

4.3 Experimental Evaluation

In this section, we will run the rules previously defined over our evaluation data set and verify if they are triggered more frequently on days of failure. Also, we will understand how early those rules were triggered, to predict the failure as soon as possible. A sliding window, that starts from the first two hours of the 1st of March and is incremented by five minutes until 31st of July will be implemented. The goal is to simulate the online working of the system, which sends data to the server every five minutes.

APU01

The number of times the rules were triggered on APU01, is represented in Figure 4.6, with the failure days, surrounded by a red circle. The increasing number of rules triggered before the failure days is evident when comparing

[1] a protocol suite used to transfer data on the internet.

to the other days. It is possible to verify an increasing number of rules at least one day before a failure happens.

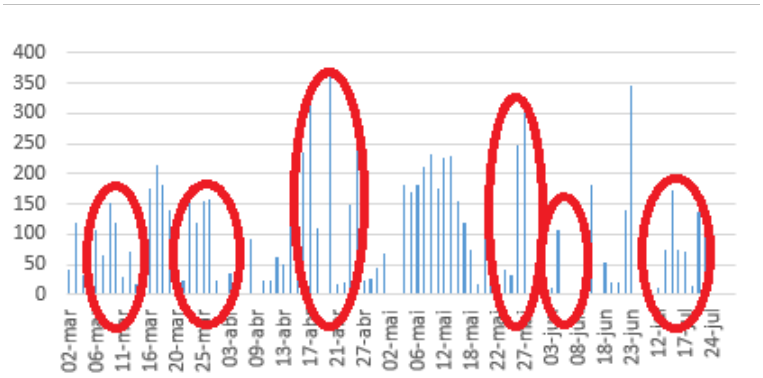


Figure 4.6: Plot with the number of rules triggered by day on APU01 with failure days surrounded

The graphs that represent the number of times that each rule was triggered on APU01 are represented in Appendix B. On APU01, Rules 1 and 2 were only triggered in a few days. Rules 3, 4, and 6 were triggered more times, comparing to the others. However, Rule 6 was triggered in fewer days comparing to Rule 4 and Rule 3. Rule 5, was only triggered when the compressor stopped, and when the pressure decreased below 7 bar. Based on this, we concluded that Rules 1, 2, and 6 are more restricted than the others.

APU02

The number of times the rules were triggered on APU02, is represented in Figure 4.7, with the failure days, surrounded by a red circle. The increasing number of rules triggered is only evident in the first month of operation. This might be an indicator of an hidden failure or anomaly.

The graphs that represent the number of times that each rule was triggered on APU02 are represented in Appendix B. In general, the rules were trigged on APU02 fewer times than on APU01. Also, they are focused on the beginning of March, when no failure was reported.

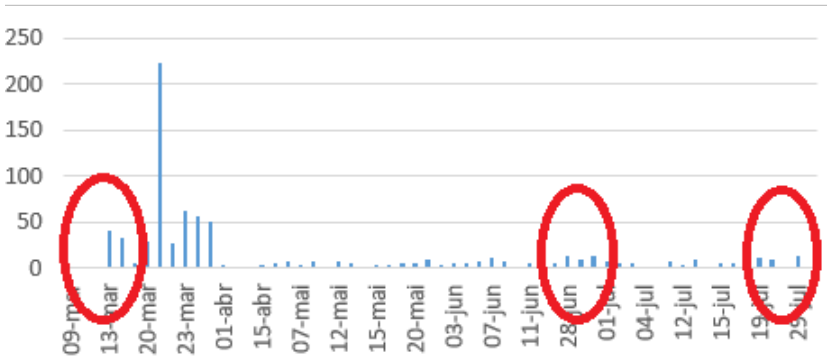


Figure 4.7: Plot with the number of rules triggered by day on APU02 with failure days surrounded

Rule evaluation

In this section, we will evaluate each of the defined rules as a failure prediction approach, based on the known system failures. This evaluation is based on the data collected from 1st of March until 31st of July. The rules will be evaluated based on the percentage of Predicted Failures and the False Alarms.

We found a huge difference in the results between the two APU. The number of times each rule was triggered on failure days was visibly on APU01 in Figure 4.6. On APU02, most of the rules were triggered in March, when no failure was reported. On days of failure, the rules were triggered a few times and it suggests possible unreported failures on APU02. In Table A.1, it is possible to verify that the moments when the train went to maintenance on APU02 are aligned with the days when most of the rules were triggered. This proves that APU failure prediction based on rules is more effective on APU01. On APU02, the rules are useful to detect anomalies but they are not correlated with failures of the compressor system.

The Table 4.4 represents the percentage of of each rule, on both APU01 and APU02. A Predicted Failure is considered if the rule was displayed at least 3 days before failure happen. A False Alarm is considered when the rule does not have a subsequent failure.

On APU01, Rules 3 and 6 were able to predict almost all failures, with a low percentage of False Alarms, comparing to the others. The other rules

predicted the same percentage of failures, but only Rule 2 presents a small false alarm rate. Based on this analysis, we conclude that Rule 2, 3, and 6 are the best rules to predict failures with few false alarms. This idea is reinforced by the results presented in Table 4.5, which shows which failures were predicted by each rule. It is possible to verify that those rules were able to predict all the failures that have occurred on APU01.

Table 4.4: Rules evaluation on APU01 and APU02 based on Predicted Failures and False Alarms

	Predicted Failures		**False Alarms**	
	APU01	APU02	APU01	APU02
Rule 1	37.5%	66%	79%	71%
Rule 2	37.5%	100%	42%	88%
Rule 3	87.5%	0%	37%	100%
Rule 4	37.5%	66%	90%	96%
Rule 5	37.5%	66%	92%	75%
Rule 6	75%	66%	50%	88%

Table 4.5: Rules triggered as prediction of each reported failure

	F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**
Rule 1	X		X	X				
Rule 2	X						X	X
Rule 3	X	X	X	X		X	X	X
Rule 4	X	X	X					
Rule 5			X				X	X
Rule 6	X		X	X	X	X	X	

Chapter 5

Clustering

In this chapter, we will test different clustering algorithms to characterize the different working modes of the train. Then, we will test the outlier detection based on clustering as a failure prediction approach.

5.1 Learning the Normal Behavior

In the previous section, we detected significant differences between the behavior of APU01 and APU02. We will consider the data received from both separately, to analyze the differences between them in terms of how they are divided in clusters, characterizing their different working modes. This analysis will be based on the data collected from 1st of March until 30th of April.

Clustering is a type of unsupervised learning, which is used with unlabeled data sets that do not have defined categories or groups, just like real-time data that is received from sensors. It consists of dividing the data points into groups, where the data points in the same groups are more similar to each other and different from other group's data.

The main idea behind using clustering for anomaly detection is to learn the normal behavior in the data available and then using this information to point out if one point is anomalous or not when new data is received (Syarif, Prugel-Bennett, and Wills, 2012).

If there is a data evolution over time, the model should learn the new behavior, to avoid that this new behavior is classified as anomalous. When dealing with time series data, and the problem is that a value is increasing

too fast, but still in a normal range of values, clustering is not enough to deal with this problem. Because of that, we can say that clustering and anomaly detection, presented in Chapter 4, are complementary methods to deal with anomaly detection. Clustering looks at a sample individually and verifies if it is anomalous. Many consecutive anomalous samples are an indicator of failure.

The clustering analysis will be performed in three different data sets with different variables included (Chapter 4):

Data set 1 (APU01) - Data received from the sensors installed on APU01, except the ones that are not working as expected: Flowmeter, Caudal Impulses, Oil Level;

Data set 2 (APU02) - Data received from the sensors installed on APU02, except the ones that are not working as expected: Reservoirs, Pressure switch drying towers;

Data set 3 (APU01+APU02) - Data received from the sensors installed on both APU01 and APU02, except the ones that are not working as expected on at least one of them: Flowmeter, Reservoirs, Caudal Impulses, Pressure switch drying towers.

5.1.1 K-Means

K-Means (MacQueen, 1967) is the best-known and most commonly used clustering algorithm. Since we are still on a preliminary phase of the project, and the objective is to know the data set and learn how to deal with it, we have decided to use K-Means to distinguish the different working modes on the compressor system and characterize them.

The algorithm K-Means was performed to understand how the data is divided into clusters and to characterize each cluster. The K-Means algorithm is a typical partition-based clustering algorithm where the data is divided into several predefined sets. K-Means algorithm involves an interactive and intuitive approach to know as expectation-minimization. This approach consists of:

1. randomly guess some clusters centers;
2. assign points based on the closest cluster center (expectation step);
3. set the cluster centers to the mean (minimization step).

The most important advantage is the simplicity and speed of this method, so it can be applied to large data sets. However, the algorithm may not

produce the same result in each run and cannot handle outliers. Before performing the K-Means algorithm, the optimal number of clusters should be calculated. As long as the number of clusters increase, the differences between clusters are smaller and the differences between samples inside clusters also increase. The goal is to find a balance point in which the samples in a group are homogeneous and the clusters are different from one another.

Silhouette analysis (Rousseeuw, 1987) measures how close each point in a cluster is to the points in its neighboring clusters. Silhouette values lie in the range of between -1 and 1. The higher the value, the better is the cluster configuration. To define the optimal number of cluster based on silhouette analysis, the following points should be considered:

1. the mean value should be as close to 1 as possible;
2. the plot of each cluster should be above the mean value as much as possible;
3. lastly, the width of the plot should be as uniform as possible.

Table 5.1 shows that the silhouette value is closer to one when the number of clusters is equal to 2.

Table 5.1: Silhouette analysis to find the optimal number of clusters

Clusters	APU01	APU02
N=2	0.76	0.78
N=3	0.69	0.56
N=4	0.68	0.55
N=5	0.62	0.54

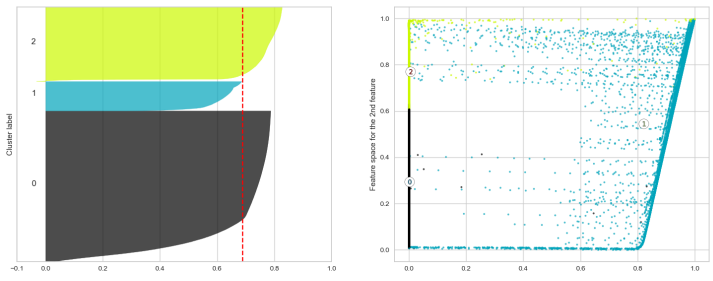


Figure 5.1: Silhouette analysis APU01

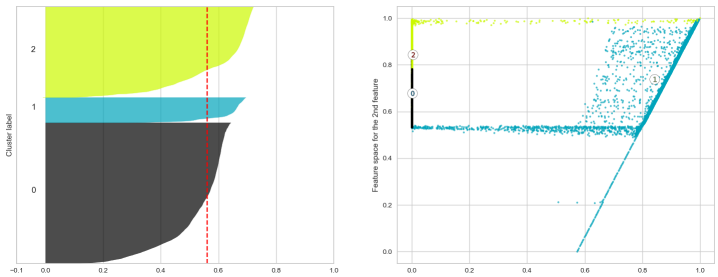


Figure 5.2: Silhouette analysis APU02

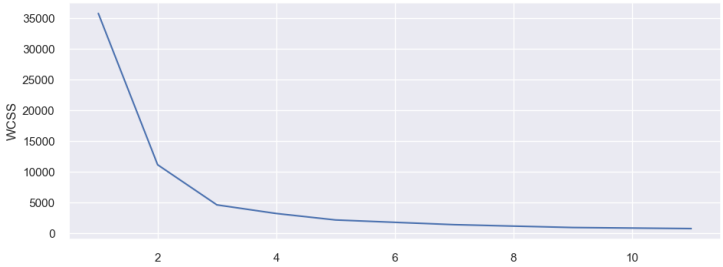


Figure 5.3: Elbow method APU01

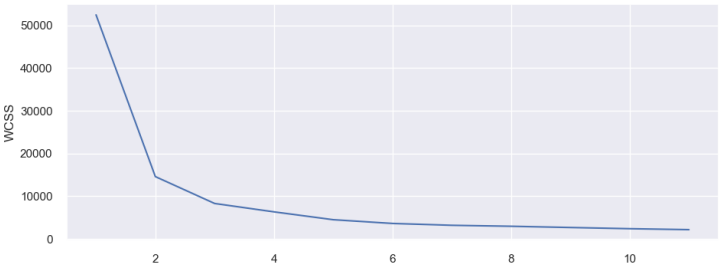


Figure 5.4: Elbow method APU02

However, as we can see in Figures 5.1 and 5.2, the plot is more uniform when the number of clusters is equal to 3. The left pic depicts a sorted list of Silhouette analysis clusters of each point in a given cluster. The right pic is the visualization of the cluster assignment. Using the Elbow method (Thorndike, 1953), we have obtained an optimal number of clusters equal of 3 for both APU01 and APU02, where the curve is accentuated, as we can see in Figures 5.3 and 5.4. After the analysis to find the optimal number of clusters, based on the Silhouette analysis and the Elbow method, the optimal

number of clusters defined for each APU was 3. The centers of the three data sets in the analysis were calculated using K-Means with: k=3. The cluster centers of Data set 1, Data set 2 and Data set 3 are represented in Tables 5.2, 5.3, 5.4, respectively.

Table 5.2: K-Means Centers APU01

TP2	TP3	H1	DV Press.	Reservoirs	Oil Temp.	Motor Current	COMP	DV Electric	TOWERS	MPG	LPS	Drying Towers	Oil Level
0.38	8.69	8.24	-0.02	8.69	55.46	4.34	0.95	0.05	0.97	0.95	0.00	1.00	1.00
0.00	9.50	9.49	-0.02	9.50	65.57	6.16	1.00	0.00	1.00	1.00	0.00	1.00	1.00
9.04	9.15	0.00	0.58	9.14	68.59	8.66	0.00	1.00	0.50	0.00	0.00	0.99	1.00

Table 5.3: K-Means Centers APU02

TP2	TP3	H1	DV Press.	Oil Temp.	Flowmeter	Motor Current	COMP	DV Electric	TOWERS	MPG	LPS	Oil Level	Caudal Impulses
0.13	8.58	7.11	1.21	57.28	0.25	4.28	0.97	0.03	0.99	0.97	0.00	0.00	0.00
0.01	9.21	9.17	-0.02	69.58	0.32	6.09	1.00	0.00	1.00	1.00	0.00	0.00	0.00
9.52	9.15	0.00	0.02	67.26	11.98	8.70	0.01	0.99	0.50	0.01	0.00	0.00	0.01

Table 5.4: K-Means Centers APU01 and APU02

TP2	TP3	H1	DV Press.	Oil Temp.	Motor Current	COMP	DV Electric	TOWERS	MPG	LPS
0.00	9.31	9.30	-0.02	68.05	6.10	1.00	0.00	1.00	1.00	0.00
0.25	8.70	8.00	0.36	55.98	4.34	0.96	0.04	0.98	0.96	0.00
9.12	9.11	0.00	0.33	67.82	8.66	0.01	0.99	0.50	0.01	0.00

Comparing the two compressors it is possible to observe some similarities and differences between them. The values of pressure (TP2 , TP3 , and H1) are slightly higher on APU01, and the Oil Temperature is higher on APU02.

The compressor knows three different ways of working:

1. Off - when the compressor is off with no charge (motor current close to 4A).
2. Off Loaded - when the compressor is working with no charge (motor current close to 7A).
3. Under load - when the compressor is working with charge (motor current close to 8.8A).

Based on the the analysis of the cluster centers, it is possible to identify the three working modes of the compressor. The analysis of the tables allows us to characterize each cluster and understand the normal working of the compressor system. The plots generated by the algorithm are represented in Appendix C.

Cluster 1

- Motor current close to 8A, meaning that the **compressor is working under load**.
- Highest values of pressure on the compressor (TP2).
- Highest values of pressure on the pneumatic panel (TP3).
- Pressure on H1 equal no 0, with no discharge of the cyclonic filter.
- Higher values of DV Pressure.
- Highest, MPG - Inactive (Working under load).
- DV Electric - Active (Working under load).

Cluster 2

- Motor current close to 6A, meaning that the **compressor is working off loaded**
- No pressure on the compressor (TP2).
- The maximum value of on the pneumatic panel (TP3).
- Highest values of H1.
- Lowest values of DV Pressure.
- Highest Values of Oil Temperature.
- COMP, MPG , TOWERS - Active (Off mode or off loaded).
- DV Electric - Inactive (Off mode or off loaded).

Cluster 3

- The minimum value of motor current, meaning that the **compressor is turned off** (close to 4A).
- No pressure on the compressor (TP2).
- The minimum value of pressure on the pneumatic panel (TP3).
- Highest values of H1.
- Highest values of DV Pressure.
- Lower Values of Oil Temperature.

- COMP, MPG , TOWERS - Active (Off mode or off loaded).
- DV Electric - Inactive (Off mode or off loaded).

The other sensors assume different behaviors and are characterized below:

- LPS sensor is inactive in every cluster, as expected. This sensor is only activated when the pressure is below 7.5 bar and the APU is about to fail.
- Oil Level sensor was installed in reverse on APU01, and its behavior should not be considered on clustering. On APU01, all the clusters present the sensor as active and on APU02, all the clusters present the sensor as inactive.
- TOWERS sensor is active or inactive, with its center equal to 0.5 when the compressor is working under load. It is active when tower 1 is drying the air and tower 2 is draining humidity and it is inactive if tower 2 is drying the air and tower 1 is draining humidity.

K-Means clustering algorithm was useful to distinguish the different working modes of the train and characterize them. To overcome the fact that it does not consider outliers, DBSCAN clustering will be used.

5.1.2 DBSCAN

DBSCAN (Ester, Kriegel, Sander, and Xu, 1996) is a density-based clustering approach, based on the concepts of density and attraction of objects. The idea is to create clusters of dense multi-dimensional areas where objects attract each other. It is used with noisy data sets, captures complex shapes of clusters and detects outliers as a by-product. It works by identifying points that are in crowded regions of the feature space, where many data points are close together. This algorithm faces some challenges, including failure to find clusters of varied densities(Pierobon, 2018). The steps to the DBSCAN algorithm are:

1. pick a point randomly that has not been assigned to a cluster and compute its neighborhood to determine if it is a core point; if it is, start a cluster around this point; if it is not, label the point as an outlier;
2. once it finds a core point, and thus a cluster, expands the cluster by adding all directly-reachable points to the cluster; it performs "neighborhood jumps" to find all density-reachable points and add them to

the cluster; if an outlier is added, change that point's status from outlier to border point;
3. repeat these two steps until all points are either assigned to a cluster or designated as an outlier.

The biggest advantage of the density-based clustering to our work is the fact that it considers all the data points, even the outliers. This is why the number of clusters of the data received from the trains will naturally be higher using a density-based approach.

The DBSCAN algorithm requires two parameters (do Prado, 2017):

1. *eps*: to specify how close points should be to each other to be considered a part of a cluster; it means that if the distance between two points is lower or equal to this value, these points are considered neighbors.
2. *minPoints*: the minimum number of points to form a dense region; for example, if we set the *minPoints* parameter as 5, then we need at least 5 points to form a dense region.

DBSCAN was implemented using Python's ML library *sklearn* (Pedregosa et al., 2011). Before the implementation of the model, the data was normalized using the MinMaxScaler() function from Python. Another thing that should be considered when implementing DBSCAN is the distance metric. Euclidean distance will be used since it is faster than the other metrics. It should be only used with continuous variables, so we will use the analogical sensor features only.

One of the biggest challenges that when using the DBSCAN algorithm is the dimension of the data set and the computational memory needed to perform the algorithm. We have tested different approaches to understanding what kind of data should be used to perform the algorithm and the values to set the parameters. Also, it is important to decide the size of the data set, because the larger the data set, the larger the *minPoints* value that should be chosen. To avoid memory issues and high computational effort, we have decided to randomly select 20K data points and perform the algorithm from them. The algorithm will run 30 times and the results will be grouped in the end.

The algorithm gives us the optimal number of clusters, the number of noisy points, and the silhouette coefficient, which is based on the percentage of data points in each cluster. We have tested different combinations of *eps* and *minPoints* to obtain the highest silhouette coefficient possible.

The *eps*=0.05 and *minPoints*=100 reveal the best combination with a silhouette coefficient close to 0.75. The results obtained are presented in Table 5.5

Table 5.5: DBSCAN Clustering Results

Data set	Nr. of Clusters	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
APU01	4	∼8%	∼83%	∼8%	∼1%	-
APU02	5	∼5.5%	∼88%	∼5.5%	∼0.5%	∼0.5%
APU01 + APU02	5	∼7%	∼85%	∼7%	∼0.7%	∼0.3%

DBSCAN assigns points to clusters and return the labels of clusters. If it cannot assign the value to any cluster (because it is an outlier), it returns -1. Depending on the context, the outliers might be considered anomalous points.

- Values of TP3 lower than 7.5 bar when TP2 is higher than 0.
- Values of in Reservoirs lower than 7.5 bar.
- Values of DV Pressure higher than 10 bar.
- Values of H1 between 0 and 7 bar, with motor current between 6 and 9 A.

5.1.3 Cluster Characterization

In this section, we will do a full characterization of the sensors, based on the information collected by K-Means and DBSCAN algorithms. The three data sets defined earlier will be distinguished and compared. The clusters are characterized in Table 5.6. The behavior of the sensors installed on APU01 is the same as the one described in the analysis of K-Means. The only difference of the Cluster 4 is the sensor TOWERS. It means that tower 1 is drying the air and tower 2 is draining humidity 8% of the time and tower 2 is drying the air and tower 1 is draining humidity 8% of the time. The same happened on Data set 2 (APU02) and Data set 3 (APU01 + APU02) with this sensor. However, in a lower portion of the time. This new cluster was identified by the DBSCAN algorithm.

The Cluster 5 represents an anomalous behavior of the compressor, that is happening enough times to be considered a cluster. It happens when there is no pressure on the compressor, meaning that it is off or working at no load. The DV Pressure and the pressure on the compressor (TP2) is close

to 0, as expected, but the pressure TP3 and H1 are the lowest. Also, the LPS sensor is active, indicating an anomaly on the compressor. Some of the sensors values are according to the outlier characterization defined on the previous section.

Table 5.6: Cluster characterization using DBSCAN

Data set	Sensor	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
APU01	TP2	>8 bar	≈0 bar	>8 bar	≈0 bar	
	TP3	≈9 bar	≈9 bar	≈9 bar	>9 bar	
	H1	≈0 bar	≈9 bar	≈0 bar	>9 bar	
	DV Pressure	≈0 bar	>0 bar	≈0 bar	≈0 bar	
	Oil Temperature	>65ºC	≈60ºC	>65ºC	>65ºC	
	Flowmeter	-	-	-	-	
	Reservoirs	≈9 bar	≈9 bar	≈9 bar	≈9 bar	
	Motor Current	8A	4A	8A	7A	
	COMP	0	1	0	1	
	MPG	0	1	0	0	
	DV Electric	1	0	1	1	
	TOWERS	0	1	1	1	
	Drying Towers	1	1	1	1	
	LPS	0	0	0	0	
APU02	TP2	≈9 bar	≈0 bar	≈9 bar	≈0 bar	≈0 bar
	TP3	≈9 bar	≈9 bar	≈9 bar	<8 bar	<8 bar
	H1	≈0 bar	≈9 bar	≈0 bar	≈0 bar	≈0 bar
	DV Pressure	≈0 bar	>0 bar	≈0 bar	≈0 bar	≈0 bar
	Oil Temperature	>65ºC	>65ºC	>65ºC	≈40ºC	≈40ºC
	Flowmeter	≈11 bar	≈0 bar	≈11 bar	≈0 bar	≈1 bar
	Reservoirs	-	-	-	-	
	Motor Current	8A	7A	8A	4A	4A
	COMP	0	1	0	0	0
	MPG	0	1	0	0	0
	DV Electric	1	0	1	1	1
	TOWERS	0	1	1	0	1
	LPS	0	0	0	0	1
APU01 + APU02	TP2	≈0 bar	≈9 bar	≈9 bar	≈0 bar	≈0 bar
	TP3	≈9 bar	≈9 bar	≈9 bar	>9 bar	<8 bar
	H1	≈9 bar	≈0 bar	≈0 bar	>9 bar	≈0 bar
	DV Pressure	≈0 bar	>0 bar	>0 bar	≈0 bar	≈0 bar
	Oil Temperature	>60ºC	>65ºC	>65ºC	>65ºC	≈40ºC
	Motor Current	4A	8A	8A	7A	4A
	COMP	1	0	0	1	0
	MPG	1	0	0	0	0
	DV Electric	0	1	1	1	1
	TOWERS	1	1	0	1	1
	LPS	0	0	0	0	1

5.2 Experimental Evaluation

In the previous section, we have applied clustering techniques into our data set to characterize the normal and abnormal working of the APUs based on sample analysis. In this section, we will evaluate DBSCAN clustering as an anomaly detection approach, considering its ability to detect outliers. We will try to find correlations between the number of outliers found and the reported failures. The known failures at this point are identified in Table 3.2. This analysis will be based on the data collected from 1st of March until 31st of July.

APU01

The number of outliers found in each day, on APU01 is represented on Figure 5.5. The days of failure and days previous to failure are surrounded by a red circle. Notably, most of the outliers were found on the days of failure, with a growing number of outliers found on the days previous to the failure. The number of outliers found in days of failure and days close to failure is clear, comparing to the days without failure. This is important to achieve our goal, which is to predict failures as soon as possible before a failure happens.

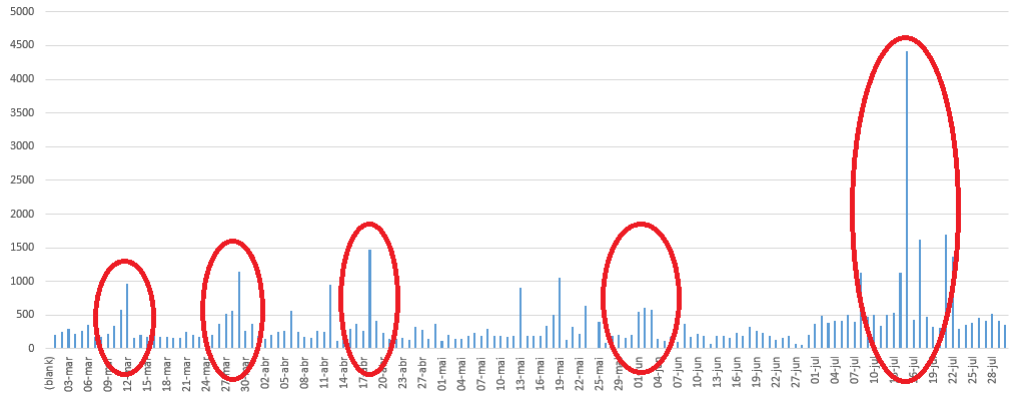


Figure 5.5: Number of outliers on APU01 by day using DBSCAN

APU02

The number of outliers found in each day, on APU02 is represented in Figure 5.6. The days of failure and days previous to failure are surrounded by a red circle. When making the same experimental evaluation on APU02, the results are not so satisfactory. This reinforces the idea of Chapter 4, that APU02 might have anomalies on its operation that are not associated with the APU. Also, the focus of the abnormal points is in the first month of working, as it was verified in Section 4.3.

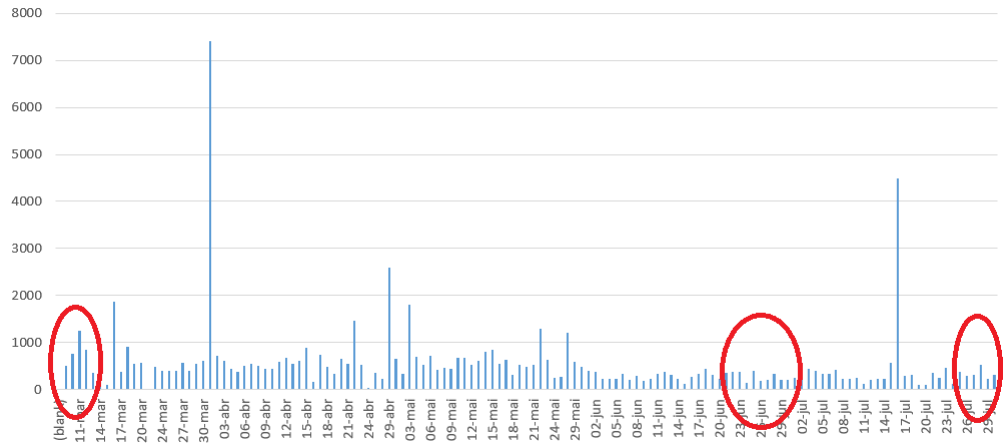


Figure 5.6: Number of outliers on APU02 by day using DBSCAN

Using DBCAN, the number of anomalies on days of failure were exceeded by other days when no failure has occurred, on APU01. On APU02, the days when most outliers were detected, using DBSCAN clustering, are associated with the days when the LPS was triggered, as in Figure B.2. It means that although they are not associated with a failure, they might be associated with a system anomaly.

Based on this, we conclude that in addition to the identification of the normality modes of the compressor, the clustering approach can be used to predict failures. An increasing number of outliers is a reliable way to warn the maintenance team about a possible failure.

Now that we have confirmed that most of the outliers found in the data set were associated with the days of reported failures, we will evaluate outlier detection based on DBSCAN clustering as a failure prediction approach and

identity how soon the failure can be predicted.

A sliding window, that starts from the first two hours of the 1st of March and is incremented by five minutes until 31st of July will be implemented, considering the data received by APU01. The goal is to simulate the online working of the system, which sends data to the server every five minutes. For the analysis the DBSCAN algorithm with *eps*=0.05, *metric*=Euclidean and *min_samples*=100 will be used.

The Table 5.7 presents the periods when most of the outliers were found during the experimental period. It only considers periods when half of the data set (3100 samples) were identified as outliers, as a collective anomaly (Chandola et al., 2009). Also, the analysis only considers the periods when the train is operating.

The following parameters will be used to evaluate the algorithm:

- N - The number of periods when the alarm was raised.
- Mean - The mean number of anomalies found during the two hours in N observations.
- Root Cause - The root cause identified to explain the number of anomalies.
- t before failure - The time that has elapsed since the anomaly was identified by the DBSCAN until it was reported.

Using the DBSCAN clustering to detect outliers, we were able to predict 50% of the failures reported on APU01, at least 3 hours in advance. The same approach was used on APU02 and the results are represented in Table 5.8. The clustering algorithm was able to predict 50% of the failures that occurred when the vehicle was operating, but only 1 hour before it was reported. The first anomaly identified by the clustering approach was not associated with any failure or anomaly reported and can be considered a false alarm. To conclude this chapter, we will use the the percentage of failures detected and false alarms to evaluate clustering as a failure prediction approach. The outlier detection based on clustering was able to detect 50% of the failures of the APU and 50% of the alarms were considered a false alarm. Based on this, we concluded that an increasing number of outliers might be a good indicator of failure. Also, the failures were predicted at least 3 hours before reported on APU01 and 1 hour on APU02. However, some of the failures are

immediate and are not possible to predict by the evolution of the sensor's values.

Table 5.7: Anomalous periods based on outlier detection on APU01

Anomaly Reported	N	Mean	Root Cause	t before failure
From: 12-03-2020 00:45 To:12-03-2020 08:25	40	7200	Failure at 8AM	7 hours
From: 12-04-2020 12:00 To: 12-04-2020 21:30	145	7200	-	-
From: 18-04-2020 04:10 To: 18-04-2020 07:25	32	7200	Failure at 7AM	3 hours
From: 13-05-2020 09:00 To: 13-05-2020 22:55	87	7113	-	-
19-05-2020 09:05	3	3326	-	-
From: 20-05-2020 04:40 To: 20-05-2020 17:45	153	7200	-	-
From: 05-06-2020 09:55 To: 05-06-2020 18:00	98	7195	Failure at 6PM	8 hours
From: 15-07-2020 06:05 To: 15-07-2020 15:20	220	4642	Failure at 6PM	12 hours

Table 5.8: Anomalous periods based on outlier detection on APU02

Anomaly Reported	N	Mean	Root Cause	t before failure
29-04-2020 10:45	2	5988	-	-
28-06-2020 13:50	3	6173	Failure at 7PM	1 hour

Chapter 6

Deep Learning

In this chapter, we will explore Deep Learning techniques to be applied to our Predictive Maintenance problem.

After the analysis of abnormal patterns, the next step in the predictive maintenance cycle (Coleman et al., 2017) is the application of ML techniques to make predictions. In our study, we have decided to explore Deep Learning techniques. The use of Deep Learning in PdM maximizes equipment lifetime, operational efficiency, and uptime. The process is known as "training", enables the DL algorithms to detect anomalies and test correlations while searching for patterns across the various data feeds (Gonfalonieri, 2019).

6.1 Data Preprocessing

In this section, we will introduce the data preprocessing steps that should be applied to our data set before applying DL models. Data preprocessing is necessary to convert the data set into a form able to extract condition indicators. This is often a challenging and time-consuming process, as we have mentioned before in Section 3.1.2.

Data cleaning

Usually, the first step of data preprocessing is to clean the data set, detecting missing, and duplicate data. Our data set does not contain any missing data and the duplicated data should only be removed when the algorithm does not consider the temporal information. Another important

preprocessing step is outlier detection. However, they are useful to detect anomalies, so they were kept on the data set.

Removing features of low importance can improve accuracy, and reduce model complexity and overfitting. The feature selection involved two steps:

1. eliminate the sensors that were not installed on both APU, to make a consistent analysis;

2. eliminate one of the two features that are highly correlated between them, with correlation higher than 0.9; this will reduce the size of the data set and avoid misleading results of precision and recall.

From step 1, the features Flowmeter, Reservoirs, Caudal Impulses and Pressure switch drying towers were removed. From step 2, the features MPG and DV Electric were eliminated. The features TP2, Oil Temperature, and H1, also reveal a high correlation between them. However, the differences between them might be a good indicator of failure and we decided to keep them on our data set.

Data normalization

Machine learning algorithms are usually very sensitive to the input data and the data needs to be in the range of -1 to 1 or 0 to 1. Thus, the data must be normalized with MinMaxScaler() function from *sklearn* (Pedregosa et al., 2011) library in Python.

Data dimension

Some DL algorithms, such as Neural Networks require the input data to be in 3-Dimensional form. In this case, the data set needs to be transformed from 2-Dimensional data to 3-Dimensional data. The 2-Dimensional data set (*samples* $\times$ *features*), should be transformed into a 3-Dimensional data set (*samples* $\times$ *lookback* $\times$ *features*). The *samples* represent the number of observations and the *features* represent the number of features present in the input data. The *lookback* represent the time t the LSTM will process past data up to *t-lookback* to make a prediction.

6.2 Modeling

In this section, we will introduce the models that fit our problem of predicting the future based on historical data.

The activation of the sensor LPS, indicates that the pressure on the compressor is below 7 bar and that the train stopped or is about to stop. At this point, the activation of the sensor LPS is the most meaningful event to predict.

We focus our work on Deep Learning algorithms, that has proven its efficiency on time series data. Also, DL algorithms can automatically extract the right features from the data, eliminating the need for manual feature engineering (Nimesh Sinha, 2018). The availability of Deep Learning APIs [1], such as *Keras* (Gulli and Pal, 2017) and *TensorFlow* (Abadi, Barham, Chen, Chen, Davis, Dean, Devin, Ghemawat, Irving, Isard, et al., 2016), have made model building and experimentation easier. However, it is important to understand the fundamentals of the model to obtain the best results. When predicting future values on the data set, different approaches can be considered:

Univariate input - Univariate input prediction should be used when one variable is varying over time, such as data collected from a single sensor.

Multiple input - This is a multivariate time series forecasting problem, where the output series is separated but dependent upon the input time series.

6.2.1 Long Short Memory Term

Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are a special kind of Recurrent Neural Networks (RNN), capable of learning long-term dependencies. They are usually applied to time series data by making it possible to look back for longer periods to detect failure patterns. LSTM can be used with both numerical data and binary data (0 or 1).

It was popularized by many researchers, working well on the large variety of problems.

The key to LSTM is the cell state, that runs through the top of the diagram, as in Figure 6.1. It is described step-by-step below:

[1] Application Programming Interface

Step 1 - Decide what information will be thrown from the cell state. Outputs 0 or 1 when 1 means to keep the information and 0 means to delete the information.

Step 2 - Decide what new information is going to be stored in the cell state.

Step 3 - Update the old cell state, into the new cell state.

Step 4 - Decide the output. This output will be based on our cell state but will be a filtered version.

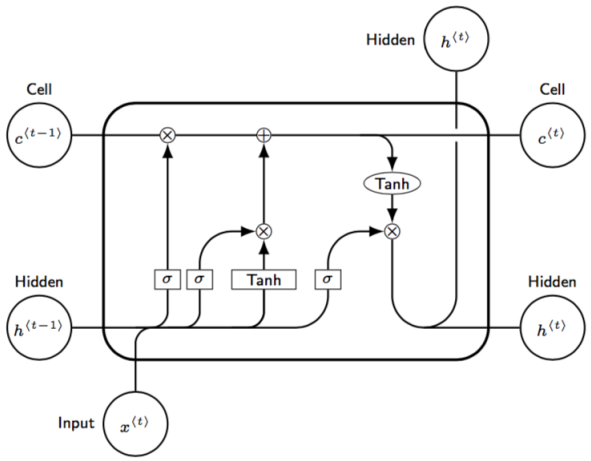


Figure 6.1: Illustrative image of a Long Short Term Memory Network (Sagheer and Kotb, 2019)

For time series, it is important to maintain temporal information in the data so the LSTM network can learn patterns from the correct sequence of events. Therefore, it is important not to shuffle the data when creating test and validation sets and also when fitting the model.

6.2.2 Autoencoders

Autoencoders (Ballard, 1987) are a type of self-supervised learning model that can learn a compressed representation of input data. They are usually used when the data set is imbalanced, with more positively labeled samples than negative, such as a typical rare-event problem, where the positively labeled data is around 5% of the total (Baldi, 2011).

The Autoencoder approach for classification is similar to anomaly detection: learning the pattern of a normal process. Anything that does not

follow this pattern is classified as an anomaly. It contains three components Figure 6.2.

Input layer - The data is inputted using an input layer of size p.

Encoder - Learns the underlying features of a process. The size of the encoding layer is k. When k $\geq$p, it leads to an over-representative model, and consequently $\approx$0 reconstruction error.

Decoder - Recreates the original data from these underlying features. In a decoder, the data is reconstructed from the encoding.

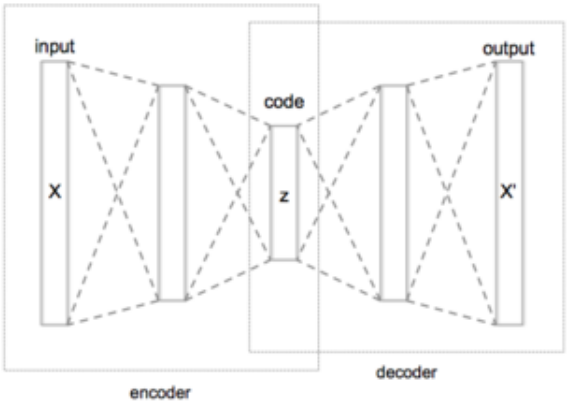


Figure 6.2: Illustrative image of an Autoencoder structure (Chervinskii, 2015)

The data is divided into two parts: positively and negatively labeled. The negatively labeled data is treated as a normal state of the process and is used to train the Autoencoder, while the positively labeled data will be ignored. A well-trained Autoencoder predicts the data that comes from the normal state of the process and shares the same pattern or distribution. The other observations, that does not follow the normal state, will have an higher reconstruction error.

6.2.3 LSTM Autoencoder

LSTM and Autoencoders can work together to extract the temporal features from the data set, which is important on failure prediction (Rama and

Çöltekin, 2016).

The LSTM Autoencoder consists of two parts:

LSTM Encoder - Takes a sequence and returns an output vector (return_sequences = False)

LSTM Decoder - Takes an output vector and returns a sequence (return_sequences = True)

In the end, the encoder is a many to one LSTM and the decoder is a one to many LSTM . Also, the LSTM Autoencoder is composed by different layers (Ranjan, 2019b):

1. **LSTM (x)** - Reads the input data and outputs x features with y timesteps for each because return_sequences=True.
2. **LSTM (z)** - Takes the $y \times x$ input from Layer 1 and reduces the feature size to $1 \times z$. Since return_sequences=False, it outputs a feature vector of size z.
3. **RepeatVector** - Repeats its input n times, because the LSTM does not return sequences and it is the easiest way to turn this output into a sequence of the same time-steps of the model input. Acts as a bridge between the encoder and decoder modules.
4. **TimeDistributed** - Used to mix RNN layers with other kind of layers.

The process is similar to the Autoencoders described before: Build an Autoencoder on the normal (negatively labeled) data, use it to reconstruct a new sample, and if the reconstruction error is high, it is labeled as a break.

6.3 Experimental Evaluation

In this section, we will test the possibility to predict the activation of the LPS sensor, which represents a pressure below 7 bar on the compressor, and it means that the train is about to stop. To predict the activation of the sensor LPS, a multivariate approach will be used. We will compare the performance of Autoencoders and LSTM Autoencoders.

6.3.1 Experimental Setup

The cause of a rare event, as the failure of the APU, is usually instant. The failures of the APUs, represented in Section 3.1.1 have different causes.

Some of them lead to the degradation of the system until the failure, and others are instant. In an extremely rare event problem, we have less than 1% positively labeled data, such as the activation of the LPS sensor.

In Section 2.3.2, we found that DL performs better than ML, if the data is sufficient. Also, to implement ML, the data set should be balanced and undersampled from negatively labeled data. However, the undersampling will result in a data set that is ≈1% of the size of the original data, ignoring the information present in the remaining ≈99% of the data (Ranjan, 2019a). This is why we choose an approach based on Deep Learning to failure prediction.

The use of Autoencoders can be a well rated approach to predict the activation of the sensor LPS, which is positively labeled only on 0.005% of our data set. We decided to compare the performance of Autoencoders with LSTM Autoencoders approach because the second considers the temporal information and might have a better performance.

The following steps were taken to model both Autoencoder and LSTM Autoencoder.

Step 1 - To start building the model, we have experimented to shift the LPS variable by different n rows, corresponding to *n/3600* minutes. The goal is to keep the "failure" status for a while and also, to prevent the classifier from learning to predict a failure after it has already happened. The following steps should be performed:

1. delete the consecutive rows that are positively labeled;
2. find the first row r positively labeled, and made the rows (r-n) (r-(n-1)) and so on, equal to 1;
3. delete the row r. This will help the classifier learn up to n rows ahead prediction;
4. shift the labels up to n rows up.

Step 2 - Drop the temporal and categorical features

Step 3 - Split the data into train, valid and test sets. The following steps should be performed:

1. split the data set into train set (70%) and test set (30%);
2. split the train set into train set (70%) and valid set (30%);

3. split each of the train, valid and test sets by negatively labeled and positively labeled;
4. split each of the train, valid and test sets by X (other features) and Y (LPS).

Step 4 - Normalize the data set. Normalize the entire data after split into train-test, is important to keep the test data completely unseen to anything during the modeling.

Step 5 - Train the model with the negatively labeled data only.

- number_epochs = 20
- batch_size = 72

6.3.2 Performance Metrics

On a classification problem, the most common performance metrics to be applied are (Ghoneim, 2020):

Accuracy - Ratio of the correctly labeled subjects to the whole pool of subjects. It is not a good evaluation metric when dealing with unbalanced data sets.

Precision - Precision is the ratio of the correctly labeled samples to all samples, given by the Equation 6.1.

Recall - Recall is the ratio of the correctly labeled samples to all real positive samples, given by the Equation 6.2.

F1 Score - Considers the balance on both precision and recall.

ROC/AUC - The ROC (Receiver Operating Characteristic) curve is a probability curve and AUC (Area Under Curve) represents the degree of separability. It shows the capability of the model to distinguish between classes and measures the quality of the model's predictions.

$$precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{6.1}$$

$$recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{6.2}$$

On a binary classification problem with class labels 0 and 1, with a threshold of *th*, the predicted values less than the threshold *th* are assigned to class 0, and values greater than or equal to *th* are assigned to class 1.

The threshold selection is based on the precision and recall values, and the goal is to obtain a good balance between the two measures, using the information of Figure 6.3. The True Positive and False Positive will be defined by the dots above the threshold line in Figure 6.4. This threshold value can be optimized considering the ROC Curve or the relation between precision/recall.

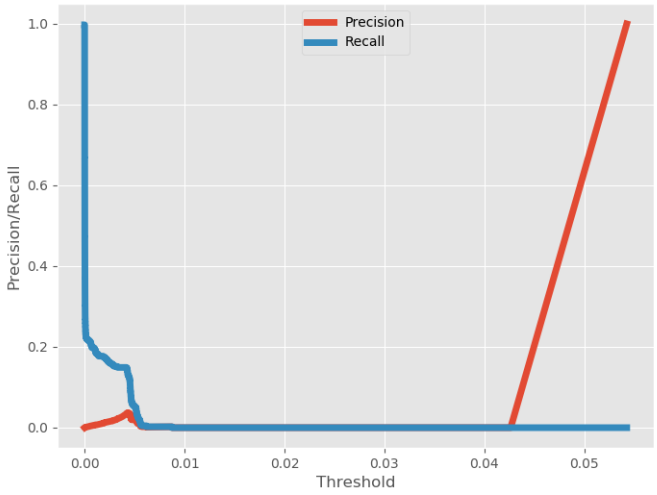


Figure 6.3: Precision and recall plot for each threshold

The AUC can be calculated based on probabilities or based on the predictions. The function *auc(falsepositive_rate, truepositive_rate)* from the library *sklearn* (Pedregosa et al., 2011) in Python calculates the AUC based on the bobabilities and does not depend on the threshold. The function *roc_auc_score(actual, predicted)* from the same library calculates the AUC based on the predictions and depends on the threshold.

The AUC measures the quality of the model's predictions independently of the threshold. It describes the discriminate power of a classifier independent of class distribution and was chosen as the best evaluation model to our problem. The ROC and AUC measures are represented as in Figure 6.5. The number of True Positives and False Positives is obtained by the confusion matrix. A true positive is an outcome where the model correctly

predicts the positive class and a false positive is an outcome where the model incorrectly predicts the positive class. The structure of the confusion matrix is represented in Table 6.1.

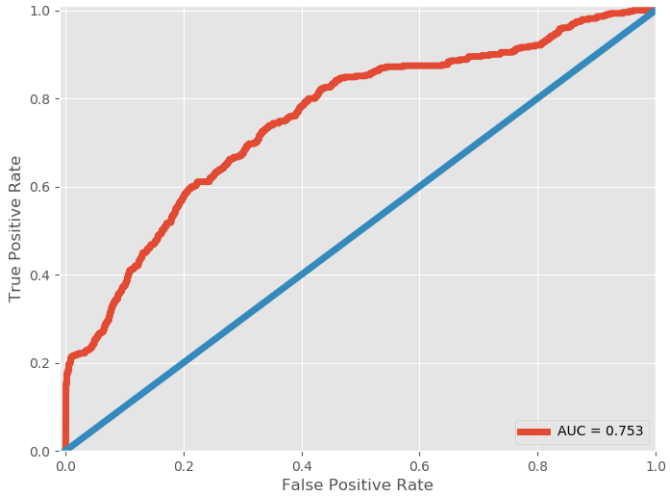


Figure 6.4: Reconstruction error line based on the defined threshold

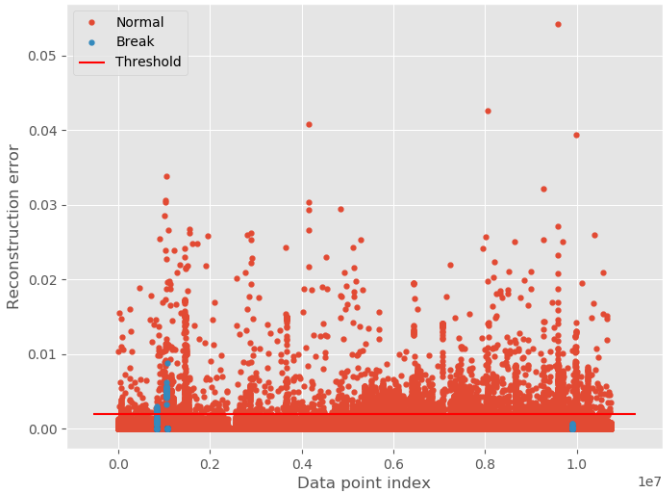


Figure 6.5: ROC curve and AUC

Table 6.1: Confusion Matrix structure

	Actual Positive	Actual Negative
Predicted Positve	True Positve (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

6.3.3 Obtained Results

In this section, we will present the results obtained when comparing Autoencoders with LSTM Autoencoders.

To train the model, the selected features were: TP2; TP3;H1; DV Pressure; Oil Temperate; Motor Current. The data set contains the data received from 1st of March until 31st of March. In the previous chapters, we have concluded that APU01 and APU02 have different working and failure modes. Based on this, we will evaluate their performance separately and together.

On this first experiment, the data set was shifted for 300 rows, equivalent to 5 minutes. For both, we have used Adam [2] (Kingma and Ba, 2014) optimizer.

For Autoencoders, the encoding dimension used was 6, the number of features on our data set. For LSTM Autoencoders, the 2D array was transformed into a 3D array. This should be done after the transformation to avoid losing the initial rows of samples up till the *lookback* [3]. The first selected *lookback* was 10 rows, equivalent to 10 seconds. The results obtained are represented on Table 6.2.

Table 6.2: Comparison between the AUC of Autoencoders and LSTM Autoencoders

	Autoencoders	LSTM Autoencoders
APU01	0.604	0.583
APU02	0.636	0.594
APU01+APU02	0.651	0.631

The first experiment has proved that, the use of Autoencoders, when compared to the use of LSTM Autoencoder with *lookback*=10 rows, presents

[2]Stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

[3]Time t the algorithm will process past data to make a prediction

a better AUC, on at least 2%, with the maximum difference of 4.2% . Also, the Autoencoders proved to be way faster when training the algorithm. Based on this, increase the *lookback* to improve the results will be more time and memory consuming. For that reason, we will proceed with the study of Autoencoders.

This first experiment was made with the data from the month of March only. Now, is important to understand how the results evolve when the size of the training set increase. The goal is to verify if the results improve when the model learns with new negatively labeled data. The results obtained are represented on Table 6.3.

Table 6.3: Evolution of the AUC of Autoencoders over time

	March	March to April	March to May	March to June	March to July
APU01	0.604	0.607	0.670	0.570	0.604
APU02	0.636	0.568	0.576	0.551	0.517
APU01+APU02	0.651	0.647	0.604	0.574	0.589

If we compare the results obtained by APU01 and APU02 we conclude that the higher AUC was achieved by the model that was trained with the data received from APU01. It means that the classifier is better at distinguishing the normal and abnormal behavior on APU01. This reinforces the idea of Chapters 4 and 5, that the degradation of the system before a failure is more evident on APU01 and that its failures are more predictable.

In some cases, the evolution of the results shows that the model was able to learn and achieve better results when it has more training data. However, it does not present a perfect increase rate, suggesting that some data have "confused" the model and the AUC results decrease. This is expected, because the activation of the LPS does not always mean that the APU of the train has failed. It might be related to the failure of other components of the train that does not affect the APU sensors values. Even if it is related to an APU failure, it can be a sudden failure that does not cause system degradation.

Since the goal is to predict failures on the APU and not on the train as a whole, we have decided to eliminate the noise on our data set. To do this, we added a new feature, that will only consider the activation of the sensor LPS on days of failure, and we will try to predict the new feature. The variable LPS was then removed from the data set.

We will train the model using the negatively labeled data to predict if it is capable to predict the APU fault in the next 5 minutes. The model will consider all the data from 1st of March until 31st of July and consider all the failures that happened in this period. The data from APU01 and APU02 will be first divided and then analyzed together, to compare the results.

The results obtained for different threshold values are represented in Table 6.4. Is contains the AUC based on probabilities, the percentage of True Positives and False Positives and the ROC/AUC score based on predictions.

Table 6.4: APU failure prediction results

	AUC	Threshold	TP	FP	ROC/AUC
APU01	0.753	0.0001	60%	68%	0.59
		0.001	20%	0.6%	0.58
		0.01	0%	0.002%	0.4997
		Best Threshold = 0.0063	0%	0.002%	0.4995
APU02	0.672	0.0001	33%	1,4%	0.66
		0.001	20%	1%	0.497
		0.01	0%	0.001%	0.4997
		Best Threshold = 0.0011	0%	0.001%	0.493
APU01 + APU02	0.580	0.0001	40%	37%	0.517
		0.001	0.6%	0.4%	0.501
		0.01	0.2%	0.001%	0.500
		Best Threshold = 0.025	0%	0.0004%	0.499

Based on the results, it is possible to define the best AUC values were obtained when the APUs were analyzed separately.

The experiments with different thresholds are important to understand that the number of failures predicted vary according to the value defined. However, when the number of predicted failures increase, the number of false alarms also increase. The Best Threshold value is defined based on the index of the the largest F1-Score, that considers the balance between precision and recall. However, none of them were able to predict failures, but obtained the lowest false alarm rate.

Based on this, we conclude that the model is able to predict failures at least 5 minutes before they happen, depending on the threshold. The threshold value should then be defined considering the cost of a failure and the cost of a false alarm.

Chapter 7

Conclusions

The objective of the project FailStopper is to predict failures on the APU, a component installed in each of the existing Eurotram trains and that is vital to its operation. For that purpose a total of 16 sensors were installed in the APU of two trains: APU01 and APU02.

The goal of this dissertation was to understand how to predict failures on the APU. Our work was developed at the first stage of the project. We analyzed the different sensors readings and how they behave individually and together. We then characterized the abnormal behaviorand evaluated the possibility of predicting failures using DL.

7.1 Contributions

The contributions of this study are three-fold: (i) peak frequency analysis to characterize normal/abnormal behavior of the sensor readings; (ii) use clustering techniques to detect normal and outlier working modes of the APU; and, (iii) use of Autoencoders to failure prediction.

We introduced the peak frequency as an analytical approach to characterize the normal and abnormal behavior of the compressor. We calculated the mean number of peaks on the analogical sensors in a period of two hours. They represent the number of times the compressor turns off. Here, we have concluded that APU01 and APU02 present distinct operation modes, and APU02 presented more peaks than APU01. The analysis of the duration of the peaks has confirmed this deviation. From this analysis, we concluded that APU01 is working in off mode for periods close to 22 minutes and APU02 is

working in off mode for periods close to 13 minutes. After this, characterized the failures, based on its peak frequency and peak duration. This analysis allowed us to create a notification system based on 6 Rules. When one of the rules is triggered, the system send an email to alert about a possible failure. We proved that those six rules were able to predict all the failures that occurred on APU01. From those rules, we have distinguished Rule 2, 3 and 6, with the best ratio between True Positives and False Positives. Rule 2 is about periods of two hours with more than 10 peaks detected, Rule 3 is about periods when the compressor is working under load for more than 5 minutes, and Rule 6 is about samples with pressure on the pneumatic panel outside of the range of 7.5 and 10.5 bar. Based on this, we concluded that the failures might have different sources and different types, but all of them cause perturbations on the sensor peaks. It makes the analysis of the peak frequency and peak duration a well-rated approach to predict APU failures.

The second approach was to cluster our data set. The goal was to characterize the different working modes of the APU, and test outlier detection based on clustering as a possible way to detect failures on the APU. We started by finding the optimal number of clusters using the Elbow method and Silhouette analysis together. Then, a K-Means (MacQueen, 1967) approach was used to characterize the normal working of the APU. We found that the APU knows three different normality modes: when the compressor is working under load, when the compressor is working off and working off loaded. The normality modes were characterized using K-Means on APU01, APU02 and on the full data set that contains data from both. Once more, some differences between the behavior of APU01 and APU02 were reported. DBSCAN (Ester et al., 1996), a density-based clustering approach that considers outliers, was tested to reinforce the conclusions obtained with K-Means and to characterize the outliers on the data set. Using DBSCAN, we found an increasing number of outliers on the data set when the failure was approaching. The method was able to predict 50% of the failures, at least one hour before they happen. It means that outlier detection based on clustering is only able to predict some of the failures on the APU. Most of those failures occurred on the pneumatic panel. Based on this, we conclude that some of the failures are not identifiable by the presence of abnormal sensor readings. It also explains the success obtained by the rules based on peak frequency and peak duration, that do not consider samples of data, but the variations of its values.

The third approach is based on the comparison of Deep Learning methods

to predict failure on the APU. We compared the use of Autoencoders (Ballard, 1987) and LSTM Autoencoders (Rama and Çöltekin, 2016) to predict failure 5 minutes before it happen. The approach based on Autoencoders has proved to make a better distinction between the normal and the abnormal behavior of the system. Also, it was more efficient at the computational level, which is important when dealing with time series. On the experiments, the model was able to predict 60% of the APU01 failures and 33% of the APU02 failures. Also, the model obtained better results when training the two APUs separately.

7.2 Limitations

One of the limitations of our work was the fact that the sensors were only installed on two trains. The APU02 exhibited a distinct behavior since the beginning and that might be the cause of the nonconformity on the experimental evaluations on this APU. Since we only have two APUs to compare, it becomes harder to understand which is the deviant behavior and which is the "normal" one. Thus, it became difficult to adjust the characterization of the abnormal behavior to the two trains.

Another limitation of this work is the fact that we do not have all the sensors installed and working as expected on both trains. We ended up missing the opportunity to find abnormalities on those sensors and lose potential relationships with failure.

Also, due to the Covid-19 pandemic situation, there were some periods when trains were not operating. This caused a reduction in the amount of data from the trains in operation, but also in the amount of failures for analysis.

7.3 Future Work

In this work, we have proved that the peak frequency is a well-rated approach to anomaly detection, and capable of identify degradation patterns on the data set. This approach should be improved, based on the future data and future failures. Also, some Machine Learning models should be tested here, to adapt the rules based on the data that is received.

The prediction of the minimum peaks on the TP3 sensor with ML should

be explored, since it was proved to be one of the most reliable ways to predict failure. Also, Online Learning should be applied to this problem, to adapt the trained models to new types of failure that will certainly occur in the future.

A suggestion to the future, with more data and different types of failures available, is to categorize the failures by their root cause. This should be helpful to distinguish the best methods to use in each case.

Bibliography

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283.

Allah Bukhsh, Z., A. Saeed, I. Stipanovic, and A. G. Doree (2019). Predictive maintenance using tree-based classification techniques: A case of railway switches. *Transp. Res. Part C Emerg. Technol. 101*(February), 35–54.

Alom, M. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electron. 8*(3).

Amer, M., M. Goldstein, and S. Abdennadher (2013). Enhancing one-class Support Vector Machines for unsupervised anomaly detection. *Proc. ACM SIGKDD Work. Outlier Detect. Descr. ODD 2013*, 8–15.

Badnjević, A. (2019). Evidence-based clinical engineering: Machine learning algorithms for prediction of defibrillator performance. *Biomed. Signal Process. Control 54*.

Baldi, P. (2011). Autoencoders, unsupervised learning and deep architectures. UTLW'11, pp. 37–50. JMLR.org.

Ballard, D. H. (1987). Modular learning in neural networks. pp. 279–284.

Benedetti, M., F. Leonardi, F. Messina, C. Santoro, and A. Vasilakos (2018, 05). Anomaly detection and predictive maintenance for photovoltaic systems. *Neurocomputing 310*.

Bengtsson, M. (2004, oct). *Condition Based Maintenance Systems–An investigation of technical constituents and organizational aspects*. Ph. D. thesis.

Borghesi, A., A. Bartolini, M. Lombardi, M. Milano, and L. Benini (2019). A semisupervised autoencoder-based approach for anomaly detec-

tion in high performance computing systems. *Eng. Appl. Artif. Intell. 85*(June), 634–644.

Brauer, D. C. and G. D. Brauer (1987). Reliability-centered maintenance. *IEEE Transactions on Reliability R-36*(1), 17–24.

Chandola, V., A. Banerjee, and V. Kumar (2009). Survey of Anomaly Detection. *ACM Comput. Surv. 41*(3), 1–72.

Chervinskii (2015). Schematic picture of an autoencoder architecture. `https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png`. Online; accessed 29 July 2020.

Coleman, C., S. Damodaran, M. Chandramouli, and E. Deuel (2017). Making maintenance smarter. `https://www2.deloitte.com/us/en/insights/focus/industry-4-0/using-predictive-technologies-for-asset-maintenance.html`. Online; accessed 29 April 2020.

Dickey, D. A. and W. A. Fuller (1979). Distribution of the estimators for autoregressive time series with a unit root.

D'mello, A. (2019). The top 3 challenges of preparing iot data. `https://www.iot-now.com/2019/02/08/92826-top-3-challenges-preparing-iot-data/`.

do Prado, K. S. (2017). How dbscan works and why should we use it? `https://towardsdatascience.com/how-dbscan-works-and-why-should-i-use-it-443b4a191c80`. Online; accessed 30 May 2020.

Ester, M., H.-P. Kriegel, J. Sander, and X. Xu (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pp. 226–231. AAAI Press.

Fernández-Delgado, M., E. Cernadas, S. Barro, and D. Amorim (2014). Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res. 15*, 3133–3181.

Fridolfsson, J., M. Börjesson, C. Buck, Ö. Ekblom, E. Ekblom-Bak, M. Hunsberger, L. Lissner, and D. Arvidsson (2019, May). Effects of frequency filtering on intensity and noise in accelerometer-based physical activity measurements. *Sensors 19*(9), 2186.

Fumeo, E., L. Oneto, and D. Anguita (2015). Condition based maintenance in railway transportation systems based on big data streaming analysis. *Procedia Comput. Sci. 53*(1), 437–446.

Ge, M., X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly (2019a). Deep learning-based intrusion detection for iot networks. In *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing*

(PRDC), pp. 256–25609.

Ge, M., X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly (2019b). Deep learning-based intrusion detection for iot networks. pp. 256–25609.

Ghoneim, S. (2020). Accuracy, recall, precision, f-score and specificity, which to optimize on? `https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124`. Online; accessed 11 September 2020.

Gonfalonieri, A. (2019). How to implement machine learning for predictive maintenance. `https://towardsdatascience.com/how-to-implement-machine-learning-for-predictive-maintenance-4633cdbe4860`. Online; accessed 29 July 2020.

Goodman, D. L., J. Hofmeister, and R. Wagoner (2015). Advanced diagnostics and anomaly detection for railroad safety applications: Using a wireless, IoT-enabled measurement system. *AUTOTESTCON (Proceedings) 2015-Decem*(March 2017), 273–279.

Gulli, A. and S. Pal (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Hale, J. (2019). Scale, standardize, or normalize with scikit-learn. `https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02`. Online; accessed 20 April 2020.

Havinga, P. J. M. (2010). Ensuring high sensor data quality through use of online outlier detection techniques Yang Zhang *, Nirvana Meratnia and. *7*(3).

Hochreiter, S. and J. Schmidhuber (1997, November). Long short-term memory. *Neural Comput. 9*(8), 1735–1780.

Hook, J. (2014). Smoothing non-smooth systems with low-pass filters. *Physica D: Nonlinear Phenomena 269*, 76 – 85.

Jiawei Han, M. K. and J. Pei (2012). *Data Mining: Concepts and Techniques*.

Joanes, D. N. and C. A. Gill (1998). Comparing measures of sample skewness and kurtosis. *Journal of the Royal Statistical Society. Series D (The Statistician) 47*(1), 183–189.

Jones, E., T. Oliphant, P. Peterson, et al. (2001–). SciPy: Open source scientific tools for Python. `http://www.scipy.org/`. Online.

Kang, S., S. Sristi, J. Karachiwala, and Y. C. Hu (2018). Detection of anomaly in train speed for intelligent railway systems. *2018 Int. Conf. Control. Autom. Diagnosis, ICCAD 2018*.

Kennedy, R. (2006). Examining the Processes of RCM and TPM :. *Group* (January), 1–15.

Khan, S. and T. Yairi (2018). A review on the application of deep learning in system health management. *Mech. Syst. Signal Process. 107*, 241–265.

Kingma, D. and J. Ba (2014, 12). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Koons-stapf, A. (2015). Condition Based Maintenance : Theory , Methodology , & Application Amanda Gillespie. (January).

Koprinkova-Hristova, P. (2013). Reinforcement Learning for Predictive Maintenance of Industrial Plants. *Inf. Technol. Control 11*, 21–28.

Laskov, P., C. Schäfer, and I. Kotenko (2004). Intrusion detection in unlabeled data with quarter-sphere Support Vector Machines. *Lect. Notes Informatics (LNI), Proc. - Ser. Gesellschaft fur Inform. P 46*, 71–82.

Lee, W. J. (2017). Anomaly detection and severity prediction of air leakage in train braking pipes. *Int. J. Progn. Heal. Manag. 8*(Special Issue 7).

Li, Q., Z. Gao, D. Tang, and B. Li (2016). Remaining useful life estimation for deteriorating systems with time-varying operational conditions and condition-specific failure zones. *Chinese Journal of Aeronautics 29*(3), 662 – 674.

Li, Z. and Q. He (2015). Prediction of railcar remaining useful life by multiple data source fusion. *IEEE Trans. Intell. Transp. Syst. 16*, 1–10.

Liao, L. and F. Kottig (2014, jan). Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction. *IEEE Trans. Reliab. 63*.

Lopes Gerum, P. C., A. Altay, and M. Baykal-Gürsoy (2019). Data-driven predictive maintenance scheduling policies for railways. *Transp. Res. Part C Emerg. Technol. 107*(October 2018), 137–154.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. pp. 281–297.

Manco, G. (2017, November). Fault detection and explanation through big data analysis on sensor streams. *Expert Syst. Appl. 87*(C), 141–156.

Maya, S., K. Ueno, and T. Nishikawa (2019). dLSTM: a new approach for anomaly detection using deep learning with delayed prediction. *Int. J. Data Sci. Anal. 8*(2), 137–164.

Nguyen, K. T. and K. Medjaher (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliab. Eng. Syst. Saf. 188*(February), 251–262.

Nimesh Sinha (2018). Understanding lstm and its quick implementation in keras for sentiment analysis. `https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras`

-for-sentiment-analysis-af410fd85b47. Online; accessed 29 July 2020.

Oliveri, P. (2017). Class-modelling in food analytical chemistry: Development, sampling, optimisation and validation issues – a tutorial. *Analytica Chimica Acta 982*, 9 – 19.

Pearl, J. (2019). The seven tools of causal inference, with reflections on machine learning. *Commun. ACM 62*(3), 54–60.

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

Pereira, P., R. P. Ribeiro, and J. Gama (2014a). Failure prediction – an application in the railway industry. In *Discovery Science*, pp. 264–275. Springer International Publishing.

Pereira, P., R. P. Ribeiro, and J. Gama (2014b). Failure prediction – an application in the railway industry. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 8777*, 264–275.

Pierobon, G. (2018). Dbscan clustering for data shapes k-means can't handle well (in python). https://towardsdatascience.com/dbscan-clustering-for-data-shapes-k-means-cant-handle-well-in-python-6be89af4e6ea. Online; accessed 30 May 2020.

Pinto, P. (2019). *Projeto de um sistema de aquisição de dados para a manutenção preditiva de uma unidade embarcada de produção de ar comprimido*. Ph. D. thesis, University of Porto.

Rabatel, J., S. Bringay, and P. Poncelet (2011). Anomaly detection in monitoring sensor data for preventive maintenance. *Expert Syst. Appl. 38*(6), 7003–7015.

Rama, T. and Çöltekin (2016). Lstm autoencoders for dialect analysis. In P. Nakov, M. Zampieri, L. Tan, N. Ljubesic, J. Tiedemann, and S. Malmasi (Eds.), *VarDial@COLING*, pp. 25–32. The COLING 2016 Organizing Committee.

Ranjan, C. (2019a). Extreme rare event classification using autoencoders in keras. https://towardsdatascience.com/extreme-rare-event-classification-using-autoencoders-in-keras-a565b386f098. Online; accessed 29 July 2020.

Ranjan, C. (2019b). Step-by-step understanding lstm autoencoder layers.

https://towardsdatascience.com/step-by-step-understanding-lstm-autoencoder-layers-ffab055b6352. Online; accessed 29 July 2020.

Rassam, M. A., M. A. Maarof, and A. Zainal (2014). Adaptive and online data anomaly detection for wireless sensor systems. *Knowledge-Based Syst. 60*(May 2019), 44–57.

Ribeiro, M., S. Singh, and C. Guestrin (2016, feb). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*.

Rosasco, L. and T. Poggio (2015). Machine learning: a regularization approach. *MIT-9.520 Lectures Notes*.

Rousseeuw, P. (1987, November). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math. 20*(1), 53–65.

Sagheer, A. and M. Kotb (2019). Unsupervised Pre-training of a Deep LSTM-based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. *Sci. Rep. 9*(1), 1–16.

Sharma, S., Y. Cui, Q. He, R. Mohammadi, and Z. Li (2018). Data-driven optimization of railway maintenance for track geometry. *Transp. Res. Part C Emerg. Technol. 90*(March), 34–58.

Syarif, I., A. Prugel-Bennett, and G. Wills (2012). Unsupervised clustering approach for network anomaly detection. In R. Benlamri (Ed.), *Networked Digital Technologies*, Berlin, Heidelberg, pp. 135–145. Springer Berlin Heidelberg.

Tamilselvan, P. and P. Wang (2013). Failure diagnosis using deep belief learning based health state classification. *Reliab. Eng. Syst. Saf. 115*, 124–135.

Thorndike, R. (1953). Who belongs in the family? *Psychometrika 18*(4), 267–276.

Wang, G., T. Xu, T. Tang, T. Yuan, and H. Wang (2017). A Bayesian network model for prediction of weather-related failures in railway turnout systems. *Expert Syst. Appl. 69*, 247–256.

Wang, W., Q. He, Y. Cui, and Z. Li (2018). Joint prediction of remaining useful life and failure type of train wheelsets: Multitask learning approach. *J. Transp. Eng. Part A Syst. 144*(6).

Yan, H., J. Wan, C. Zhang, S. Tang, Q. Hua, and Z. Wang (2018, feb). Industrial Big Data Analytics for Prediction of Remaining Useful Life Based on Deep Learning. *IEEE Access PP*, 1.

Yeh, Y.-c., C.-y. Hsu, and Y.-c. Yeh (2019). Application of Auto-Encoder for

Time Series Classification with Class Imbalance.

Yuan, J. and X. Liu (2013). Semi-supervised learning and condition fusion for fault diagnosis. *Mech. Syst. Signal Process. 38*(2), 615–627.

Zhang, Y., N. Meratnia, and P. Havinga (2008). An Online Outlier Detection Technique for Wireless Sensor Networks. *3rd Eur. IEEE Conf. Smart Sens. Context*, 151–156.

Zhang, Y., N. Meratnia, and P. Havinga (2009). Adaptive and online one-class support vector machine-based outlier detection techniques for wireless sensor networks. *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA* (May 2009), 990–995.

Appendix A

Data Understanding

A.1 Data Exploration

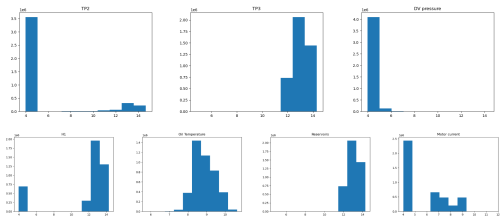


Figure A.1: Histograms of analogical sensors of APU01

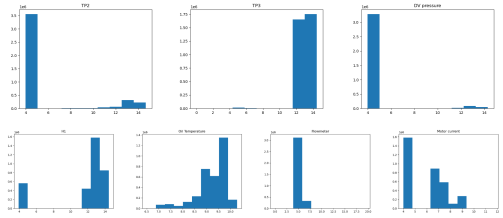


Figure A.2: Histograms of analogical sensors of APU02

A.2 Data Analysis

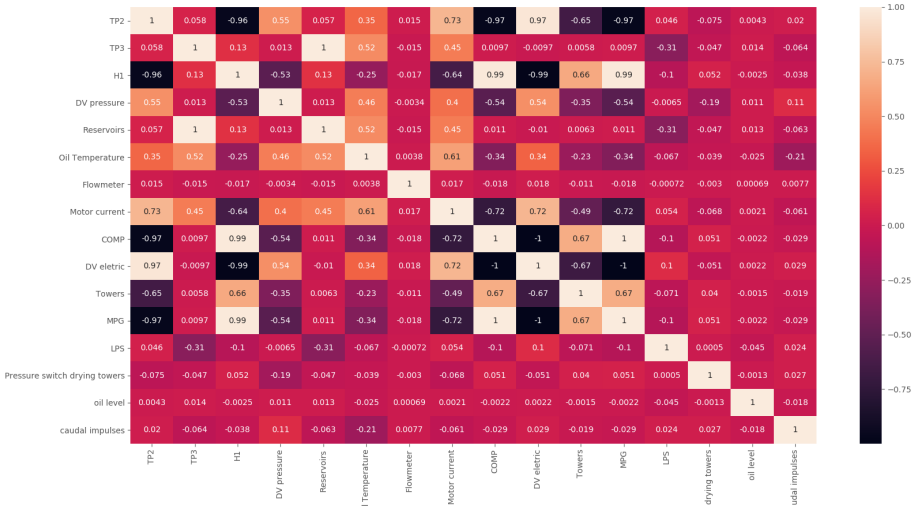


Figure A.3: Correlation between APU01 sensors

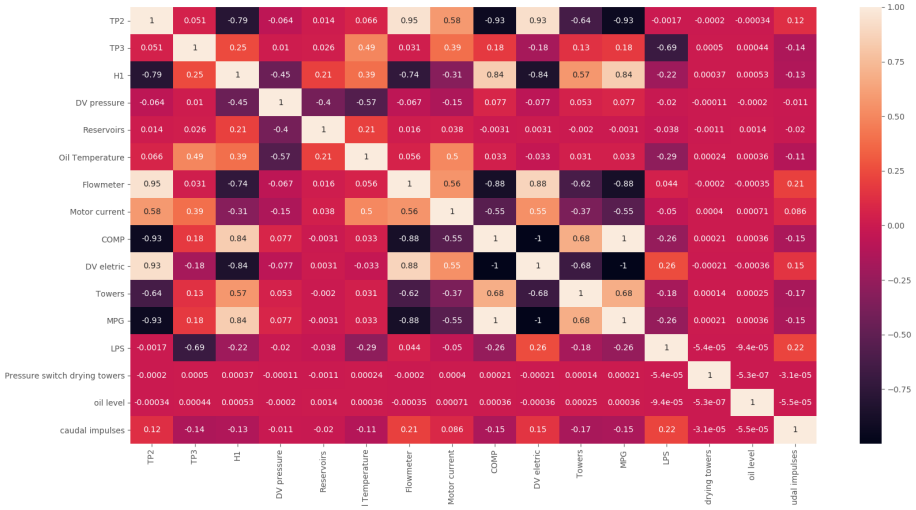


Figure A.4: Correlation between APU02 sensors

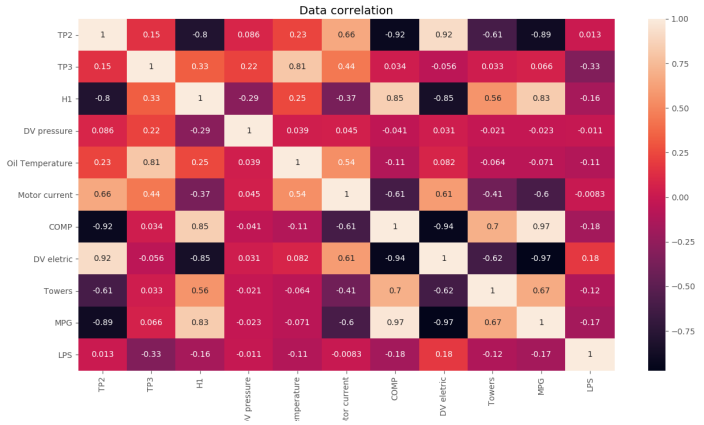


Figure A.5: Correlation between full dataset (APU01+APU02) sensors

Table A.1: Timestamp when APU01 and APU02 went to maintenance during the evaluation period

APU01	APU02
06-03-2020 01:08	12-03-2020 12:22
06-03-2020 08:10	17-03-2020 22:23
11-03-2020 10:37	19-03-2020 21:14
12-03-2020 11:30	20-03-2020 21:14
15-03-2020 13:00	24-03-2020 21:23
25-03-2020 21:23	28-03-2020 01:47
27-03-2020 01:40	31-03-2020 01:45
05-04-2020 01:47	31-03-2020 01:45
05-04-2020 23:00	03-04-2020 01:47
07-04-2020 06:22	04-04-2020 01:40
17-04-2020 22:40	04-04-2020 20:00
19-04-2020 02:13	08-04-2020 09:00
21-04-2020 01:36	09-04-2020 02:40
30-04-2020 10:45	16-04-2020 08:00
30-04-2020 12:40	22-04-2020 17:30
30-04-2020 10:45	24-04-2020 01:40
30-04-2020 12:40	08-05-2020 21:14
01-05-2020 20:00	21-05-2020 22:10
13-05-2020 00:00	23-05-2020 13:30

15-05-2020 22:30	31-05-2020 22:16
16-05-2020 05:00	05-06-2020 21:14
30-05-2020 01:10	06-06-2020 21:00
01-06-2020 22:10	09-06-2020 02:00
02-06-2020 10:55	10-06-2020 07:45
03-06-2020 20:24	14-06-2020 02:35
05-06-2020 20:37	19-06-2020 15:25
06-06-2020 21:00	22-06-2020 22:10
10-06-2020 21:00	26-06-2020 02:00
12-06-2020 10:53	28-06-2020 19:25
17-06-2020 05:20	05-07-2020 21:25
17-06-2020 10:00	08-07-2020 01:48
22-06-2020 20:24	08-07-2020 08:40
25-06-2020 21:00	08-07-2020 21:00
30-06-2020 21:18	16-07-2020 01:36
30-06-2020 21:18	18-07-2020 22:16
03-07-2020 21:18	22-07-2020 02:13
04-07-2020 21:14	30-07-2020 01:40
13-07-2020 10:23	
15-07-2020 19:30	
17-07-2020 07:20	
18-07-2020 21:00	
21-07-2020 20:37	
26-07-2020 23:00	
27-07-2020 04:30	
28-07-2020 18:05	
31-07-2020 20:24	

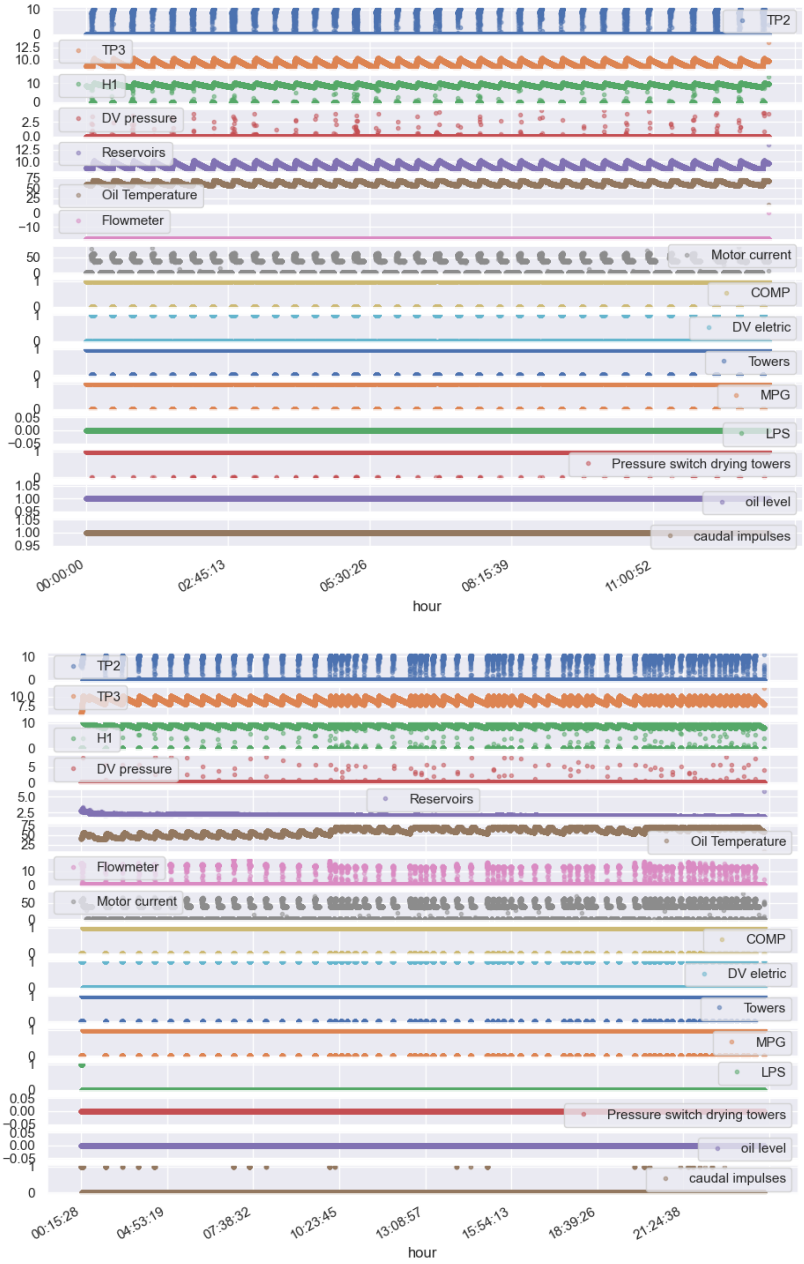


Figure A.6: Daily sensor readings: APU01 (top) APU02 (bottom)

Appendix B

Anomaly Detection

B.1 Anomaly detection based on sensors

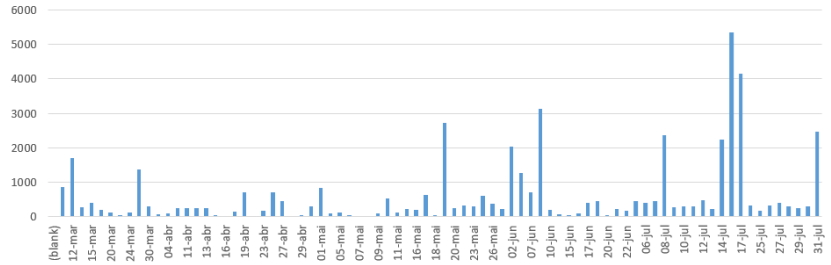


Figure B.1: Number of times the LPS was triggered when APU01 was operating

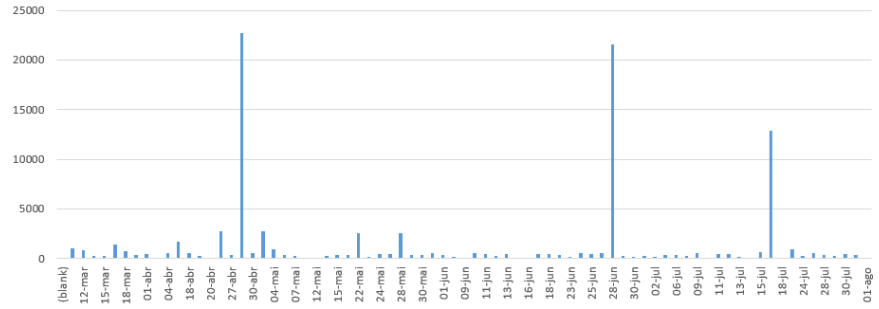


Figure B.2: Number of times the LPS was triggered when APU02 was operating

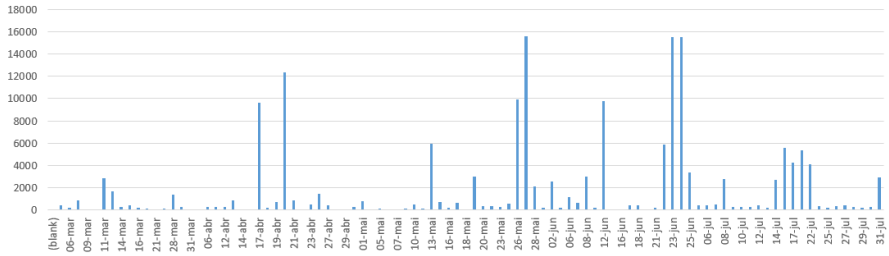


Figure B.3: Number of times TP3 outside of range when APU01 was operating

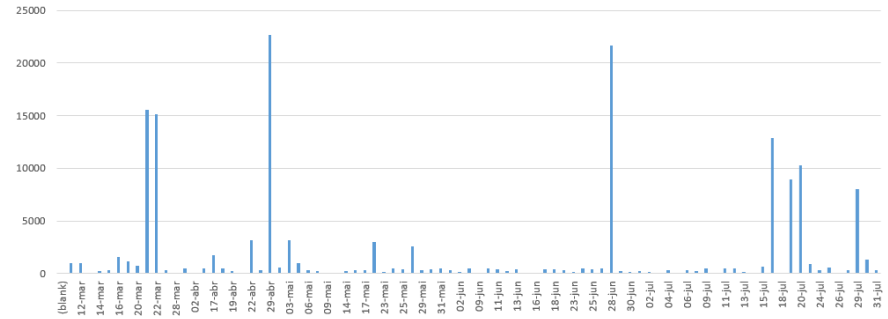


Figure B.4: Number of times TP3 outside of range when APU02 was operating

B.2 Peak Frequency

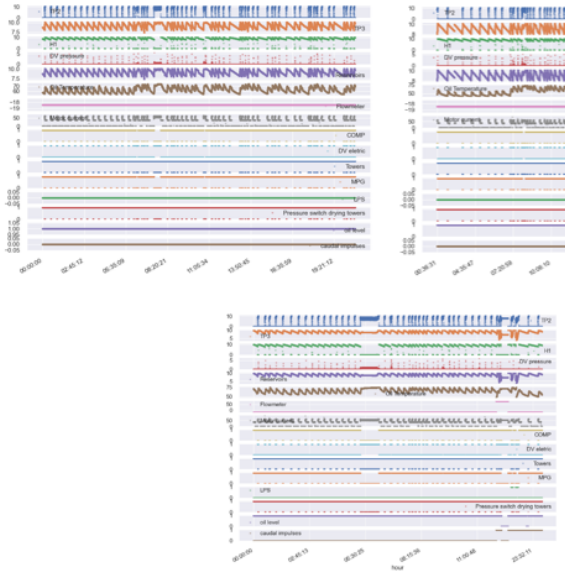


Figure B.5: Sensor readings on days before the first failure on APU01

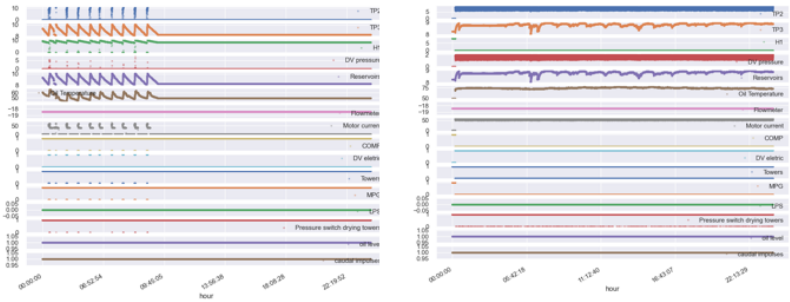


Figure B.6: Sensor readings on days before the second failure on APU02

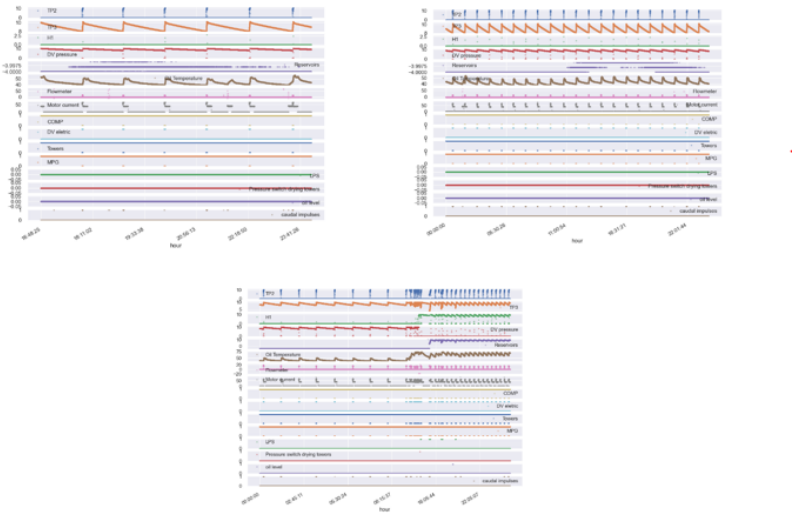


Figure B.7: Sensor readings on days before the first failure on APU02

Table B.1: Peaks in analogical sensors on days previous to a failure on APU01

APU01	09/03	10/03	11/03	12/03	26/03	27/03	16/04	17/04	18/04	19/04	20/04	Other days
TP2	6	8	4	4	4	5	4	2	3	4	1	4
TP3	6	8	4	3	5	5	5	2	3	5	1	5
H1	6	8	4	1	5	5	5	2	0	4	1	5
DV Pressure	2	3	1	2	2	3	2	1	3	2	1	2
Reservoirs	6	8	4	3	5	5	5	2	3	5	1	5
Oil Temperature	5	7	4	4	4	5	4	2	3	4	1	4
Motor Current	5	7	4	4	4	4	4	1	3	4	1	4

Table B.2: Peaks in analogical sensors on days previous to a failure on APU02

APU02	09/03	10/03	11/03	OtherDays
TP2	1	2	3	7
TP3	2	3	3	7
H1	0	0	2	7
DV Pressure	2	3	2	2
Oil Temperature	2	3	3	6
Flowmeter	3	2	3	7
Motor Current	2	2	3	6

Table B.3: Duration of peaks when compressor was off on failure days on APU01

APU01		<2 minutes	Between 2 and 10	Between 10 and 20	>20 minutes
09/03	Mean % Peaks	4s 16%	5m 40%	15m 25%	29m 17%
10/03	Mean % Peaks	3s 13%	5m 60%	15m 23%	29m 3%
11/03	Mean % Peaks	2s 2%	4m 50%	14m 45%	21m 2%
26/03	Mean % Peaks	3s 66%	2.5m 9%	11m 0.1%	27m 25%
17/04	Mean % Peaks	48s 68%	- -	- -	30m 42%

Table B.4: Duration of peaks when compressor working under load on failure days on APU01

APU01		<2 minutes	Close to 2	Between 2 and 10	Between 10 and 20	>20 minutes
09/03	Mean % Peaks	21s 12%	2m 38%	5m 35%	14m 12%	29m 12%
10/03	Mean % Peaks	18s 9%	2m 40%	5m 40%	14m 12%	25m 1,5%
11/03	Mean % Peaks	18s 7%	2m 33%	5m 43%	14m 18%	21m 1%
26/03	Mean % Peaks	18s 58%	2m 20%	3m 8%	11m 0.6%	21m 1%
17/04	Mean % Peaks	18s 24%	2m 66%	3m 4%	- -	26m 16%

Table B.5: Duration of peaks when compressor working under load on failure days on APU02

APU01	Measure	>60 minutes	<6 minutes	Between 6 and 60
09/03	Mean	66m	-	-
10/03	Mean	66m	-	-
11/03	Mean % Peaks	65m 15%	3m 12%	14m 73%

Table B.6: Duration of peaks when compressor was off on failure days on APU02

APU01	Measure	<2 minutes	1h35minutes
09/03	Mean	1m13s	
10/03	Mean	1m13s	
11/03	Mean % Peaks	18s 7%	1h 0,2%

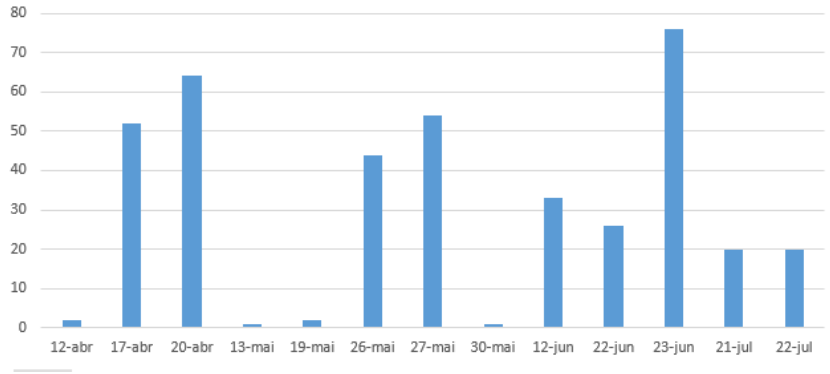


Figure B.8: Number of times rule 1 was triggered in each day on APU01

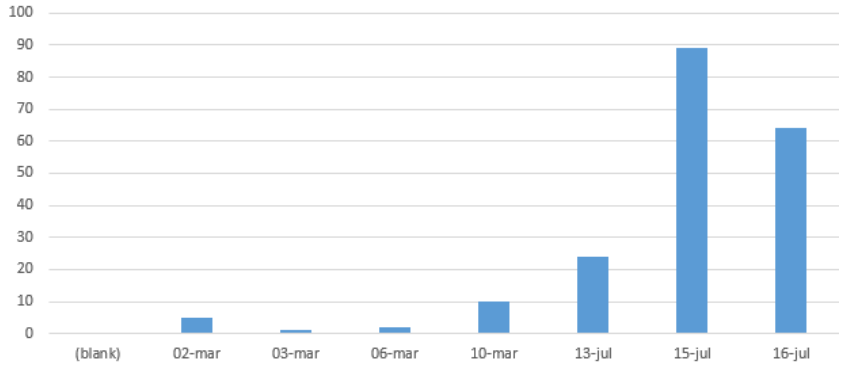


Figure B.9: Number of times rule 2 was triggered in each day on APU01

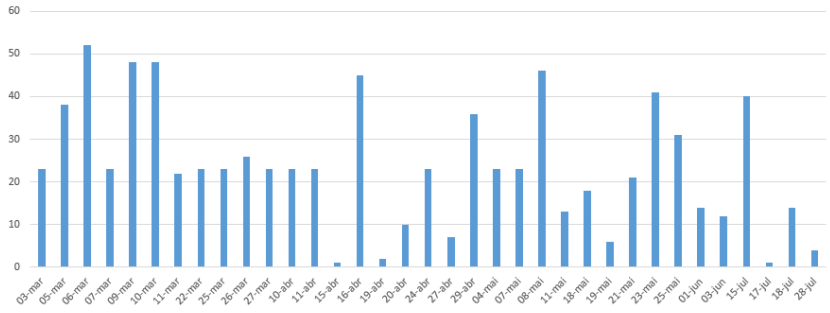


Figure B.10: Number of times rule 3 was triggered in each day on APU01

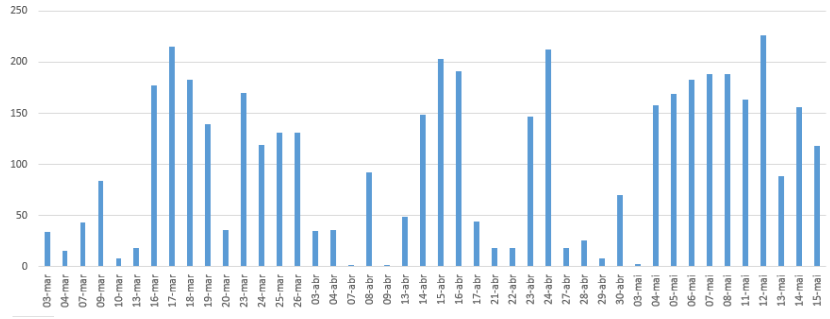


Figure B.11: Number of times rule 4 was triggered in each day on APU01

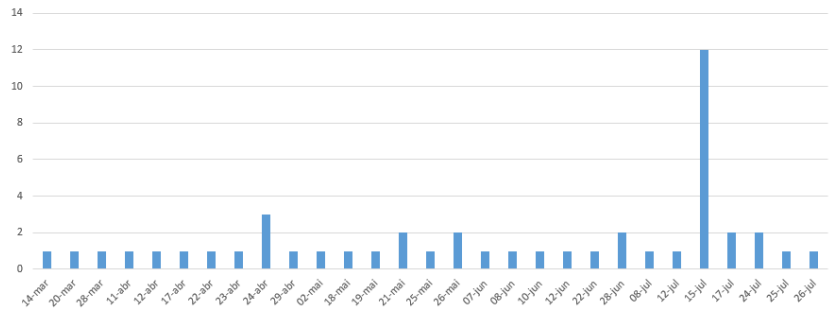


Figure B.12: Number of times rule 5 was triggered in each day on APU01

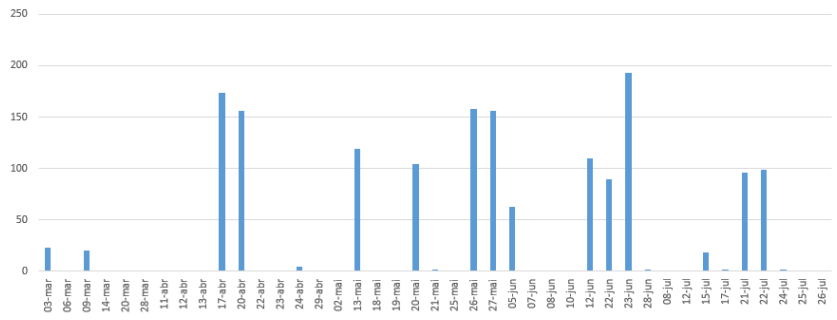


Figure B.13: Number of times rule 6 was triggered in each day on APU01

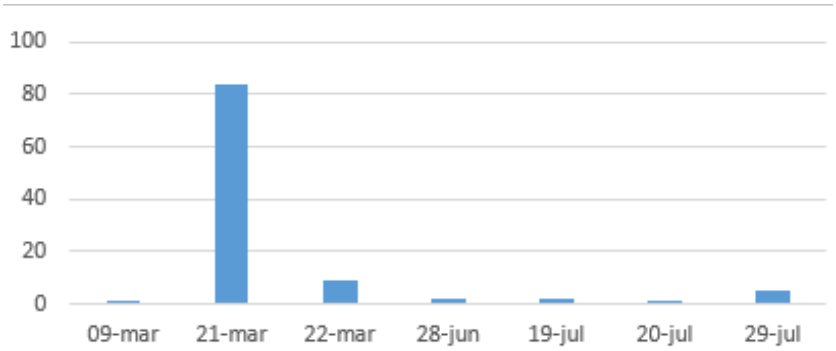


Figure B.14: Number of times rule 1 was triggered in each day on APU02

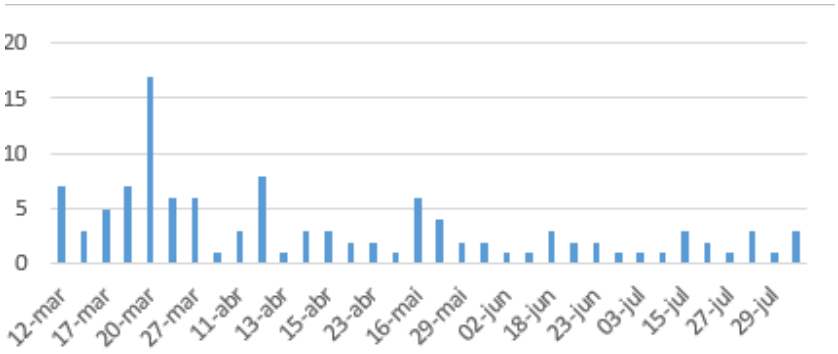


Figure B.15: Number of times rule 2 was triggered in each day on APU02

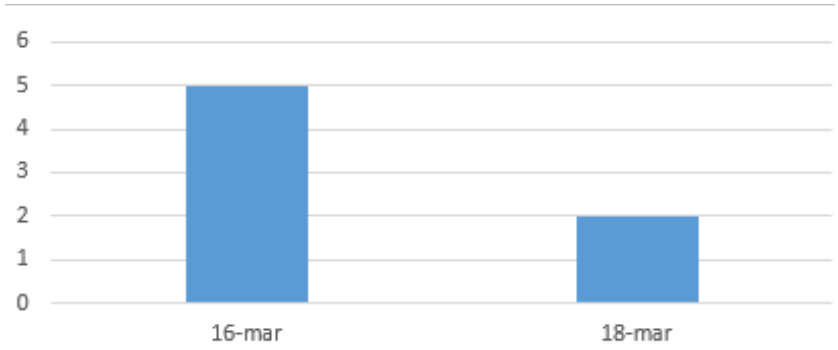


Figure B.16: Number of times rule 3 was triggered in each day on APU02

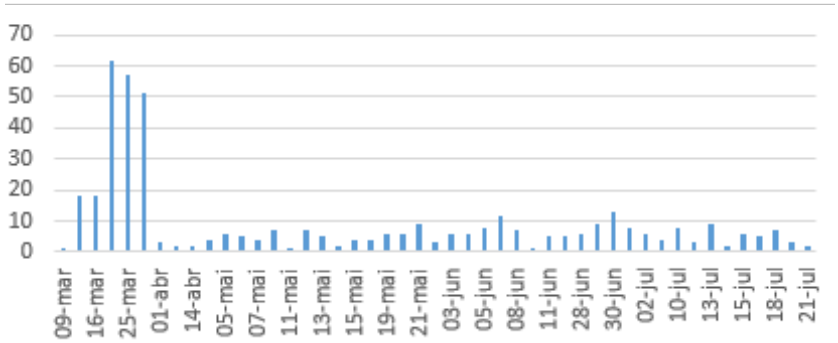


Figure B.17: Number of times rule 4 was triggered in each day on APU02

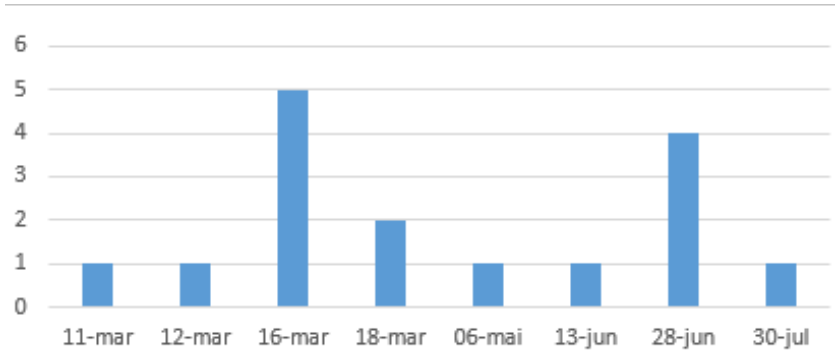


Figure B.18: Number of times rule 5 was triggered in each day on APU02

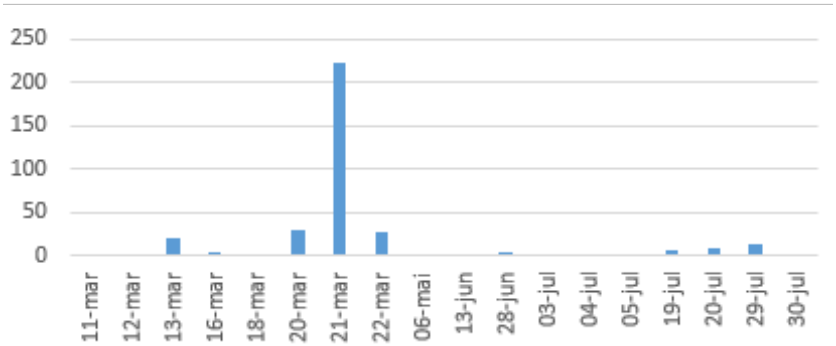
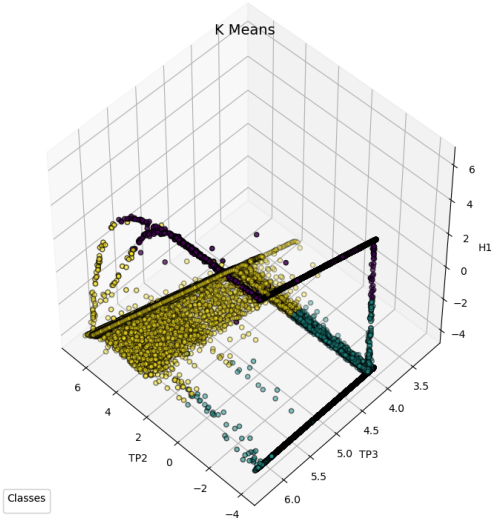


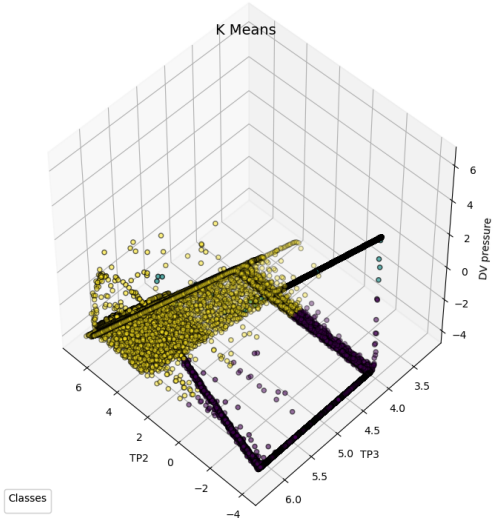
Figure B.19: Number of times rule 6 was triggered in each day on APU02

Appendix C

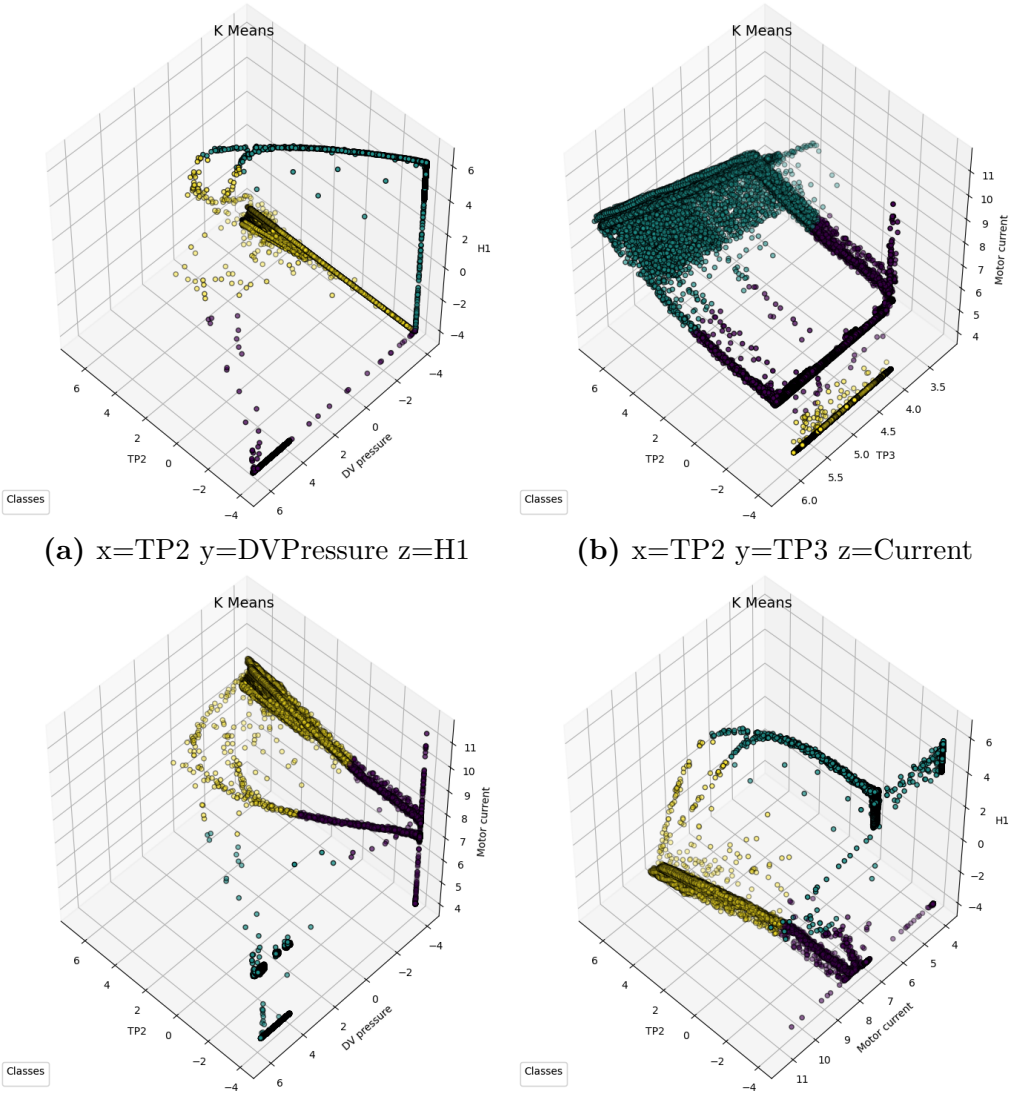
Clustering



(a) x=TP2 y=TP3 z=H1



(b) x=TP2 y=TP3 z=DVPressure



(a) x=TP2 y=DVPressure z=H1

(b) x=TP2 y=TP3 z=Current

(c) x=TP2 y=DVPressure z=Current

(d) x=TP2 y=Current z=H1

Figure C.2: K-Means clustering results

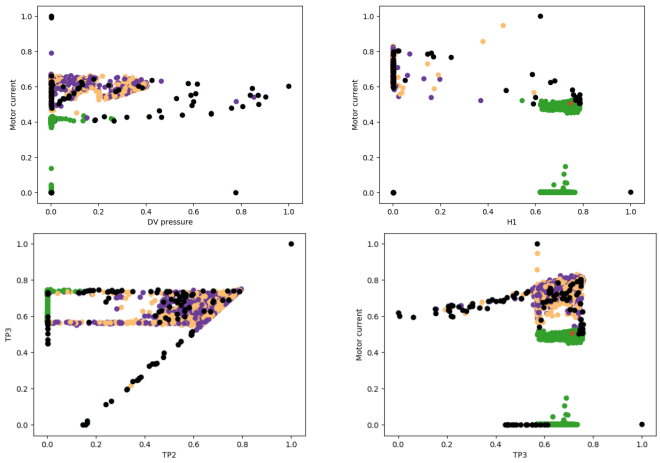


Figure C.3: APU01 DBSCAN clustering

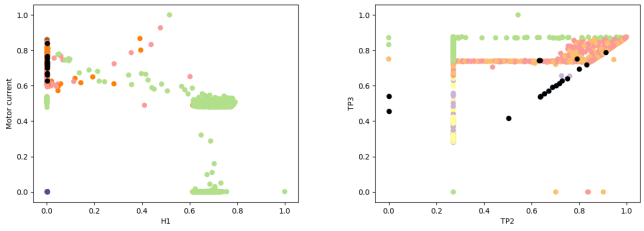


Figure C.4: APU02 DBSCAN clustering

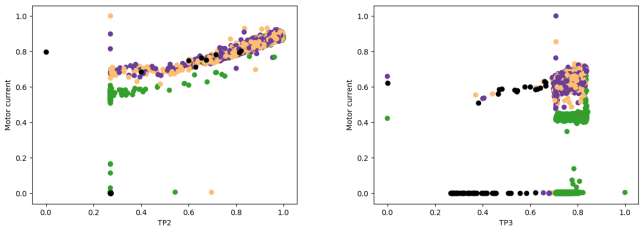


Figure C.5: APU01+APU02 DBSCAN clustering