INSTITUTO POLITÉCNICO DE BRAGANÇA Escola Superior de Tecnologia e Gestão

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Intelligent Systems for Defect Monitoring and Diagnosis in the Body shop of a Automotive Assembly Line

Júlio César Guimarães Costa - 58945

Thesis presented to the School of Technology and Management of Bragança to obtain the master's degree in Informatics within the scope of the double degree program with the Federal University of Technology – Paraná.

Supervisors:

Prof. Paulo Jorge Pinto Leitão

Prof. José Fernando Lopes Barbosa

Prof. Gleifer Vaz Alves

This document does not include the suggestions made by the board.

Bragança

2023-2024

Intelligent Systems for Defect Monitoring and Diagnosis in the Body shop of a Automotive Assembly Line

Júlio César Guimarães Costa - 58945

Thesis presented to the School of Technology and Management in the scope of the Master in Informatics.

Supervisors:

Prof. Paulo Jorge Pinto Leitão

Prof. José Fernando Lopes Barbosa

Prof. Gleifer Vaz Alves

This document does not include the suggestions made by the board.

Bragança

2023-2024

Dedication

I dedicate this work to God and to everyone who supported me throughout its development, especially Prof. Gleifer Vaz Alves and Prof. Andre Pinz Borges. I would also like to express my gratitude to my mother, Detânia Guimarães, and my father, João Costa, who believed in me from the very beginning.

Acknowledgment

This work was partially supported by the HORIZON-CL4-2021-TWIN-TRANSITION-01 openZDM project, under Grant Agreement No. 101058673. Also, it was supported by national funds through FCT/MCTES (PIDDAC): CeDRI, UIDB/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and UIDP/05757/2020 (DOI: 10.54499/UIDP/05757/2020); and SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020). I also want express my gratitude to Prof. José Carlos Rufino, Prof. Rui Pedro Lopes and the IPB team for configuring the development environment in order to replicate the shop-floor production machine resources and for devising the micro-service infrastructure. Finally I want to thank the openZDM consortium specially those representing the vehicular industry for providing useful shop-floor knowledge.

Abstract

This study investigates data analysis techniques and the development of two analytical tools in the openZDM project for the body shop and final assembly measurement stations of a automotive industry plant. These systems were designed to generate quality information, helping the industry to achieve the Zero Defect Manufacturing (ZDM) goals. The use case section provides essential background for understanding the nuances and limitations of the case study. Next sections devised the tools to apply the Zero Defect Manufacturing strategies, firstly the proposed rule-based monitoring framework, which enables quick adaptation of context-specific and general rules for statistical, trend, and Machine Learning based monitoring. Two micro-services serve as the diagnostic tool, evaluating the car and measurement quality while classifying defects, and generating their causes, consequences, and process implications. The tests carried out demonstrated the system's effectiveness in handling a high-frequency data environment having a processing time 30 times faster than the required by the use case, and offering reliable outputs for proactive decision-making. The tools development was a collaborative effort within the industry partners, this concern with the operator comfort reinforces mainly the adaptation to the existing industrial process.

Keywords: ZDM, Analytical Tools, Monitoring, Diagnosis

Resumo

Este estudo investiga técnicas de análise de dados e o desenvolvimento de duas ferramentas de analise de dados no projeto openZDM para as estações de medição da montagem final e da funilaria em uma fábrica automotiva. Esses sistemas foram projetados para gerar informações de qualidade, ajudando a indústria a alcançar os objetivos do Zero Defect Manufacturing (ZDM). A seção do caso de estudo fornece o contexto essencial para compreender as nuances e limitações do estudo de caso. As próximas seções detalham as ferramentas aplicadas para implementar as estratégias de ZDM, começando com o framework de monitoramento baseado em regras, que permite a rápida adaptação de regras específicas e gerais para o monitoramento estatístico, de tendências e baseado em aprendizado de máquina. Dois microsserviços funcionam como ferramentas de diagnóstico, avaliando a qualidade do carro e das medições, classificando defeitos e gerando suas causas, consequências e implicações nos processos. Os testes realizados demonstraram a eficácia do sistema em lidar com um ambiente de dados de alta frequência, com um tempo de processamento 30 vezes mais rápido que o necessário para o caso de uso, além de fornecer saídas confiáveis para decisões proativas. Este desenvolvimento foi fruto da colaboração entre o autor e os parceiros da indústria, e a preocupação com o conforto do operador reforça a viabilidade e qualidade do sistema no ambiente de manufatura.

Palavras-chave: ZDM, Ferramentas de Análise, Monitoramento, Diagnostico.

Contents

List of Figures

Acronyms

CPU Central Processing Unit.

ESTiG Escola Superior de Tecnologia e Gestão.

IPB Instituto Politécnico de Bragança.

JSON JavaScript Object Notation.

ML Machine Learning.

openZDM Open platform for realizing zero defects in cyber-physical manufacturing.

RAMI4.0 Reference Architectural Model Industrie 4.0.

ZDM Zero Defect Manufacturing.

Chapter 1

Introduction

1.1 Context Statement

In modern car manufacturing, maintaining precision and ensuring quality at every stage of the production process is critical to delivering a high-performing, reliable, and aesthetically appealing vehicle. A key part of this process is the use of measurement stations in the body shop and assembly lines, which capture vital data to ensure that the car body structure and its components meet stringent dimensional and alignment specifications. Accordingly with S. A. Spiewak et al. [37] *"The cost incurred as a result of equipment failures can reach for individual machine tools hundreds of thousands of dollars per incidence. In the case of complex installations such as automobile assembly lines it can be as high as $20k per minute."*, this statement ensures that preventing, classifying and diagnosing already mapped problems could minimize not only waste of materials, but of money and investments.

This thesis focuses on a specific use case within the car manufacturing industry, where three measurement stations collect data to ensure that the X, Y, Z coordinates of the car body and the gap and flush values between certain points and junctions from metal plates are within design tolerances. The first station captures geometric data (X, Y, Z coordinates) of the car body after the welding process, while subsequent stations monitor

the gap and flush alignment at critical points on the car doors, front, back, and roof sections.

These measurements are crucial for preventing assembly errors, maintaining product quality, and ensuring customer satisfaction. However, the complexity of modern car designs and the increasing demand for tighter tolerances have led to challenges in achieving high levels of precision consistently across large-scale manufacturing processes. S. A. Spiewak [36] also add that the effective use of Monitoring and Diagnosis systems can significantly improve the overall productivity by decreasing the troubleshooting time and avoiding major unexpected breakdowns. Leading to the possibility to explore new emerging data analytics strategies in this high quality, and high demand environments. This use case will serve as a foundation for exploring potential solutions aimed at improving measurement accuracy and overall process efficiency to achieve the Zero Defect Manufacturing.

1.2 Objectives

This thesis have as the main objective develop solutions for monitoring and diagnosis in the body shop and assembly lines into a vehicular industry, applying data-driven intelligent systems, to achieve the main goal of Zero Defect Manufacturing, grounded on that context the following sub-objectives are described as follows:

- **Analyze the Current Measurement Process:** To examine the role of measurement stations in the car manufacturing industry, focusing on how they capture data related to the X, Y, Z coordinates of the car body structure and gap and flush measurements at different stages of assembly.

- **Identify Key Challenges:** To identify the challenges and limitations associated with the current measurement systems, such as issues related to accuracy, consistency, and efficiency, especially in complex vehicle designs.

- **Propose an Improved Solution:** To suggest a more effective solution that integrates advanced technologies or optimized processes, aimed at enhancing the precision of measurements, minimizing errors, and improving overall efficiency in the manufacturing process.
- **Evaluate the Proposed Solution:** To demonstrate the potential impact of the proposed solution through case studies, simulations, or data analysis, highlighting its benefits over existing practices in terms of accuracy, cost-efficiency, and product quality.

1.3 Document Structure

The following sections will outline the function of each measurement station, their importance in maintaining product quality, by improving the data collection, analysis and processes. Also this use case will serve as the foundation for proposing an enhanced solution aimed at improving accuracy, efficiency, and data integration across the manufacturing process.

Chapter 1 presents an introduction to the work and objectives, contextualize the project environment and why it should be developed.

Chapter 2 reviews existing research and technologies relevant to the current study, focusing on Data Analysis mainly in techniques for analyzing large datasets, particularly in manufacturing environments. The role of micro-services in modernizing manufacturing processes by improving scalability and flexibility. And a discussion on the tools and technologies available for data monitoring and diagnosis.

Chapter 3 focuses on the methodology and use case within the car manufacturing industry, providing an in-depth exploration of the data collected from three measurement stations. Also within the application of the KDD - Knowledge Discovery in Databases methodology in analyzing the provided data.

Chapter 4 presents the proposed solution, focused on enhancing zero-defect manufacturing using data analytic tools deployed in a micro-service architecture. The Experiments

conducted to test both systems and the experimental results.

Chapter 5 evaluates the performance of the proposed solutions, discussing the results of the monitoring and diagnosis micro-services in detail. Focusing on how well the monitoring rules performed in detecting and preventing defects, how effective the diagnosis micro-service was in identifying and prioritizing defects and how well the services worked in testing and production phases.

Finally Chapter 6 summarizes the key findings of the research and presents the overall conclusions of the study. It also discusses the potential for future work in further optimizing manufacturing processes and expanding the application of data analytics in automotive production.

Chapter 2

State of the Art

This chapter provides a summary of the technologies and ongoing research aimed at achieving objectives similar to this study. It discusses the applications implemented, the results achieved, and examines the development and use of monitoring and diagnoses systems in the manufacturing environment into achieving the Zero Defect Manufacturing paradigm.

2.1 Zero Defect Manufacturing

Recent studies emphasize that intelligent monitoring and diagnosis systems capable of addressing manufacturing challenges are critical for the realization of ZDM in Industry 4.0 environments [38]. The combination of data preprocessing techniques, such as imputation for missing data and normalization algorithms, with advanced analytics plays a crucial role in ensuring the robustness and reliability of these systems. This architectural evolution introduces several challenges and opportunities, mainly due to the high availability of data. Particularly, an opportunity is to develop real-time data analytics that converts data into knowledge, generating insights for potential continuous improvements in production systems. This is particularly important to implement ZDM strategies [9], where the objectives include reducing production costs and adding value to the product by reducing rework.

Trebuna et al. [38] realize the research gaps in ZDM in *"Comparing modern tools, mainly software tools, with their effect on ZDM strategies opens the question of how academic research is disconnected from real industrial problems. Many companies across industries still use spreadsheet software, such as Excel, for daily short-term scheduling compared to the academia sector, which already discusses connecting discrete event simulation tools with Advance Scheduling tools. This type of company commonly collects data manually, record them into ERP solutions, and then manually triggers any change."*, strengthening the fact that industries have a slow pace in terms of adapting to this new technologies.

Trebuna also define ZDM as a holistic approach for ensuring both process and product quality by reducing defects through corrective, preventive, and predictive techniques, using mainly data-driven technologies and guaranteeing that no defective products leave the production site and reach the customer, aiming at higher manufacturing sustainability, which is in accordance with this work objectives.

Minnetti et al. [27] introduced a solution in capturing gap and flush measurements in the body shop and assembly lines stating that quality control is a decision-making process aimed to asses if a part or a product complies to specifications; this is done by measuring specific characteristics and comparing them to the requirements set at the design phase. The outcome is a diagnostic judgment. Outlining the relevance of diagnostic systems and research's in identifying if the product complies to the specifications that will be further in this work be the foundations of the solutions.

2.2 Data Analysis and Knowledge Discovery in Manufacturing

Dacal et al. [10] suggested that feature correlation, which orders the variables by their weight in a specific dataset, can be used to minimize or maximize a quality output value. It essentially suggests which variables are the ones which seem to have correlation with an

output. Launer [40] introduced the first impressions and importance of starting analysis in datasets with a Exploratory Data Analysis (EDA) that explore behavioral patterns of the variables of study, establishing a hypothesis with the least possible structure. Conversely Abelairas and Astorkiza [1] states that Exploratory Spatial Data Analysis includes a set of techniques that describe and visualize those spatial effects: spatial dependence and spatial heterogeneity. It describes and visualizes spatial distributions, identifies outliers, finds distribution patterns, clusters and hot spots and suggests spatial regimes or other forms of spatial heterogeneity and, it is being increasingly used. Both works define the foundations of the exploratory analysis in this work for extracting new knowledge from the contextualized data.

Into knowledge discovery a robust process methodology widely applied is the Knowledge Discovery in Databases, that consists into a 5 steps process (See Figure 2.1) that can be recursively applied in order to acquire knowledge about a dataset.

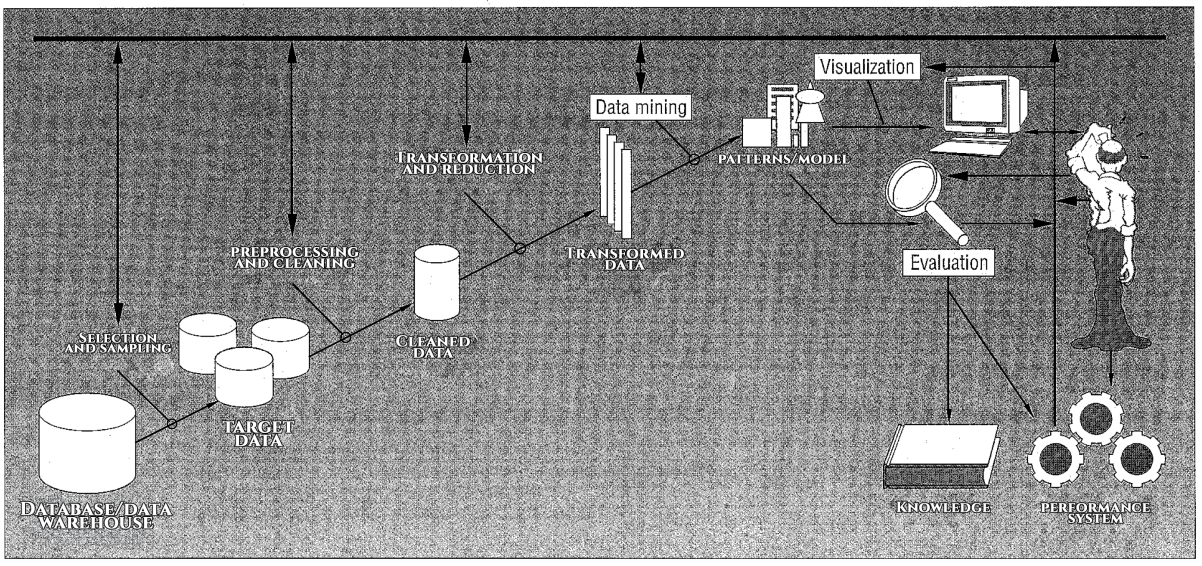


Figure 1. An overview of the KDD process.[2] (For simplicity, the illustration omits arrows indicating the multitude of potential loops and iterations.)

Figure 2.1: KDD Process [11]

Maki and Teranishi [25] states the relevance of applying the KDD methodology in manufacturing saying that the KDD process generally consists mainly of two steps, which are “discovery” and “verification”. In quality control, hypothesizing the cause of problems

of a production process corresponds to the discovery step, and a detailed data analysis corresponds to the verification step. This logic can be applied recursively in order to acquire more knowledge about the behavior of the data but also hidden patterns though the data mining techniques that are part of the KDD process, not only this allows an enchantment in the diagnosis system that is the objective of this work.

2.3 Micro-Services in Manufacturing

Architectural paradigms in software engineering are widely discussed now a days as they imply a lot of de disadvantages and advantages in software development. The adoption of a micro-service infrastructure allows the systems to be highly configurable as a flexible framework. Otherwise some limitations of this infrastructure paradigm can notice in certain resource optimizations, but the advantages far outweigh the disadvantages as resources now a days are not that limiting. Irudayaraj [19] et al. listed the technical advantages of adopting this architecture, claiming that service autonomy allows each component a undependable and isolated loosely coupled service which can be deployed independently. As you break the more independent services, it is simplify to reuse and integrate with all front-end services and exposed as third party services.

Murugan (2024) [28] states the limitations of old solution into adapting to I4.0 standard that imply the need for scalability, agility, flexibility, fault tolerance, interoperability and decentralized data management in *"Traditional monolithic architectures, prevalent in legacy industrial systems, face challenges in meeting the demands of Industry 4.0. The rigidity of monoliths poses obstacles in scalability, adaptability to change, and efficient resource utilization. As industries strive for real-time insights, dynamic process orchestration, and seamless integration of diverse technologies, the limitations of traditional architectures become apparent."*., to demonstrate that, figure 2.2 shows the structural difference in between both monolith and micro-service approaches.

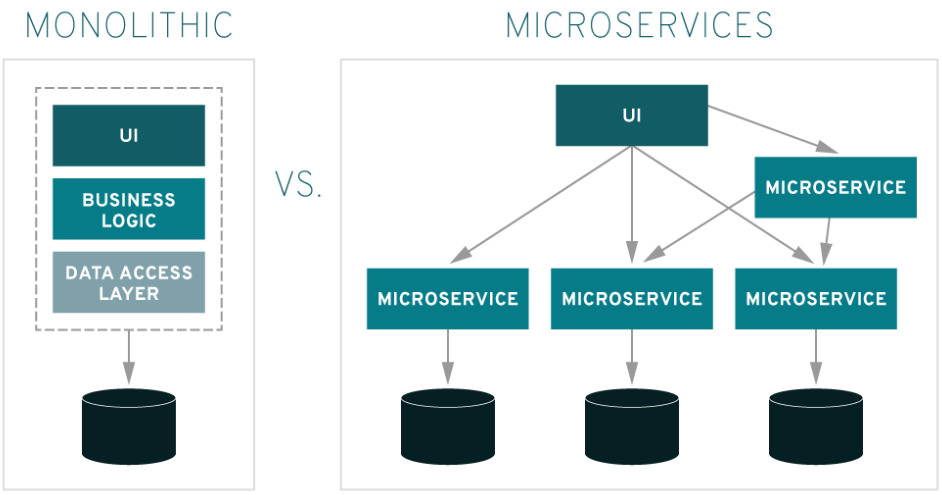


Figure 2.2: Monolith vs Micro-services development architectures. [33]

2.4 Rule-Based Monitoring Systems in Manufacturing

Monitoring tools usually require collecting vast amounts of data from different data sources, and Internet of Things (IoT) and Machine-to-Machine technologies can be used for this purpose. The use of advanced data analytics, and particularly Machine Learning (ML) techniques, allows to perform real-time monitoring aiming to detect deviations and trends of the parameter's evolution over time. In particular numerous studies have explored the application of rule-based systems for real-time variable monitoring in manufacturing, highlighting its scientific and commercial importance. For instance, Yang et al. [42] conducted a longitudinal study that introduced a real-time rule-based engine integrated specifically for Key Performance Indicators (KPIs) for managerial insights.

Conversely, Gunasegaram et al. [15] investigated the use of machine-learning models in an uncertain manufacturing environment, demonstrating their effectiveness in managing stochastic conditions. Liu et al. [23] introduced an intelligent quality prediction and autonomous decision system to improve quality management in natural product manufacturing by leveraging ZDM concepts and multi-agent with reinforcement learning.

Similarly, Smith et al. [34] proposed a novel functional outlier detection method that leverages domain knowledge to identify defective products, especially those with latent defects, presenting a systematic framework for integrating domain knowledge into outlier detection methodologies.

Ghansiyal et al. [14] applied adversarial neural networks that focus on discriminator training focused on detecting defects to achieve the ZDM environment. Still, besides all these innovative approaches and good results, there are some manufacturing limitations, according to data collection, pre-processing, and a consistently changing environment, that could be explored. Finally, May, Gokan and Kiritsis, Dimitris [26] proposed a methodology architecture to ZDM environments called Z-Factor, this approach, besides being robust in handling different use cases from the manufacturing environment has shown some limitations by using a holistic architecture, this could hinder the integration process in older companies.

The existing computational applications to monitor the operation parameters usually present difficulties in integrating data from multiple and heterogeneous data sources, usually legacy systems with proprietary protocols for accessing data, and problems in expressing the rationale for identifying anomalies and tendencies for a multiplicity of scenarios. Additionally, most of the existing quality control is concentrated on single-stage processes, such as Statistical Process Control (SPC), Design Of Experiments (DoE), and Six Sigma and Lean tools. However, in multi-stage manufacturing systems, a change in an upstream quality parameter may affect some downstream quality parameters in subsequent stages, which is known as cascade property [4].

Some research's suggest that hybrid systems combining data-driven ML models with rule-based systems can offer superior performance in dynamic environments [21] [6] such as the vehicular manufacturing. This work presents a rule-based framework aiming allow the implementation of different types of algorithms in a fast data driven scenario allowing for not only statistical rules to be devised but also ML and other neural network algorithms, that from classifications can generate useful information to be monitored.

When evaluating industrial process the rules usually aim to verify the product quality

into matching the standards defined by the companies, with this goal a widely used set of rules are the Nelson Rules that introduces quality control statistical rules [30] that offer the means of verifying whether a sequential variable is within control by analyzing a set of data points. Figure 2.3 along with the following items describe the 8 Nelson Rules and the quality problems indicated by triggering them.

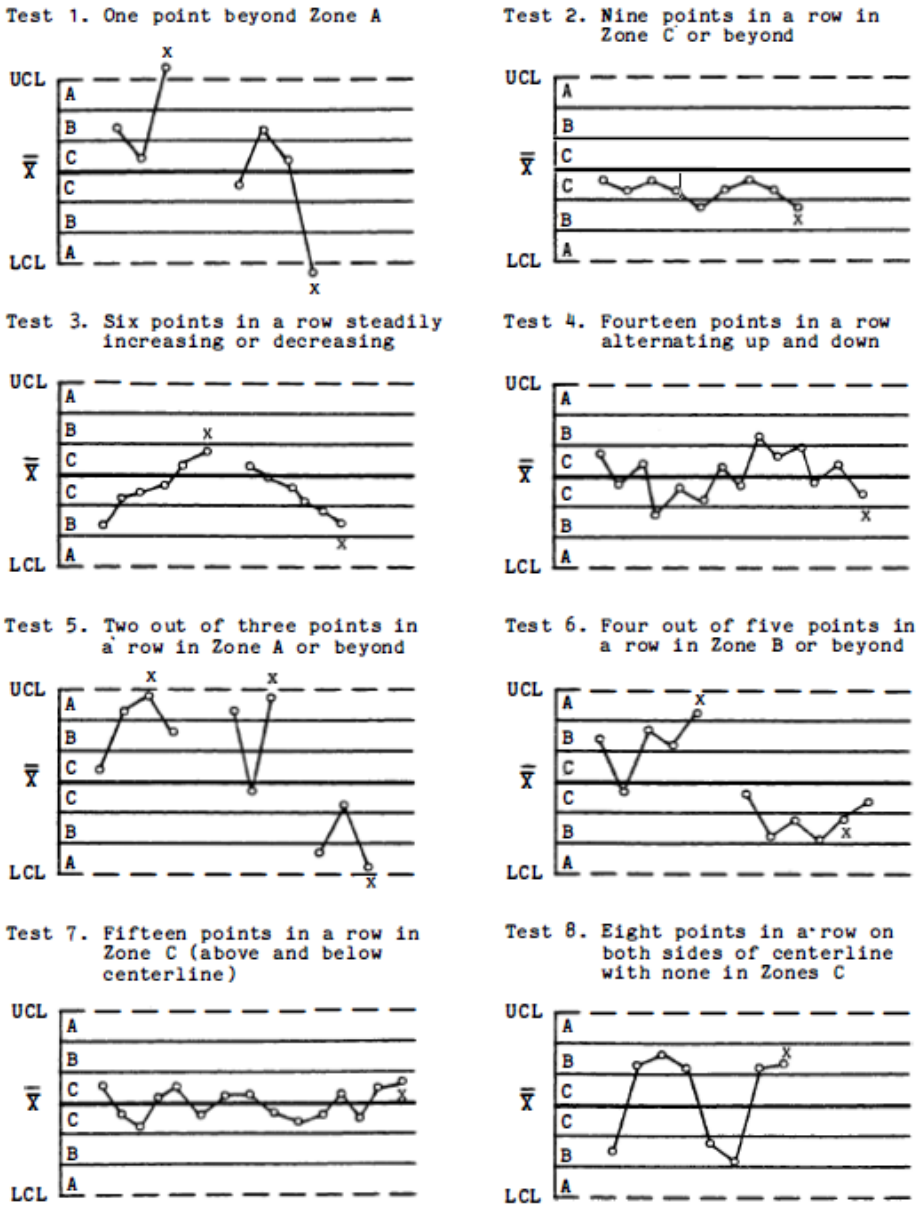


Figure 2.3: Nelson Rules [30]

- Nelson Rule 1: A single sample point is significantly out of control.
- Nelson Rule 2: There is evidence of a sustained bias.
- Nelson Rule 3: A clear trend is present in the data.
- Nelson Rule 4: The observed level of oscillation exceeds random variability.
- Nelson Rule 5: There is a moderate likelihood that samples are moderately out of control.
- Nelson Rule 6: There is a strong tendency for samples to be slightly out of control.
- Nelson Rule 7: With one standard deviation, more variation would typically be expected.
- Nelson Rule 8: Abrupt shifts from above to below, skipping the first standard deviation band, are rarely due to random chance.

2.5 Diagnosis Systems in Manufacturing

In reviewing the related work on diagnostic systems and their applications, it is evident that such systems are more commonly found in medical contexts. However, the principles of defect classification and the presentation of causes, consequences, and solutions are now being applied to industrial settings. For instance, the U.S. National Institute of Standards and Technology (NIST) [31] proposed a project to develop Monitoring, Diagnostics, and Prognostics for Manufacturing Operations, highlighting the growing importance of diagnostics in this field. Notably, there is no clear consensus on the definition of diagnostics in manufacturing, with some authors equating it as a monitoring problem [3] [22], while others focus on defect classification [43] [20] [32].

For instance Yuan et al. [43] implemented a framework applying a Convolutional Neural Network to captured images for identifying defects with two different approaches, a binary classification and multi-class defect level classification. Conversely, Mohammed

et al. [3] applied deep learning techniques to build a diagnostic system that was able to estimate the required services in an efficient and effective way.

From the specific perspective from those how treat the diagnosis as a defect/fault classification problem they usually implement Machine Learning [17] and neural network [29] models to identify the data behavior and then classify these instances, but besides this techniques being powerful in terms of discover data relations and pattern the application of these type of algorithms demands a good knowledge from the case study and the data collected for a good fine tuning of this models. Other two problems raise when talking about Machine Learning models for diagnosis, the training phase can take a long time in bigger amounts of data which can be harmful in real-time high frequency data collection environments. Also in this scenarios data can quickly change from one pattern to another, raising the need for new re-training.

This work presents a novel methodology for interpreting gap and flush data from a vehicular body shop and assembly lines transforming real-time monitored data into quality metrics not only about the product but also the measurement process, this quality metrics are then injected into a statistical model that evaluates and classify cars highlighting the causes, consequence and process indications. With this approach merging the non consensual approaches found in the related work. But maintaining the fast responses the context requires.

Chapter 3

Case Study

This chapter presents the use case context explanation, the functional and non-functional requirements to achieve the objective goals, and also include an exploratory data analysis to identify the industry data characteristics, limitations, circumstance behaviors, patterns and other insights.

3.1 Body Shop and Assembly Line Measurement Stations

The case study focuses on three primary measurement stations, each responsible for monitoring different aspects of the car body after key assembly and welding stages. The first measurement station captures X, Y, and Z coordinate data of the car body structure following the welding process. The second measurement station focuses on key points around the car doors and surrounding structures, measuring both Gap and Flush. Gap is defined as the perpendicular distance between two parts, while Flush refers to the distance along the axis of the normal surface between two parts. The third measurement station also measures Gap and Flush on the front, back, and roof of the car, ensuring panel alignment and fit consistency. Each station utilize a Laser Tracker measurement equipment to acquire and collect data. Each measurement collected has respectively 6

threshold values following a hierarchical structure of deviations (see Figure 3.1), which range go from the smallest acceptable deviations to those requiring rejection and further analysis.

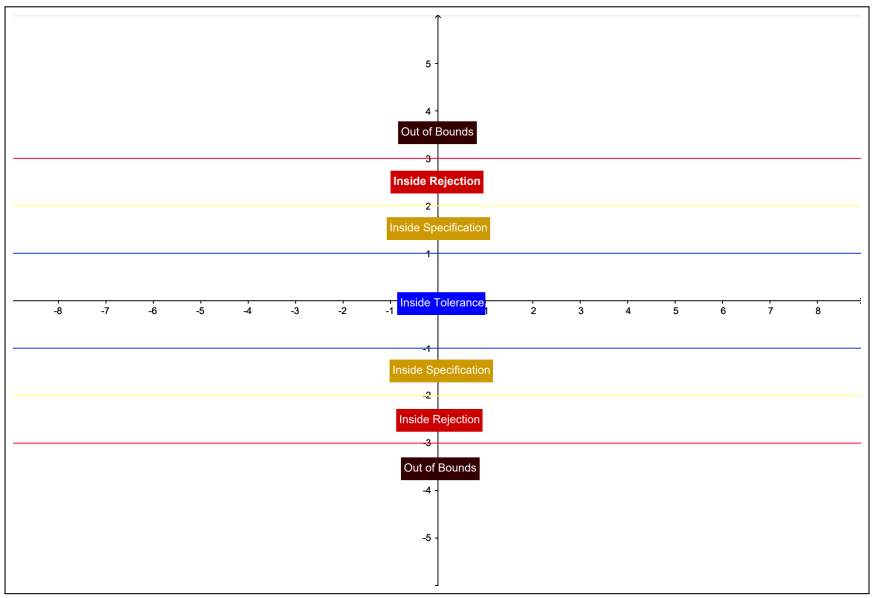


Figure 3.1: Thresholds hierarchy.

Based in a classified industry document, the data follows a unified naming convention format: 'Measurement type', 'Reference point', and 'Car position side relative to a rear reference'. The 'Reference point' names the measurement points from the car, and vary depending on each station. However, the 'Measurement type' and 'Car position side' are standardized as follows:

- **The measurement type:**
  - D - Measurement (Deviation from the Nominal)
  - UT - Upper Tolerance Limit
  - LT - Lower Tolerance Limit
  - US - Upper Specification Limit
  - LS - Lower Specification Limit

  - UR - Upper Reject Limit
  - LR - Lower Reject Limit

- **Car Position Side (relative to a back reference):**
  - LH - Left Side
  - RH - Right Side

These measurements verifies the primary conditions to the vehicle to adheres to design and engineering specifications before moving further along the assembly line. The next section presents the requirements for both data analysis and the proposed solutions, followed by the Exploratory Data Analysis and also the results from the respective analysis for each station. The data exploration reports and results are grounded into the following analysis presented in appendix A.

3.2 Requirements

The system requirements are structured to meet the specific objectives of the project. These are categorized into functional requirements (FR), which define the system's core functionalities, and non-functional requirements (NFR), which address performance, usability, and other operational characteristics.

These requirements are influenced by factors such as the type of system being developed, its target users, and the developer's overall approach. In contrast, non-functional requirements pertain to aspects that are not directly tied to the system's specific services. In this work, functional requirements focus on defining the micro-services operations and structure, while non-functional requirements establish the environment and constraints under industrial manufacturing. Sommerville [18] explains that a system's functional requirements specify the tasks it is expected to perform.

3.2.1 Analysis Requirements

The analysis requirements are focused in the exploratory data analysis and the methodology applied to retrieve useful insights and understanding from the industry collected data.

Functional Requirements

- FR01 - The analysis must contain the data first impressions.
- FR02 - The analysis must contain an analysis of duplicates.
- FR03 - The analysis must contain an analysis of missing data.
- FR04 - The analysis must contain a use case specific analysis considering the physical position of the measurements in the car.
- FR05 - The analysis must contain the analysis of symmetrical and structurally related measurements.

Non-Functional Requirements

- NFR01 - The analysis must contain the data annotation.
- NFR02 - The analysis must contain a correlation analysis.
- NFR03 - The analysis must contain a time series analysis.
- NFR04 - The analysis must contain the exploratory analysis from measurement station 1, 2 and 3.
- NFR05 - The analysis must establish a data selection protocol.
- NFR06 - The analysis must establish a data cleaning protocol.
- NFR07 - The analysis must establish a data transformation protocol.
- NFR08 - The analysis must apply a Data Mining algorithm.

3.2.2 System Requirements

The system requirements otherwise are focused in the micro-service solutions developed to achieve the objective goals, such as the monitoring and diagnosis strategies for Zero Defect Manufacturing.

Functional Requirements

- FR06 - The systems must run within the openZDM infrastructure.
- FR07 - The systems must implement monitoring and diagnosis solutions to achieve Zero Defect Strategies.
- FR08 - The systems must be developed as a micro-service.
- FR09 - The systems must generate a reliable output.
- FR10 - The systems must be tested in a machine that replicates the resource from the shop-floor production machines.
- FR11 - The rule-based monitoring system must allow the end users the easy configuration of rules.
- FR12 - The rule-based monitoring system must have a lower processing time elapsed than the data collection frequency from the measurement stations.
- FR13 - The rule-based monitoring system must generate useful and reliable rule triggering results.
- FR14 - The diagnosis system must have a lower processing time elapsed than the data collection frequency from the measurement stations.
- FR15 - The diagnosis system must classify the incoming data.
- FR16 - The diagnosis system must retrieve information about cause, consequence and possible solution to the classified data.

Non-Functional Requirements

- NFR09 - The systems must contain a data persistence solution in order to save the source information in case of any failure in the infrastructure.
- NFR10 - The rule-based monitoring system must be developed as a framework for rule definition.
- NFR11 - The rule-based monitoring system must be adjustable to any new data source through a data handler.
- NFR12 - The rule-based monitoring system must have support to context and non-context specific rules.
- NFR13 - The rule-based monitoring system must have support to Machine Learning rules.
- NFR14 - The rule-based monitoring system must have configurable parameters for the rules.
- NFR15 - The diagnosis system must consider the industry use case documents.
- NFR16 - The diagnosis system must have ways of identifying defect.
- NFR17 - The diagnosis system must calculate the cars and measurement qualities.
- NFR18 - The diagnosis system must classify the cars and the measurement system based in quality evaluations.

3.3 Data Exploration

The Exploratory Data Analysis was conducted in 3 datasets each of those corresponding to a measurement station, aiming to explore the structure, distribution, and relationships within the dataset, with the goal of extracting meaningful insights for further analysis and interpretation. To achieve better results a use case specific methodology, based mainly on

industry documents, was followed throughout the analysis. This methodology involved four key steps: data structure evaluation, data cleaning, data consistence within the industrial documents, a statistical evaluation, and a correlation analysis.

At the end of each station report, a use case specific plotting tool was also develop to extract valuable insights, focused on several key areas: comparing related and symmetrical measurements of the car, identifying outliers based on the defined industry thresholds, and detecting anomalies. Additionally, it incorporated a seasonal analysis and decomposition to uncover patterns over time. Each of these analytical steps were elaborated to provide a deeper understanding of the data, offering a comprehensive approach to both performance evaluation and anomaly detection across measurement stations.

3.3.1 Measurement Station 1

The Measurement Station 1, includes 35.379 instances and 4.018 columns, from X, Y, Z measurements over 05/12/2023 to 15/03/2024. This dataset primarily contained 3.993 continuous quantitative measurements, many of which corresponded to laser measurement and threshold features, alongside with 15 nominal and discrete timestamp features, and 10 columns with missing names. The measurement point names from measurement station 1 are described as follows:

- PMPs - Cardinal coordinates of relevant points in the car structure.
- FMs - Cardinal distance between two points.

A significant portion of the analysis was dedicated to assessing missing values across the dataset, this was done because the dataset exhibited a high degree of sparsity in some columns, with certain columns containing only NaNs (Not a number) values and some containing around 5.000 missing values. The graphs 3.2 below revealed an uneven distribution of NaNs in the dataset, one group containing less than 15% of missing data, exemplifying the critical points to be measure and another group with above 80% of missing data, showing situational measurements. This analysis helped inform the selection of features to retain for further analysis based on a threshold for missing values.

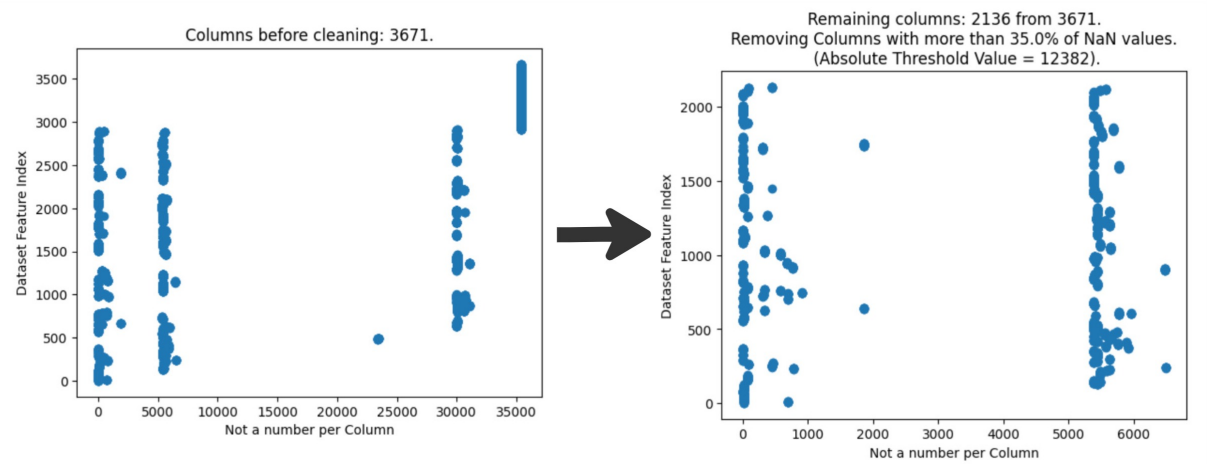


Figure 3.2: Measurement Station 1 - Cleaning Results

The dataset contained numerous X, Y, Z measurement columns, each representing deviations or thresholds in the car manufacturing dimensions. Statistical summaries of these columns provided key insights, revealing:

- **Days with Measurements:** 85 days from a period of 100 days.
- **Cars Frequency:** 416.22 Cars per day.
- **Measurement Means:** Most columns had means near zero, indicating slight variations around nominal values.
- **Standard Deviations:** Some features exhibited higher variability, with a maximum standard deviation of 0.78 in certain columns. These high-variance columns suggested possible areas of interest for further quality control or performance issues.
- **Range of Values:** The largest differences between maximum and minimum values were recorded, with deviations as large as 13.01 units in specific columns. These substantial ranges may indicate outliers or potential areas of concern in production measurements.

A correlation analysis was conducted among the measurement columns to uncover potential relationships. Figure 3.3 below shows a heatmap visualization that was generated to highlight correlations between features, allowing for the identification of strongly

correlated variables. This step was crucial for selecting relevant features for predictive modeling of the micro-services and other critical points in the car for further systems. The multivariate analysis besides the difficulties of a high number features revealed that the data is highly unstable in terms of collinearity, with one group showing possible duplicated features with a perfect multicollinearity, and another group of features with a really low correlation.

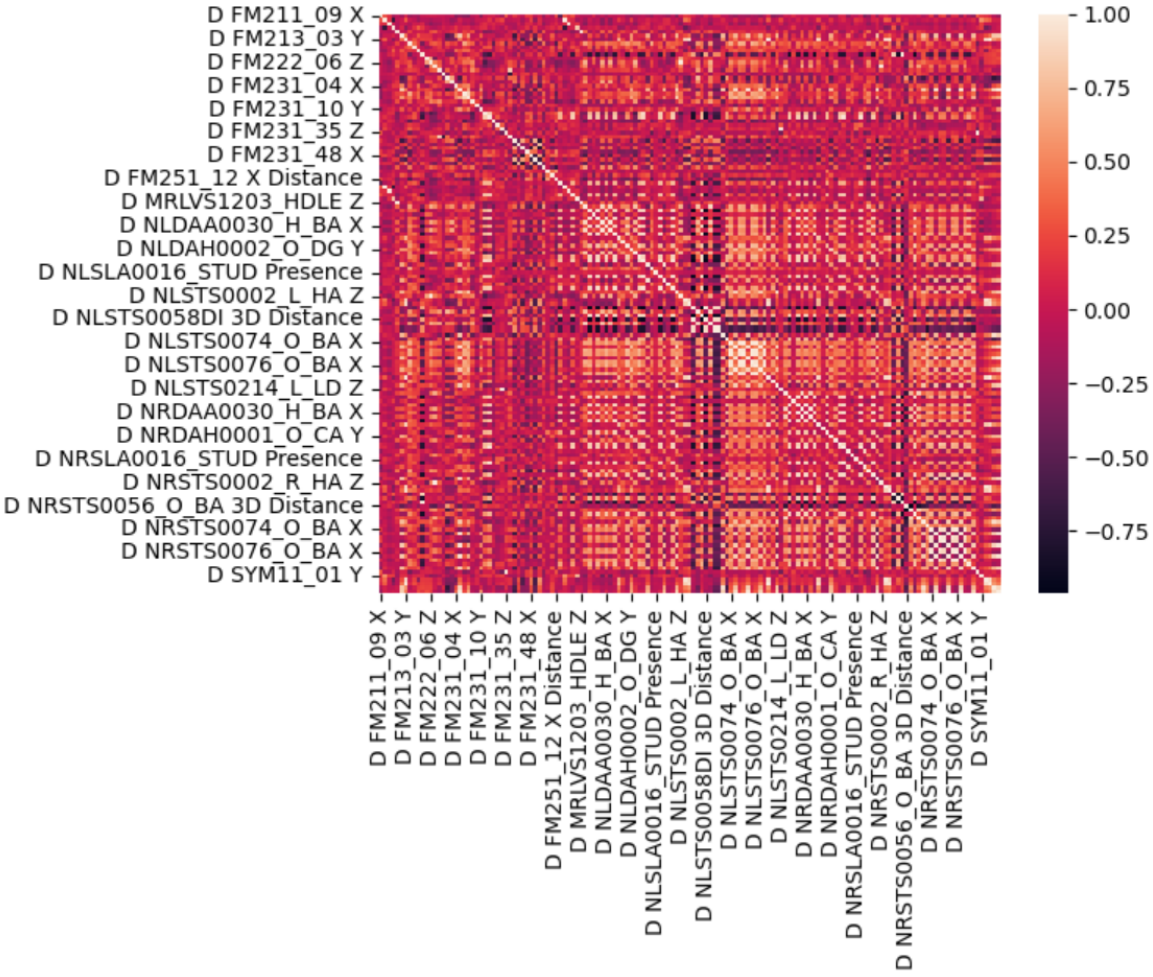


Figure 3.3: Measurement Station 1 - Pearson Variable Correlation

In the measurement station 1 the measurements are not symmetrically divided in the data, so the first analysis was dedicated to the detection of anomalies, this analysis showed that several measurements have some really high deviations from the nominal, without a

common pattern, showing a possible de-calibration or failure in the measurement system, figure 3.4 shows and example of a failure in two of these measurements. The seasonal analysis and decomposition showed at figure 3.5 revealed that the data does not follow a pattern over time, empirically this means that the de-calibration could not be related with time, for this case study.

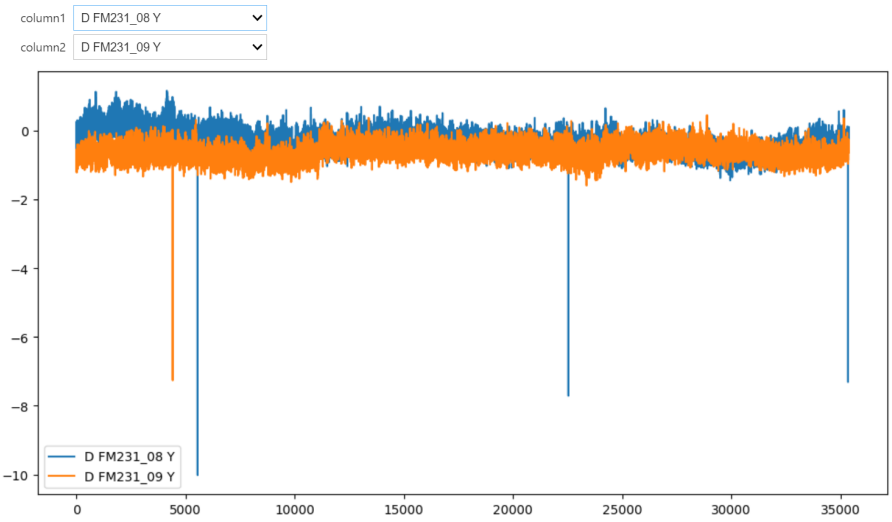


Figure 3.4: Measurement Station 1 - Measurement Evaluation

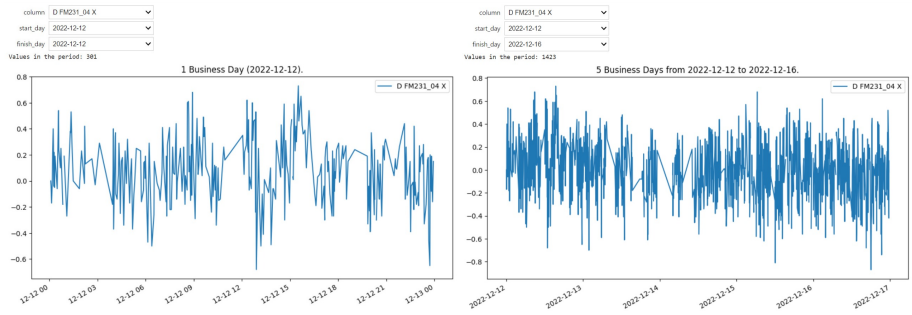


Figure 3.5: Measurement Station 1 - Seasonal Analysis

3.3.2 Measurement Station 2

The Measurement Station 2, includes 56.587 instances and 1.033 columns, from Gap and Flush measurements over 05/12/2023 to 15/03/2024. This dataset primarily contained 1.001 continuous quantitative measurements, many of which corresponded to laser

measurement and threshold features, alongside with 17 nominal and discrete timestamp features, and 5 columns with missing names. The measurement point names from Measurement Station 2 represents the following points in the car structure:

- Fender - is in between the car door and the front bumper.
- Delta - points near the windshield.
- Pilar A - is the forward-most pillar on a vehicle, supporting its roof at each corner of the windshield.
- Pilar B - is between a vehicle's front and rear side glass, where it serves as a structural support of its roof.
- Pilar C - is the rearmost part.

As with Measurement Station 1, the initial step in the analysis focused on evaluating the presence of missing values across the dataset. Some columns also containing only NaN values, while others had up to 6,000 missing entries. The chart 3.6 below highlights the uneven distribution of missing data, however differently from the measurement station 1, the measurement station 2 also showed to have a high quantity of missing data around the 20% of the dataset as well as the other two groups of missing data.

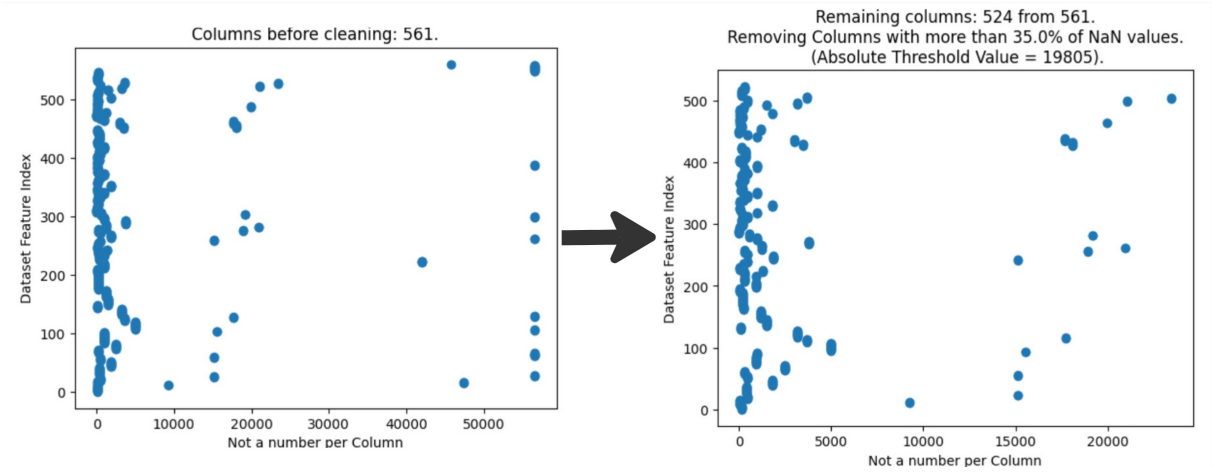


Figure 3.6: Measurement Station 2 - Cleaning Results

The dataset contained numerous measurement columns from the Gaps and Flush, each representing thresholds or deviations from the car manufacturing dimensions. Statistical summaries of these columns provided key insights, revealing:

- **Days with Measurements:** 81 days from a period of 100 days.
- **Cars Frequency:** 698.60 Cars per day.
- **Measurement Means:** Similarly most columns had means near zero, indicating slight variations around nominal values.
- **Standard Deviations:** The maximum standard deviation was 0.94 in certain columns. These high-variance columns suggested possible areas of interest for further quality control or performance issues.
- **Range of Values:** The largest differences between maximum and minimum values were recorded, with deviations as large as 23.11 units. These ranges indicates outliers or potential areas of concern in production measurements.

The correlation analysis, as shown in Figure 3.7, revealed that the data from Measurement Station 2 exhibits very low correlations. This result is somewhat unexpected for Gap and Flush data, where one would anticipate certain features to display related deviations. For instance, symmetrical measurements—such as the Gap between the A-pillar and windshield—are expected to demonstrate a linear relationship, where an increase in the Gap on the left side corresponds to a decrease on the right. However, this anticipated relationship was not observed in the analysis.

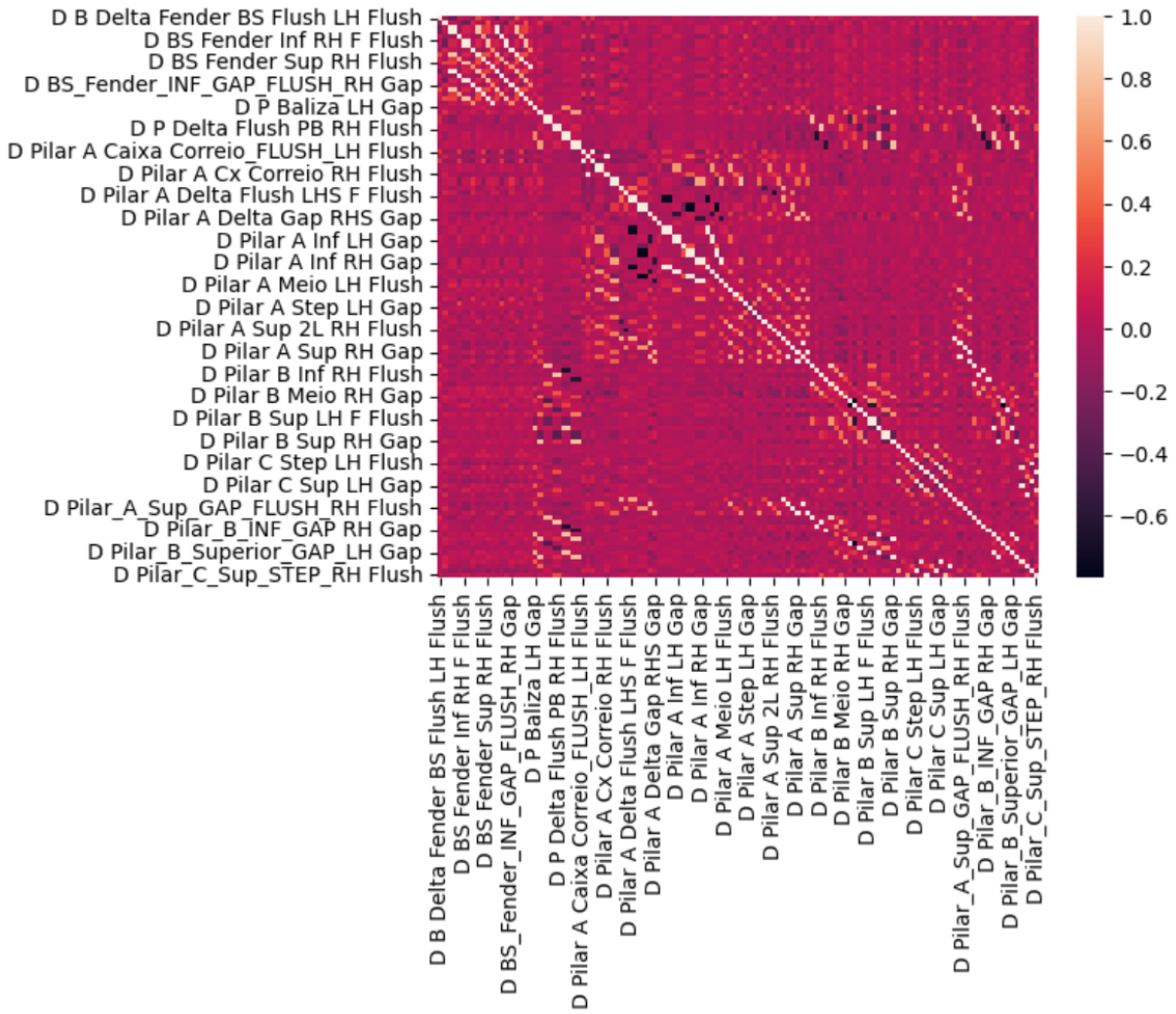


Figure 3.7: Measurement Station 2 - Pearson Variable Correlation

Unlike the previous station, Measurement Station 2 is symmetrically divided based on the car's positioning, allowing for the analysis of symmetrically related measurements, however as exemplified in Figure 3.8 most measurements presented a non-linear behavior. Similar to Measurement Station 1, the data from Measurement Station 2 does not exhibit a consistent pattern over time, as shown in Figure 3.9 the seasonal decomposition is almost similar as the observed feature. This lack of regularity in the data further emphasizes the complex and unpredictable nature of the measurements at this station.

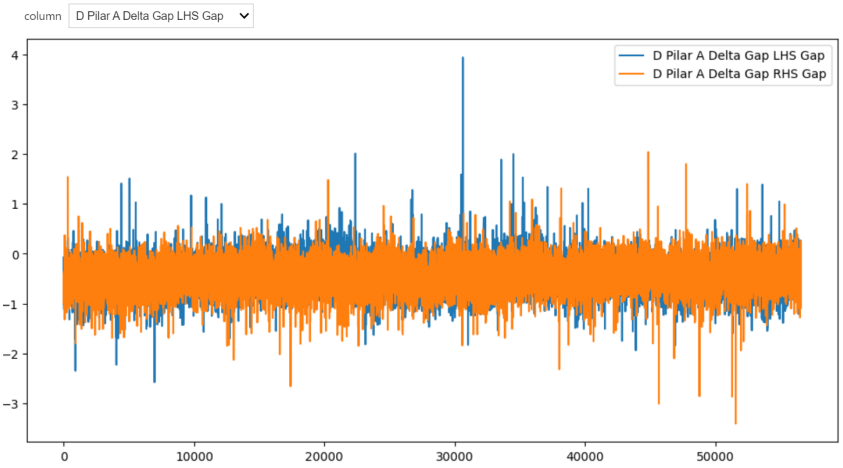


Figure 3.8: Measurement Station 2 - Symmetrical Measurements Analysis

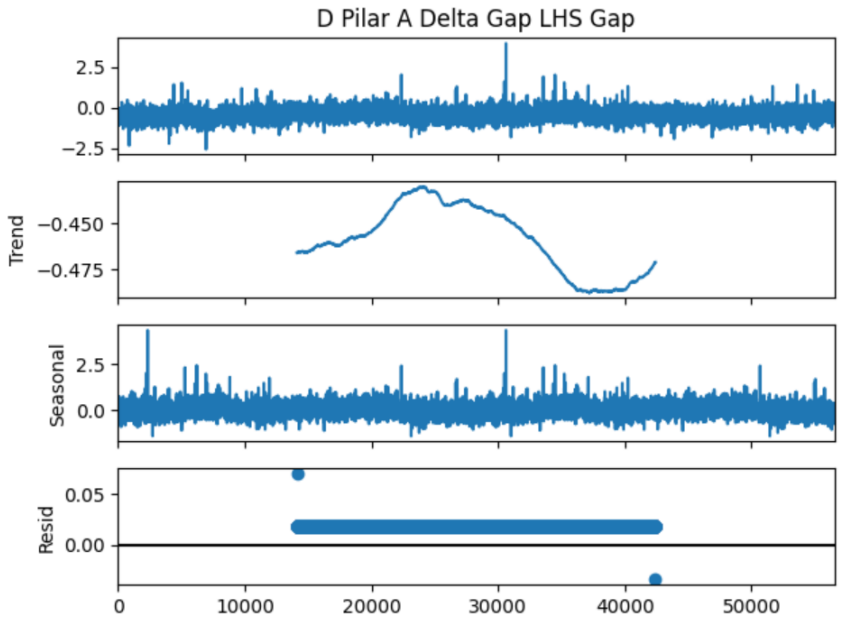


Figure 3.9: Measurement Station 2 - Seasonal Decomposition

3.3.3 Measurement Station 3

The Measurement Station 3, includes 62.552 instances and 465 columns, from Gap and Flush measurements over 05/12/2023 to 15/03/2024. This dataset primarily contained 448 continuous quantitative measurements, many of which corresponded to laser measurement and threshold features, alongside with 14 nominal and discrete timestamp features,

and 2 columns with missing names. The measurement point names from Measurement Station 3 represents the following points in the car structure:

- Fender - is in between the car door and the front bumper.
- Capot - is the car lid and/or the hood of the motor.
- Roof - is the car roof or it supports.

The analysis of missing columns revealed that more than half of the dataset contained only NaN values, while others had up to 6,000 missing entries, showing that a substantial part of the data is missing in this station. The chart 3.10 below highlights that differently from the other measurement stations only a few columns have a high quantity of missing data after the removing of only NaN columns.

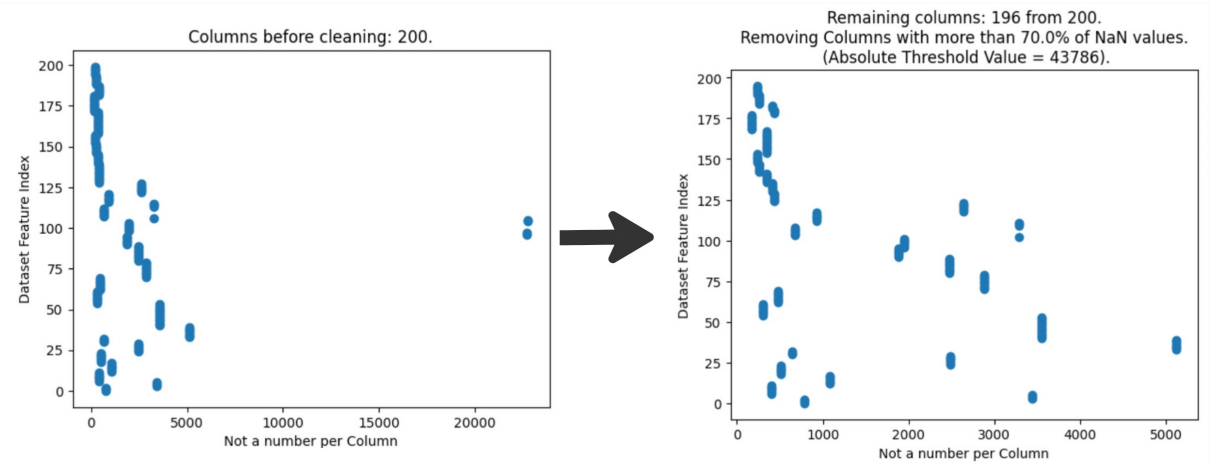


Figure 3.10: Measurement Station 3 - Cleaning Results

The dataset contained numerous measurement columns from the Gaps and Flush, each representing thresholds or deviations from the car manufacturing dimensions. Statistical summaries of these columns provided key insights, revealing:

- **Days with Measurements:** 84 days from a period of 100 days. Exemplifying that there are days in which only the measurement station 2 collected data.

- **Cars Frequency:** 744.66 Cars per day, this reveals measurement station 2 is more efficient in measuring cars than the other station, this relation can happen due to the fact that this is the station with lower amount of points to be measure.

- **Measurement Means:** Similarly most columns had means near zero, indicating slight variations around nominal values.

- **Standard Deviations:** The maximum standard deviation was 0.99 in certain columns. These high-variance columns suggested possible areas of interest for further quality control or performance issues.

- **Range of Values:** The largest differences between maximum and minimum values were recorded, with deviations as large as 28.43 units. These ranges indicates outliers or potential areas of concern in production measurements.

The correlation analysis, as shown in Figure 3.11, revealed a similar scenario to Measurement Station 2 exhibiting very low correlations. This result is somewhat unexpected for Gap and Flush data, where one would anticipate certain features to display related deviations. For instance, symmetrical measurements—such as the Gap between the A-pillar and windshield—are expected to demonstrate a linear relationship, where an increase in the Gap on the left side corresponds to a decrease on the right. However, this anticipated relationship was not observed in the analysis.

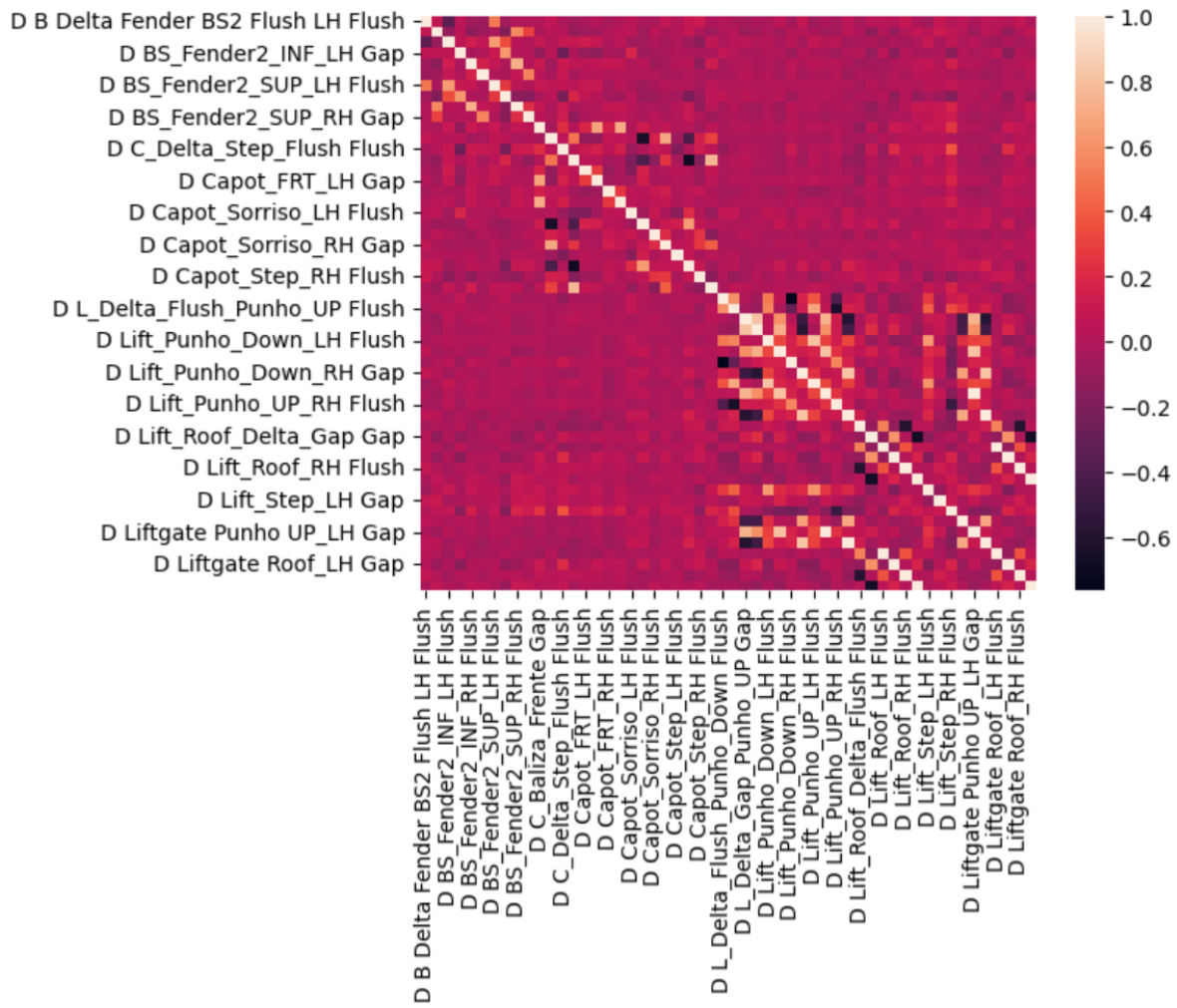


Figure 3.11: Measurement Station 3 - Pearson Variable Correlation

The symmetrical analysis and anomaly detection for Measurement Station 3 yielded results similar to those of the previous station. However, new insights regarding threshold behavior emerged that warrant attention. Specifically, some measurements were missing critical thresholds, while others had thresholds placed incorrectly, which resulted in an all-rejection scenario. Figure 3.12 highlights the missing thresholds and Figure 3.13 demonstrate the incorrect threshold placements. These errors significantly impacted the data's accuracy and led to further complications in the rejection process.

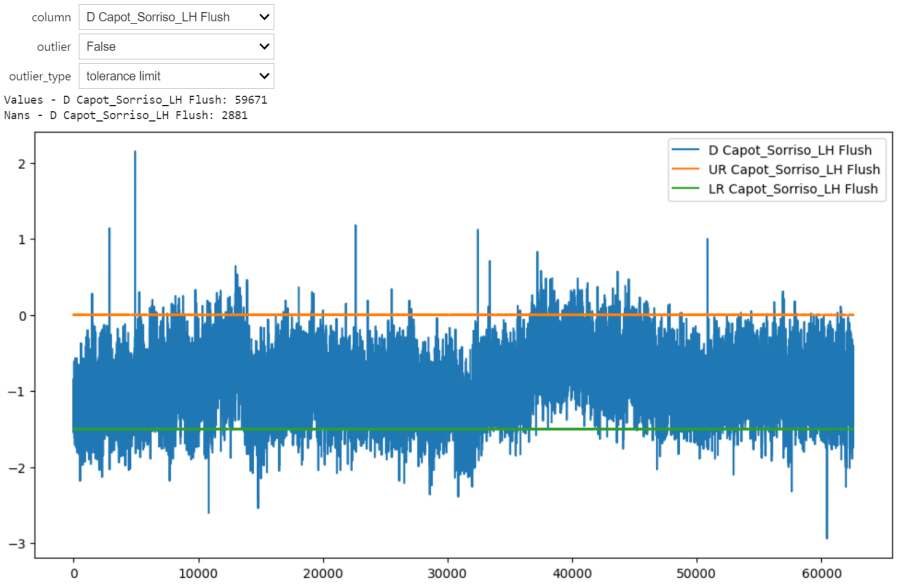


Figure 3.12: Measurement Station 3 - Missing Threshold

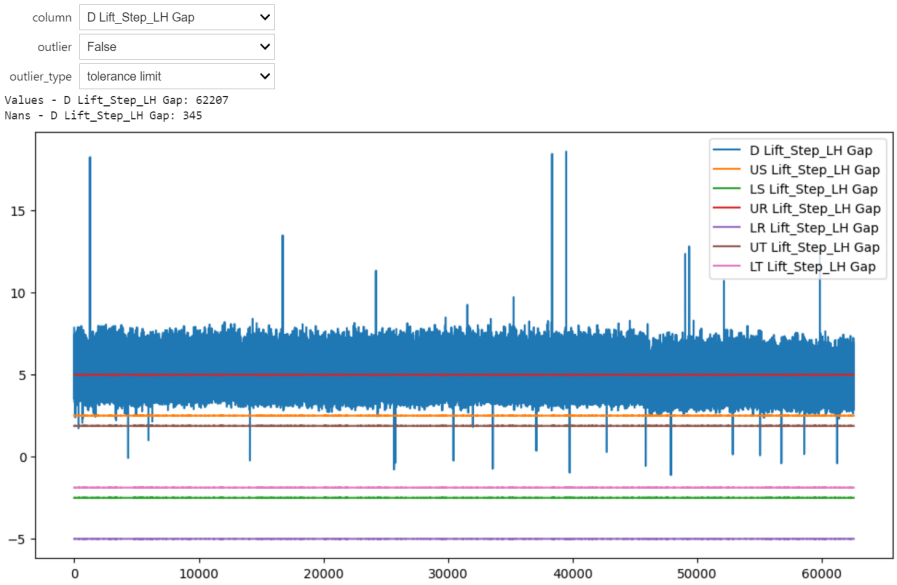


Figure 3.13: Measurement Station 3 - Wrong Threshold Placement

3.3.4 Relation between Measurement Station 2 and 3

Since the measurements from Stations 2 and 3 are empirically related with both representing Gap and Flush at different points in the car body structure, an analysis was conducted

to uncover potential relationships between the two stations measurements. By examining the identifier features, it was observed 3 candidates for merging the datasets. However, the main car identifier was selected because the other two candidates had respectively no values in the Measurement Station 3 dataset, and missing values in both datasets. Figure 3.14 shows the Venn diagram of the merge using unique values from the identifier feature between the two stations. A total of 7.112 cars, representing 12% of the 55.739 cars from Measurement Station 2, were not included in the merge. Similarly, 13.642 cars, or 21% of the 62.269 cars from Measurement Station 3, were also excluded from the merge. These information was forwarded to the industry focal points to likely apply correction in the process behavior.

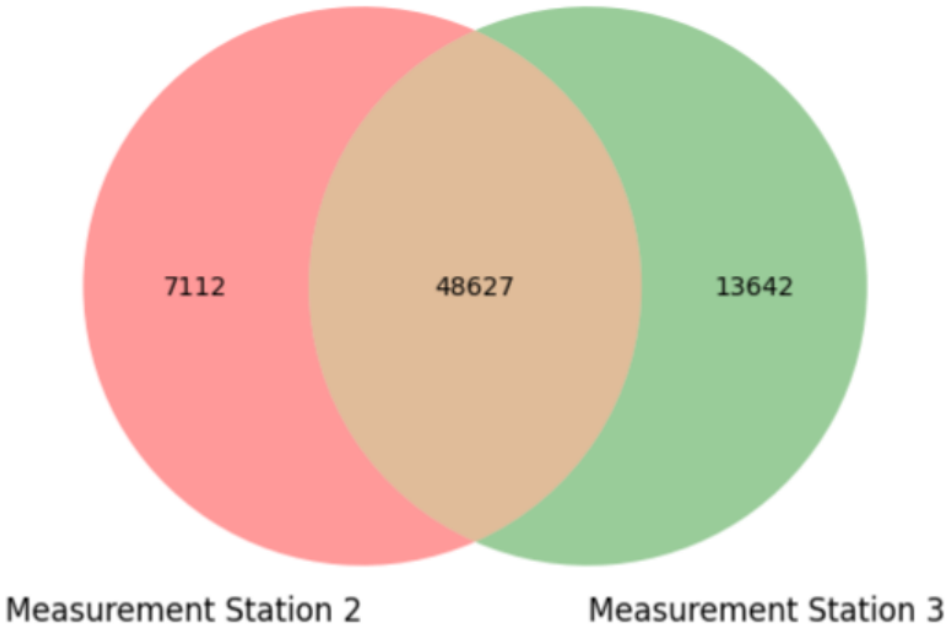


Figure 3.14: Merge between Measurement Station 2 and 3, using the identifier feature.

An analysis of the non-merged data revealed that the identifier exhibited significant continuity issues, with values frequently set to 0 or sometimes reaching the maximum 16-bit float value. It was concluded that these irregularities were due to a process problem, likely caused by some missing readings of the car label after the measurement process. Figure 3.15 below shows the identifier feature behavior after removing this inconsistencies.

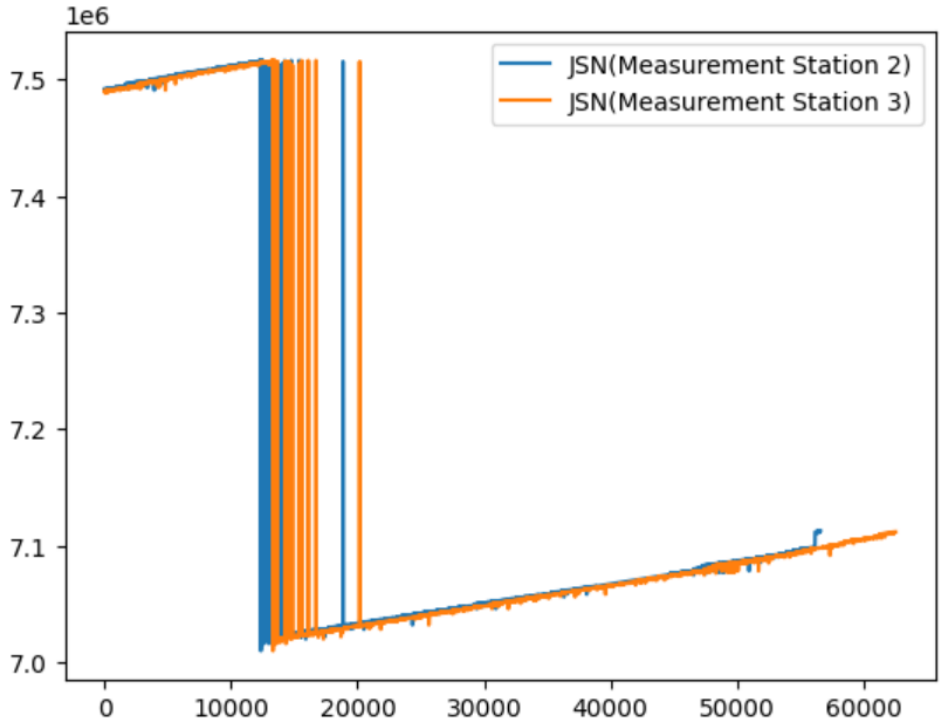


Figure 3.15: Identifier evolution over time between Measurement Station 2 and 3

3.4 Applying Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) is essential for extracting valuable insights from massive, complex datasets. It involves a systematic approach to transform raw data into actionable knowledge. KDD addresses challenges like noise, missing data, and irrelevant information. Key steps include data selection, preprocessing, transformation, Data Mining, and interpretation. This process helps uncover hidden patterns, trends, and relationships within the data.

The KDD process is especially important in this case study, as it helps gather new knowledge around the relationship between the data, this will helping in making data-driven decisions, optimize processes, and uncover non-common patterns, as well giving insights about data transformations and cause, consequence relations in the data to further help the diagnosis objective. The following sub-sections presented the application of these

methodology in the raw dataset from Measurement Station 2 as this dataset showed to be the more consistent one in terms of non-missing information.

3.4.1 Data Selection

The KDD process begins with selecting relevant subsets of data, ensuring that the focus remains on features most likely to yield insights. This step aim to reduce the complexity of the analysis by filtering out irrelevant or redundant information. To achieve that the following steps were applied:

- Drop unnamed features from the dataset.
- Drop the identifier features, and saving separately the main identifier and Timestamp features.
- Drop features with 100% correlation, leaving just one example (This step is really important because high correlated features can further interfere in the Data Mining algorithms).

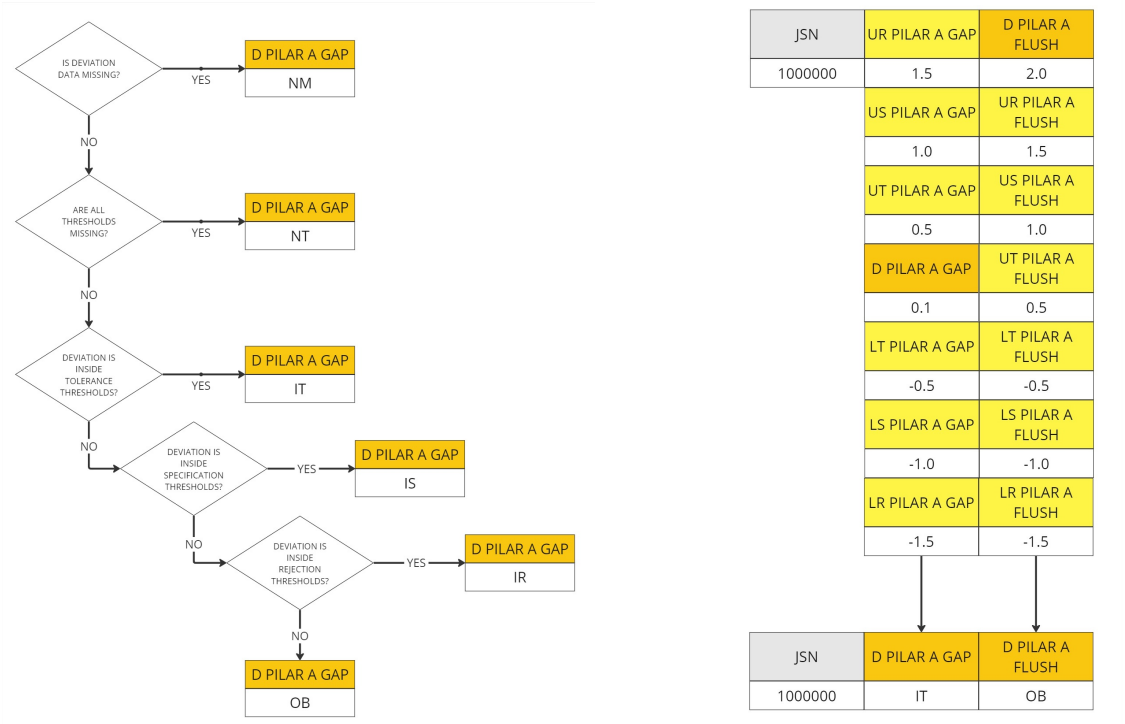
3.4.2 Data Preprocessing

To deal with the missing values, noise, or inconsistencies that can distort the results. The preprocessing phase addresses these issues by cleaning the data, normalizing values, and handling outliers. This step improves data quality and ensures more accurate and reliable outcomes in subsequent analysis stages. To achieve that the following cleaning and splitting methodology were applied to each measurement station dataset:

- Features with only NaN values were removed.
- The remaining dataset was splited between the Deviation to the Nominal and the Threshold values.

3.4.3 Data Transformation

In this step, the data was transformed and consolidated into a format suitable for mining. The Deviation and Threshold features were aggregated and reshaped to highlight key relationships, making patterns easier to detect. As Spiegel notes in The Art of Statistics: How to Learn From Data (pag. 38) [35], categorical variables can be viewed as "Numbers that have been grouped, such as obesity levels, often defined by thresholds for body mass index (BMI)". A similar transformation methodology was applied here. This process ensured the data was optimized for efficient mining of valuable information. Figure 3.16 illustrates the decision model used to convert numerical features into more meaningful categorical ones, along with an example of the model in action.



(a) Data Transformation Model

(b) Data Transformation Example

Figure 3.16: Data Transformation

After transforming the dataset (3.17) a exploratory analysis was done to retrieve new hidden patterns in the new categorical values, the first analysis consisted in counting the

occurrence of each label in the measurement and the following results were observed in figure 3.18.

	JSN	timestamp	D Pilar A Delta Gap LHS Gap	D Pilar A Cx Correio LH Gap	D Pilar A Cx Correio LH Flush	D Pilar B Step LH Gap	D Pilar B Step LH Flush	D Pilar C Inf RH Gap	D Pilar C Inf RH Flush	D Pilar A Cx Correio RH Gap	...	D SG Pilar A UP RH Gap	D SG Pilar A UP RH Flush	D SG Pilar B UP RH Gap	D SG Pilar B UP RH Flush	D SG Pilar A Down RH Gap	D SG Pilar A Down RH Flush	D SG Pilar A Down LH_new Gap	D SG Pilar A Down LH_new Flush	D SG Pilar B UP LH Gap	D SG Pilar B UP LH Flush
0	7491659.0	2022-12-06 15:24:39	IR	IT	IS	IS	OB	IT	IT	IT	...	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM
1	7491660.0	2022-12-06 15:25:52	IT	IT	IS	IS	OB	IT	IT	IT	...	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM
2	7491661.0	2022-12-06 15:27:04	IR	IT	IS	IS	OB	IT	IT	IT	...	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM
3	7491662.0	2022-12-06 15:32:05	IT	IT	IS	IS	IS	IT	IT	IT	...	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM
4	7491663.0	2022-12-06 15:33:17	IT	IT	IS	IS	IS	IT	IS	IT	...	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM

5 rows × 142 columns

Figure 3.17: Measurement Station 2 Data Transformed.

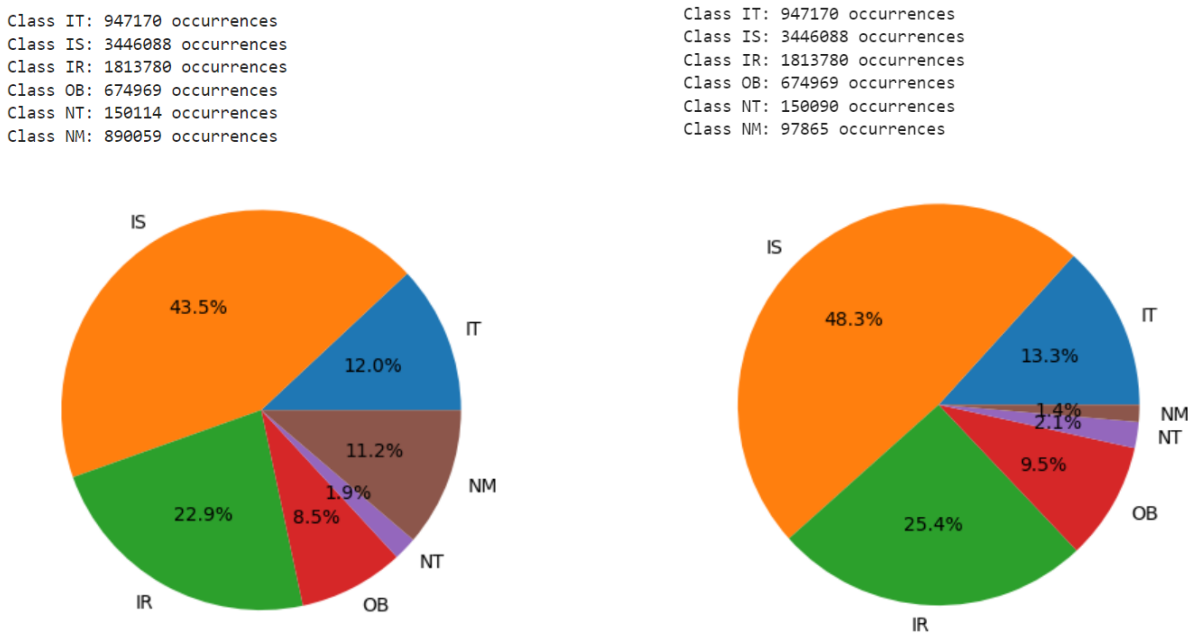
	D Pilar A Delta Gap LHS Gap	D Pilar A Cx Correio LH Gap	D Pilar A Cx Correio LH Flush	D Pilar B Step LH Gap	D Pilar B Step LH Flush	D Pilar C Inf RH Gap	D Pilar C Inf RH Flush	D Pilar A Cx Correio RH Gap	D Pilar A Cx Correio RH Flush	D Pilar B Sup LH Gap	...	D SG Pilar A UP RH Gap	D SG Pilar A UP RH Flush	D SG Pilar B UP RH Gap	D SG Pilar B UP RH Flush	D SG Pilar A Down RH Gap	D SG Pilar A Down RH Flush	D SG Pilar A Down LH_new Gap	D SG Pilar A Down LH_new Flush	D SG Pilar B UP LH Gap	D SG Pilar B UP LH Flush
IT	30090	56526	0	0	28	52018	26375	54746	0	31	...	0	0	0	0	0	0	0	0	0	0
IS	7472	0	43844	56122	33040	3392	12903	0	0	33332	...	0	0	0	0	0	0	0	0	0	0
IR	15621	0	8845	0	14612	735	16620	0	47010	16521	...	0	0	0	0	0	0	0	0	0	0
OB	3271	0	3837	18	8460	53	300	0	7736	6231	...	0	0	0	0	0	0	0	0	0	0
NT	0	0	0	0	0	0	0	0	0	0	...	2	2	2	2	1	1	1	1	2	2
NM	133	61	61	447	447	389	389	1841	1841	472	...	56585	56585	56585	56585	56586	56586	56586	56586	56585	56585

6 rows × 140 columns

Figure 3.18: Occurrences by each class in the Labeled Data.

From the analysis, we observe that the last columns predominantly contain the "No Measurement" class. This suggests that these features may have been introduced at a specific point in time for particular testing purposes. Supporting this assumption is the fact that, even in cases where data wasn't classified as "No Measurement", much of it was labeled as "No Threshold", indicating that the measurements were incomplete or loosely defined. This pattern poses a challenge for subsequent Data Mining techniques in the KDD process, as such features can negatively impact the accuracy and effectiveness of the analysis. To mitigate this, a cleaning strategy was implemented. All columns where

more than 95% of the instances were classified as "No Measurement" were removed, the option to use 95% aims to remove the loose measurement made only in some cars, leaving in the dataset the columns where data was randomly not collected. The distribution of classes before and after this cleaning process is shown in Figure 3.19 below.



(a) Distribution Before Cleaning

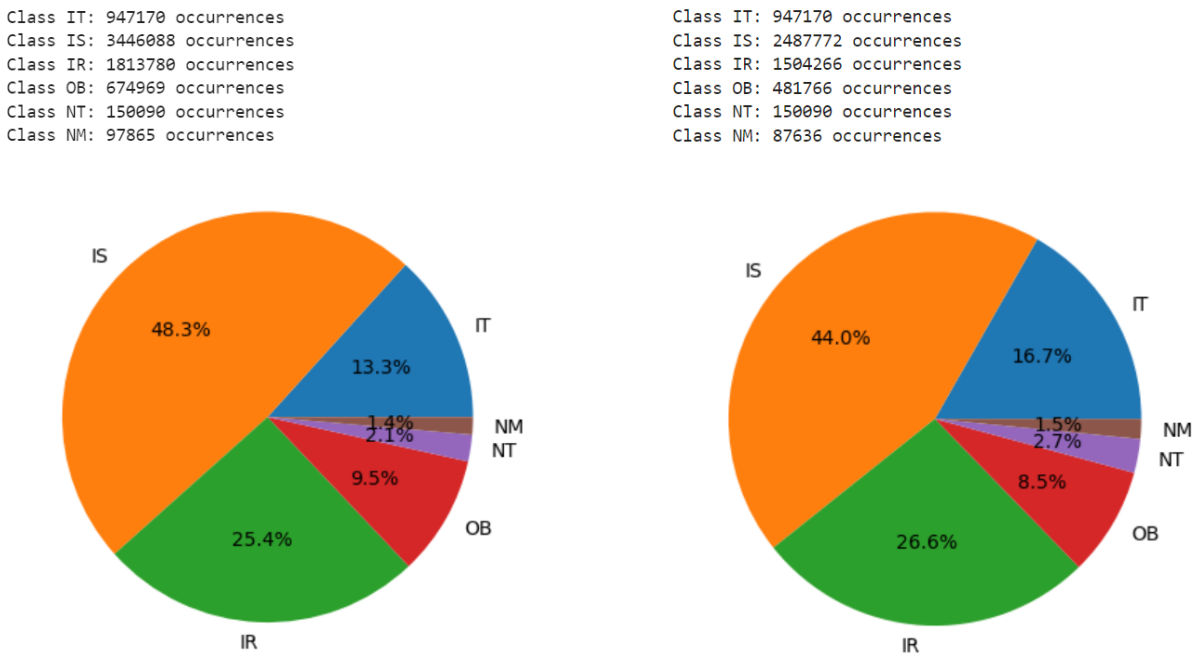
(b) Distribution After Cleaning

Figure 3.19: Labeled data cleaning, based of high frequency of No Measurements.

Another issue that can negatively impact Data Mining techniques is the presence of highly correlated features, which may indicate duplicated data or linear relationships between variables. To address this, a table of feature pairs with correlations above 95% was generated, as shown in Figure 3.20. Upon evaluating the feature names, and raw deviation data values, the high correlations confirmed the earlier hypothesis regarding duplicated features. These redundant variables not only skew the analysis but can also inflate the class distribution, leading to misleading results. To improve the accuracy of the analysis and prevent class distribution inflation, these highly correlated features were removed. Figure 3.21 illustrates the class distribution before and after the removal of redundant features.

	Feature 1	Feature 2	Correlation	Abs Correlation
200	D Pilar A Cx Correio LH Gap	D Pilar A Cx Correio LH G Gap	1.000000	1.000000
924	D Pilar A Cx Correio RH Gap	D Pilar A Cx Correio RH G Gap	1.000000	1.000000
3828	D Pilar B Inf RH Flush	D Pilar_B_INF_FLUSH RH Flush	1.000000	1.000000
3430	D Pilar A Inf RH Gap	D Pilar A Inferior_GAP_FLUSH_RH Gap	1.000000	1.000000
1469	D Pilar A Sup RH Flush	D Pilar_A_Sup_GAP_FLUSH_RH Flush	1.000000	1.000000
4314	D Pilar A Inf LH Flush	D Pilar A Inferior_GAP_FLUSH_LH Flush	1.000000	1.000000
4077	D BS Fender Inf RH Gap	D BS_Fender_INF_GAP_FLUSH_RH Gap	1.000000	1.000000
3524	D Pilar A Inf RH Flush	D Pilar A Inferior_GAP_FLUSH_RH Flush	1.000000	1.000000
7347	D P Delta Flush PB RH F Flush	D P Delta Flush PB RH Flush	1.000000	1.000000
6085	D BS_Fender_Sup_GAP_FLUSH_LH Gap	D BS Fender Sup LH Gap	1.000000	1.000000
6145	D BS_Fender_Sup_GAP_FLUSH_LH Flush	D BS Fender Sup LH Flush	1.000000	1.000000
5494	D P Delta Flush PB LH Flush	D P Delta Flush PB LH F Flush	1.000000	1.000000
6192	D BS_Fender_INF_GAP_FLUSH_LH Gap	D BS Fender Inf LH Gap	1.000000	1.000000
2981	D Pilar B Inf LH Flush	D Pilar_B_INF_FLUSH LH Flush	1.000000	1.000000
6250	D BS_Fender_INF_GAP_FLUSH_LH Flush	D BS Fender Inf LH Flush	1.000000	1.000000
1355	D Pilar A Sup RH Gap	D Pilar_A_Sup_GAP_FLUSH_RH Gap	1.000000	1.000000

Figure 3.20: Labeled Correlation Data.



(a) Distribution Before Cleaning

(b) Distribution After Cleaning

Figure 3.21: Labeled data cleaning based on correlation analysis, and duplicated features.

3.4.4 Data Mining

For the Data Mining step the Apriori algorithm [2] was applied to the preprocessed data to uncover frequent itemsets and generate association rules. As one of the most widely used algorithms in Data Mining, Apriori is particularly effective for discovering relationships between variables in large datasets. It works by identifying frequent individual items and extending them to larger itemsets, as long as these maintain a minimum level of support. The algorithm is instrumental in finding hidden associations, such as relationships between certain measurement deviation labels across different car parts. By revealing these patterns, the Apriori algorithm helps to understand the underlying structure of the data and can be used to predict trends or anomalies, offering critical insights for the overall diagnostic and decision-making process.

Before explaining the two approaches used to generate the association rules, it is important to understand the key metrics used in evaluating these rules:

- **Support:** This metric measures how frequently a particular itemset appears in the dataset. It is calculated as the proportion of records in the dataset that contain the itemset. A higher support value indicates that the itemset is more common, helping to prioritize associations that are broadly relevant.

- **Confidence:** Confidence quantifies the likelihood that a particular association rule will hold true. It is defined as the ratio of the number of records that contain both the antecedent and the consequent, to the number of records that contain only the antecedent. A high confidence value suggests that the rule is often accurate, helping to validate the strength of the relationship.

- **Lift:** Lift evaluates the strength of an association rule by comparing the observed confidence to what would be expected if the antecedent and consequent were independent. A lift value greater than 1 indicates a strong positive association, meaning the occurrence of the antecedent increases the likelihood of the consequent.

These metrics are essential for assessing the quality and significance of the association

rules uncovered by the Apriori algorithm, ensuring that only the most relevant and useful patterns are highlighted.

To generate meaningful association rules, two approaches were tested:

- **Binary Defect Association Rules Extraction:** The first approach focused on simplifying the classifications by grouping the categories "Inside Rejection", "Inside Specification", and "Inside Tolerance" into a single group called "Measurements Without Errors" (0 or False), while combining "Out of Bounds", "No Threshold", and "No Measurement" into a group called "Measurements With Errors" (1 or True). This method removes the severity of deviations but allows for the identification of patterns related to the presence of measurement errors. The minimum support for the item-sets generation was settled to 90% an only rules with a confidence higher than 70% was generated, from those the association rules with the higher confidence are devised in figure 3.22 below.

	antecedents	consequents	confidence	lift	support
0	(D Pilar A Inf RH Angle)	(D Pilar A Inf LH Angle)	1.0	1.0	1.0
1	(D Pilar A Inf LH Angle)	(D Pilar A Inf RH Angle)	1.0	1.0	1.0
2	(D Pilar B Sup LH F Flush)	(D Pilar A Inf RH Angle)	1.0	1.0	1.0
3	(D Pilar A Inf RH Angle)	(D Pilar B Sup LH F Flush)	1.0	1.0	1.0
4	(D Pilar B Sup LH F Flush)	(D Pilar A Inf LH Angle)	1.0	1.0	1.0
5	(D Pilar A Inf LH Angle)	(D Pilar B Sup LH F Flush)	1.0	1.0	1.0
6	(D Pilar B Sup LH F Flush, D Pilar A Inf RH Angle)	(D Pilar A Inf LH Angle)	1.0	1.0	1.0
7	(D Pilar B Sup LH F Flush, D Pilar A Inf LH Angle)	(D Pilar A Inf RH Angle)	1.0	1.0	1.0
8	(D Pilar A Inf RH Angle, D Pilar A Inf LH Angle)	(D Pilar B Sup LH F Flush)	1.0	1.0	1.0
9	(D Pilar B Sup LH F Flush)	(D Pilar A Inf RH Angle, D Pilar A Inf LH Angle)	1.0	1.0	1.0
10	(D Pilar A Inf RH Angle)	(D Pilar B Sup LH F Flush, D Pilar A Inf LH Angle)	1.0	1.0	1.0
11	(D Pilar A Inf LH Angle)	(D Pilar B Sup LH F Flush, D Pilar A Inf RH Angle)	1.0	1.0	1.0

Figure 3.22: Binary defect association rules extraction results.

- **Multi-class Deviation Association Rules Extraction:** The second approach

maintained the original classifications, associating each measurement with its specific classification. This method enables patterns to be more closely tied to specific issues, unlike the first approach, which only highlights the occurrence of errors without distinguishing between their types or severity. The association rules with the higher confidence are devised in figure 3.23 below.

	antecedents	consequents	confidence	lift	support
15	(D Pilar A Inf RH Flush_IS, D BS Fender Inf RH Flush_IS)	(D Pilar A Step RH Flush_IT)	0.988253	1.005344	0.902416
21	(D Pilar A Inf RH Flush_IS, D BS Fender Sup RH Flush_IS)	(D Pilar A Step RH Flush_IT)	0.987688	1.004769	0.905879
4	(D Pilar A Inf RH Flush_IS)	(D Pilar A Step RH Flush_IT)	0.987458	1.004536	0.932228
27	(D BS Fender Sup RH Flush_IS, D BS Fender Inf RH Flush_IS)	(D Pilar A Step RH Flush_IT)	0.987397	1.004473	0.935922
9	(D BS Fender Inf RH Flush_IS)	(D Pilar A Step RH Flush_IT)	0.987175	1.004248	0.950837
12	(D BS Fender Sup RH Flush_IS)	(D Pilar A Step RH Flush_IT)	0.986665	1.003728	0.954495
28	(D Pilar A Step RH Flush_IT, D BS Fender Inf RH Flush_IS)	(D BS Fender Sup RH Flush_IS)	0.984314	1.017489	0.935922
7	(D BS Fender Inf RH Flush_IS)	(D BS Fender Sup RH Flush_IS)	0.984093	1.017260	0.947868
11	(D Pilar A Meio LH Flush_IR)	(D Pilar A Step RH Flush_IT)	0.983804	1.000819	0.901727
26	(D BS Fender Sup RH Flush_IS, D Pilar A Step RH Flush_IT)	(D BS Fender Inf RH Flush_IS)	0.980541	1.018015	0.935922
6	(D BS Fender Sup RH Flush_IS)	(D BS Fender Inf RH Flush_IS)	0.979814	1.017260	0.947868
20	(D Pilar A Inf RH Flush_IS, D Pilar A Step RH Flush_IT)	(D BS Fender Sup RH Flush_IS)	0.971736	1.004487	0.905879
31	(D BS Fender Inf RH Flush_IS)	(D BS Fender Sup RH Flush_IS, D Pilar A Step RH Flush_IT)	0.971690	1.018015	0.935922
2	(D Pilar A Inf RH Flush_IS)	(D BS Fender Sup RH Flush_IS)	0.971510	1.004253	0.917172

Figure 3.23: Multi-class deviation association rules extraction.

3.4.5 New Knowledge

The binary defect association rules revealed in rules 1 and 2 that the symmetrical measurements in the "pilar" region exhibit perfect confidence, support, and lift. This can indicate that whenever a defect occurs in one of the symmetrical measurements, a defect should also be present in the corresponding measurement as well, highlighting a consistent relationship between these features. Another key insight from the association rules number 4 and 5 is the discovery of perfect confidence, support, and lift between the defects in the inverse diagonal points of the door frame, specifically between inferior part of Pilar A and the superior part of Pilar B (See figure 3.24). However, this pattern only occurs on the left side of the car. This suggests a potential issue with the measurements on the right side, as such a relationship is expected to appear symmetrically on both sides of the car.

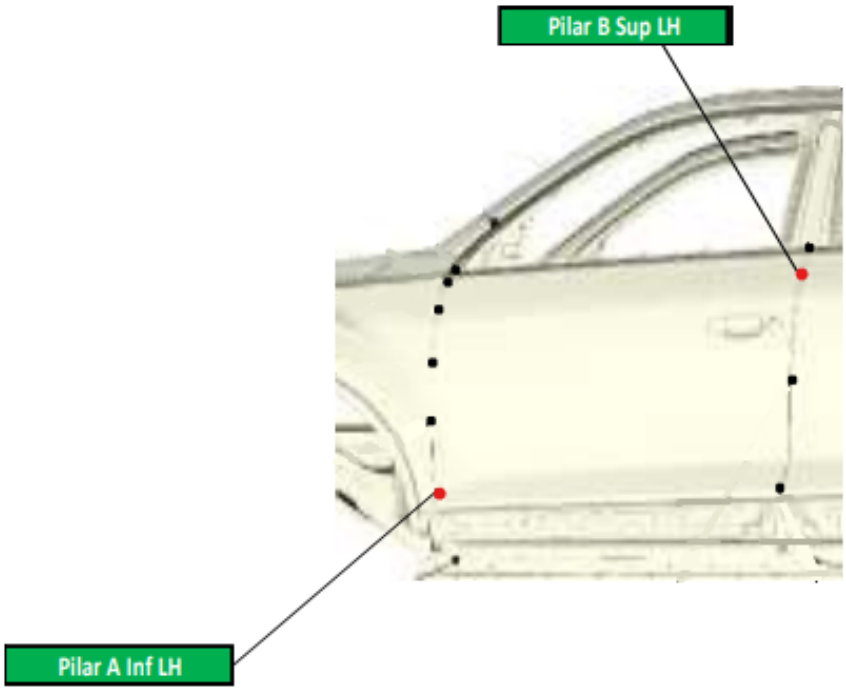


Figure 3.24: Pilar A Inf and Pilar B Sup in the car structure.

In the multi-class deviation association rules analysis, the focus was on the "Inside Rejection" and "Out of Bounds" labels, as these are the most likely to reveal useful patterns indicating measurements that fall outside acceptable limits. However, the only rules involving the "Inside Rejection" classification referred to deviations that exclusively had rejection thresholds, which naturally caused them to be classified as "Inside Rejection".

Chapter 4

Data Analytic Tools for Zero Defect Manufacturing

The growing complexity and scale of modern manufacturing processes have given rise to the need for advanced data analytics tools, particularly in the pursuit of Zero Defect Manufacturing. ZDM aims to eliminate defects at every stage of production, ensuring optimal product quality and reducing waste. To achieve this ambitious goal, real-time data collection, analysis, and feedback mechanisms are crucial. Data analytics tools play a pivotal role by continuously monitoring the production process, identifying potential issues, and providing actionable insights for immediate corrective actions. This architectural evolution introduces several challenges and opportunities, primarily driven by the vast availability of data. This opens the door to developing real-time data analytics tools that convert raw data into actionable knowledge, providing insights for continuous improvements in production systems. Such systems could monitor processes in real-time, enabling early detection of anomalies and supporting predictive maintenance by diagnosing the detected problems. This is particularly important for implementing Zero Defect Manufacturing strategies [9], where the goals include reducing production costs, enhancing product quality, and adding value to the manufacturing process.

4.1 Infrastructure Development Environment

The developed tools are part of the openZDM (Open platform for realizing zero defects in cyber-physical manufacturing) platform (openzdm.eu), which introduced an innovative architectures through the development of a micro-service framework [24], following the state-of-the-art industrial approaches. Figure 4.1 shows the established infrastructure that includes a File Watcher for data collection, an Asset Administration Shell (AAS) for data management, and a Data Lake for data persistence. In this context, this thesis project, proposes the development of two analytical tools designed to process collected data in real-time, with focus on a rule-based monitoring tool, and a fault detection and diagnosis tool, both aiming to achieve a zero defect production line.

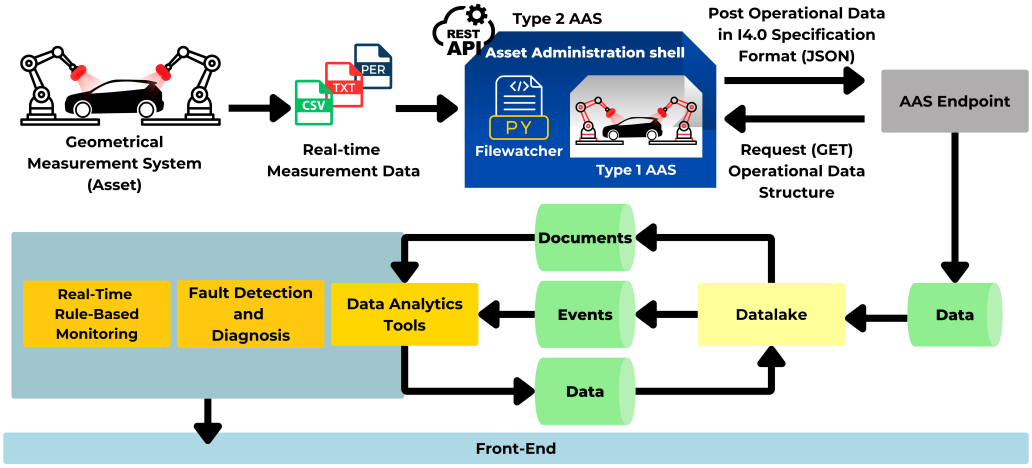


Figure 4.1: OpenZDM infrastructure. (Based on [8] and [39])

The infrastructure was developed in Python by the openZDM team [24], counting with two primary libraries: the NATS Message Broker [7] and the AVRO schema specification [12]. These were encapsulated into a custom library, ozdm, specifically for the project. This ozdm library facilitates seamless integration and interface with the micro-service architecture, enabling efficient communication and data handling across the entire system. It ensures that the various data analytic micro-services can interact smoothly, supporting real-time data processing and analysis in line with the zero defect manufacturing goals of the platform. To simplify the interface, testing and deployment of the micro-services

within the infrastructure a set of configuration and environment variables was devised, figure 4.2 shows the class diagram of the common infrastructure environment variables for all developed micro-services.

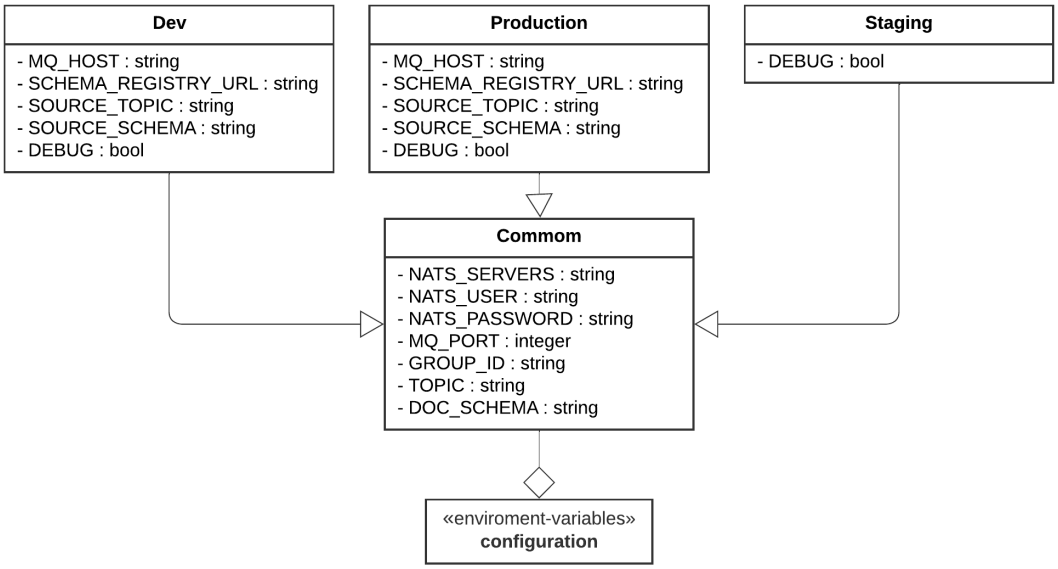


Figure 4.2: Environment Variables Class Diagram

Methodology adopted to evaluate the developed micro-services within the openZDM infrastructure

The experiments that will be devised further in the developed micro-services were done through a benchmark micro-service that was developed to evaluate the time taken to generate data in the resulting queues where the tools publish their results. It collects the data and counts the time to process different amount of cars, this evaluation allows the analysis of the time required to produce information to other micro-services and the user interfaces, with the information ready to be analyzed and displayed. The program collects the initial time in milliseconds and the time elapsed after producing a specific amount of information in the output queues, producing at the end a plot of the time behavior after producing different amounts of data. Figure 4.3 represents the program implementation to test both systems.

```python
class Benchmark:
    def __init__(self, testing_amount: list):
        self.start_time = None
        self.counter = 0
        self.testing_amount = testing_amount
        self.result = []

    def is_first(self):
        if self.start_time is None:
            return True
        return False

    def is_end(self):
        if self.counter == max(self.testing_amount):
            return True
        return False

    def save_result(self, topic):
        x, y = self.testing_amount, self.result
        freq = [round(rel[1] / rel[0], 3) for rel in zip(x, y)]
        slope, intercept = np.polyfit(x, y, 1)
        regression_line = [slope * i + intercept for i in x]
        plt.figure(figsize=(12, 8))
        plt.scatter(x, y, color='blue', label='Data Points')
        plt.plot(x, regression_line, color='red',
                 label=f'Linear Regression y={slope:.2f}x+{intercept:.2f}')
        plt.xlabel('Data Amount')
        plt.ylabel('Time in seconds')
        plt.title(f'{topic} - {freq.__repr__()}')
        plt.legend()
        plt.savefig(f'{topic}_results.png')

    def run(self, topic, logger):
        self.counter += 1
        if self._is_first():
            self.start_time = asyncio.get_running_loop().time()
        if self.counter in self.testing_amount:
            self.result.append((asyncio.get_running_loop().time() - self.start_time))
        if self._is_end():
            self.save_result(topic)
            return self.result
```

Figure 4.3: The Benchmark Program implemented to test both micro-services.

4.2 Rule-Based Monitoring Tool

The first analytical tool developed is a real-time rule-based monitoring micro-service, working as framework for developing contextualized and general purpose rules aligned with the Reference Architectural Model Industrie 4.0 (RAMI4.0) [13], to implement ZDM strategies, namely this tool is responsible in identifying early outliers, patterns and trends

by rule triggering in the collected data. This tool assumes a critical step in a day-to-day production and engineering operations since it allows the early detection of quality deviations, generating real-time alerts that enable proactive actions to be taken.

4.2.1 Architecture

Before delving into the proposed tool architecture, is important to understand the match between the framework solution and the tool objective into allowing the users to not only developed contextualize rules, but to specify the functionality to different use cases, allowing the users to easily adapt the internal parameters to new contexts and incoming data types, generating and reliable output that represents the triggering results. Figure 4.4 illustrates the architecture to achieve this goal. The encapsulated implementation simplified the configuration within the openZDM infrastructure only requiring the interfaces into subscribing to the real-time data stream inputs from the measurement stations sensors, and generating the results into a queue for other services such as the visualization and diagnosis tools.

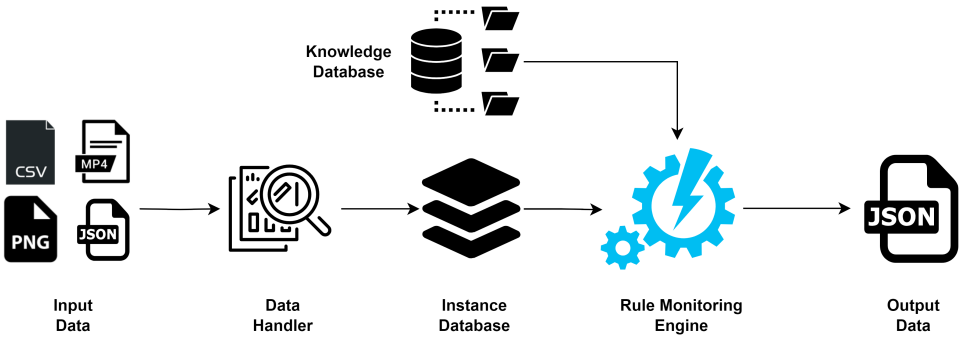


Figure 4.4: Rule-based Monitoring Tool Architecture. [8]

Each component of the architecture was designed as a flexible framework, allowing specific use case scenarios to be configured at every processing stage—from the data handler and instance database setup to the knowledge base rules and the inclusion of additional information in the output. The detailed breakdown of each component in the tool is devised below:

- **Input Data:** Data could be received from any source, in any format, and ingested into the micro-service for analysis and processing. But so far for this specific study case the information from the data source queues follows a JSON like format.

- **Data Handler:** The Data Handler is the first component to interpret, analyze, and process incoming data. This customization program, can be tailored to specific use cases, ensuring data is normalized and preprocessed, making it consistent and ready for further evaluation by the monitoring engine.

- **Knowledge Base:** This repository holds the rules, configuration files, and domain-specific knowledge needed for evaluating the data. The Rule Monitoring Engine accesses this knowledge base to retrieve the necessary rules for analyzing the current dataset.

- **Instance Database:** Acting as temporary storage, the Instance Database holds data before it is processed by the Rule Monitoring Engine. This buffer ensures that data is stored and ready for further evaluation and output generation.

- **Rule Monitoring Engine:** After normalization, the Rule Monitoring Engine retrieves data from the Instance Database and evaluates it using a set of predefined rules. These rules, stored in the Knowledge Base, include domain-specific and general logic for data analysis. The engine processes data in batches once sufficient data and all configuration criteria are met, ensuring the system runs efficiently.

- **Output Data:** After rule execution, if any rule are triggered, the system generates output data in a normalized JSON schema. (The output file will be exemplified at section 4.2.5) This data is ready for use by downstream systems, such as the front-end visualization tools and the diagnosis micro-service.

The chosen approach simplifies the customization for each data source and enables an easy integration of new data sources into the rule-based monitoring system. In this context, the system was configured for the body shop and assembly measurement stations

described in Section 3. It consumes data from four different queues, one for each measurement station and one that provides additional gap and flush measurement feedback information, at the end of the process it produces the output data in a designated queue following a reliable schema. Figure 4.5 shows how the micro-service interact with the data sources and other micro-services in the openZDM project.

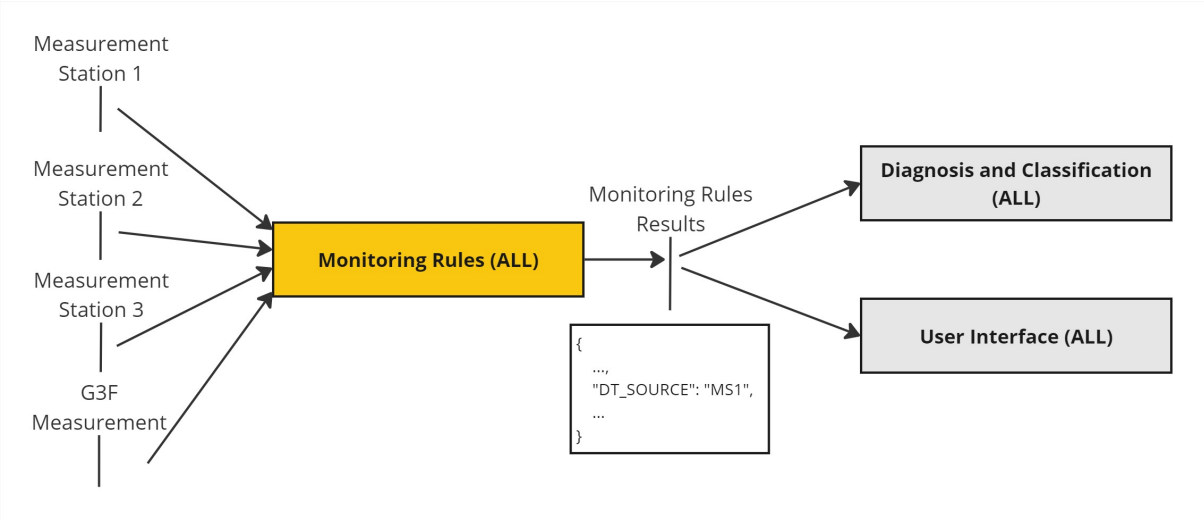


Figure 4.5: Rule-based Monitoring Tool dataflow in the openZDM project.

4.2.2 Implementation

The micro-service was implemented entirely in Python, utilizing two key libraries: NumPy [16] (Version: 1.26.4), which provides efficient numerical computations, particularly for handling arrays, matrices, and a wide range of mathematical operations specifically for coding the rule operations and Pandas [41] (Version: 2.2.2), which simplifies data manipulation and analysis, making tasks like data filtering, grouping, and cleaning more efficient during the rule testing phase.

To ensure reliable local data storage, both to prevent data loss and system failures and to accumulate information for rule processing, a SQLite database was integrated as the instance database. This was implemented using SQLAlchemy [5] (Version: 2.0.30), which provides an ORM (Object-Relational Mapping) layer, allowing for more flexible and intuitive interaction with the database. SQLAlchemy simplifies database operations

by abstracting raw SQL queries into Python objects, making it easier to manage, query, and manipulate the data within the micro-service environment. This approach offers a lightweight, fast, and file-contained solution, ensuring dependable data persistence in the isolated micro-service architecture.

The class diagram in figure 4.6, along with the following items, provides a detailed explanation of each class component in the code, describing their interactions and identifying which components are configurable.

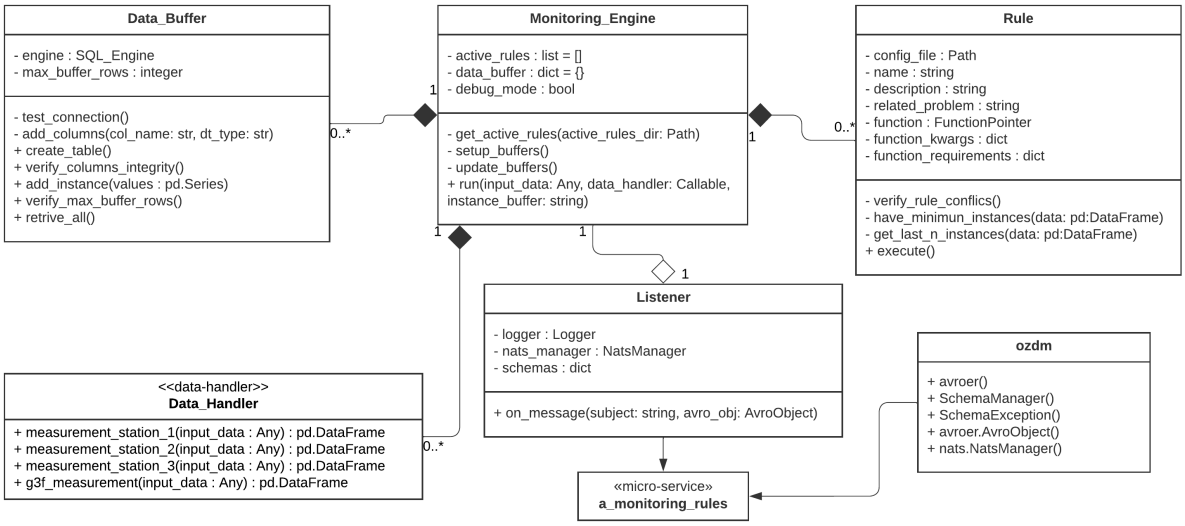


Figure 4.6: Rule-based Monitoring Tool class diagram.

- **Data_Handler:** The Data Handler is a file that contains the use case-specific functions to interpret and normalize each data source before further processing by the monitoring engine. These handler functions are passed as parameters to the monitoring engine dynamically, by the listener class.

- **Data_Buffer:** The Data Buffer class implements the instance database functionalities, acting as an interface between the monitoring engine and the SQLite databases created for each data source. A key feature of the Data Buffer is its ability to verify column integrity and automatically adjust the database structure as new measurements are collected in real time.

- **Rule:** The Rule class is responsible for implementing rule functionalities. These objects are automatically instantiated by the monitoring engine using the knowledge database, which stores the rule logic and configuration files.

- **Monitoring_Engine:** The Monitoring Engine class brings together the functionalities of the other classes. It automatically creates rule objects and data buffers. Being instantiated in the a_monitoring_rules interface file.

- **Listener:** The Listener class, is responsible for receiving the new messages from the subscribed queues and apply the monitoring engine object functionalities to the incoming data.

- **ozdm:** The ozdm library implements various functionalities and classes for managing schemas and integrating the micro-services within the openZDM infrastructure.

- **a_monitoring_rules:** This python program serves as the interface between the monitoring rules engine and the ozdm infrastructure.

To enhance reliability, and simplify the integration and deployment the tool, uses a Docker container. This approach isolates each process at the system level, meaning that the container will include all required system files and dependencies, completely separated from other micro-services and only connected to the host file system by a volume for data persistence in the instance buffer. This containerization significantly enhances system stability and predictability, allowing for consistent behavior across various environments. It also streamlines the continuous integration/continuous deployment (CI/CD) pipeline, enabling faster iterations and improvements to the micro-services without affecting the overall infrastructure. This container-based architecture ensures the reliability and scalability necessary for industrial applications, such as those running on factory-floor production computers, where downtime or system failure can have critical consequences.

Configuring the Data Handler

The data handler is not bound to a specific coding protocol, as it must be adaptable to different use cases and input data formats. However, the final step of normalization must adhere to a standard: data should be returned as a Pandas DataFrame, containing the relevant features as the columns to be processed and stored in the instance database. This ensures that the data is in an normalized format to be process. Figure 4.7 below exemplify the two data handler developed specifically for the body shop and assembly measurement stations use case respectively.

```python
GAP_FLUSH_SCHEMA = Schema(
    {
        Optional("_id"): Or(object, str),
        "TIMESTAMP": Or(object, str),
        "ID": int,
        "STATION": str,
        "MEASUREMENT_DATA": [
            {
                "REFERENCE": str,
                "GAP": Or(float, int, None),
                "FLUSH": Or(float, int, None),
            }
        ]
    }
)

def gap_flush_handler(input_data):
    # Validates Input Data
    if not GAP_FLUSH_SCHEMA.is_valid(input_data):
        warnings.warn("Warning current Gap and Flush data "
                      "is not from a valid schema")
        return None

    # Normalize input data in a pd.DataFrame object
    columns = []
    data = []
    for measurement in input_data["MEASUREMENT_DATA"]:
        columns.append(measurement["REFERENCE"] + "_Gap")
        data.append(float(measurement["Gap"])
                    if measurement["Gap"] is not None else None)
        columns.append(measurement["REFERENCE"] + "_Flush")
        data.append(float(measurement["Flush"])
                    if measurement["Flush"] is not None else None)
    return pd.DataFrame([data], columns=columns, index=[input_data["JSN"]])
```

(a) Gap and Flush Stations Data Handler

```python
X_Y_Z_SCHEMA = Schema(
    {
        Optional("_id"): Or(object, str),
        "TIMESTAMP": Or(object, str),
        "ID": int,
        "STATION": str,
        "MEASUREMENT_DATA": [
            {
                "REFERENCE": str,
                "X": Or(float, int, None),
                "Y": Or(float, int, None),
                "Z": Or(float, int, None)
            }
        ]
    }
)

def x_y_z_handler(input_data):
    # Validates Input Data
    if not X_Y_Z_SCHEMA.is_valid(input_data):
        warnings.warn("Warning current x y z data is not from a valid schema")
        return None

    # Normalize input data in a pd.DataFrame object
    columns = []
    data = []
    for measurement in input_data["MEASUREMENT_DATA"]:
        columns.append(measurement["REFERENCE"] + "_X")
        data.append(float(measurement["X"])
                    if measurement["X"] is not None else None)
        columns.append(measurement["REFERENCE"] + "_Y")
        data.append(float(measurement["Y"])
                    if measurement["Y"] is not None else None)
        columns.append(measurement["REFERENCE"] + "_Z")
        data.append(float(measurement["Z"])
                    if measurement["Z"] is not None else None)
    return pd.DataFrame([data], columns=columns, index=[input_data["JSN"]])
```

(b) X, Y, Z Station Data Handler

Figure 4.7: Data Handlers developed for the body shop and assembly measurement stations use case.

Developing New Rules

The rule definition protocol for the Rule-Based Monitoring Tool involves a two-step process. First, the business logic should be implemented as a python function that must be designed to accept two arguments: one for the incoming data and another for accessing

parameters defined in the configuration file. The second step is creating a JSON configuration file that specifies how data should flow to ensure that the rule works properly. This file also enables dynamic adjustments to internal variables without requiring changes to the rule's underlying code. Each of the arguments from the configuration file are defined below:

- **Name:** The rule name or identifier.
- **Description:** The rule description should contain the business logic and objective of the rule, explaining in at user level what the rule is tracking.
- **Related Problem:** That argument works as the triggering description and explanation of the consequence of the rule triggering to the process.
- **Function:** This argument is related to the rule python file and function names, and is used by the system to properly link the descriptions from the configuration file to the rule code.
- **FunctionKwargs:** This dictionary object parameter is responsible to pass additional argument to the rule's function code without the need to make any code changes.
- **FunctionRequirements:** This dictionary object always have three argument, and is manly used by the rule engine to understand in which format the rule expect data to sent, and what are the minimum requirements to work.
  - **DataFormat:** This argument expect one of three values: COL, ROW, ALL, they respectively how the table from the instance buffer will be spliced before the data will be sent.
  - **MinimunInstances:** This argument sets the minimum amount of instances in the Instance Database before the rule starts to be tested.
  - **LastNInstances:** Finally this argument sets how much from the last incoming instances the rule expect to receive.

To exemplify the definition of a new rule, figures 4.8 and 4.9 illustrate a non-context related ML rule that was defined using Linear Regression as a rule example in addition to a configuration file where the parameters can be adjusted. This rule triggers when the linear regression (Prediction) reaches the lower or upper thresholds within fewer than X predicted instances, indicating an extremely steep or shallow trend slope and tracking unexpected deviation trends.

```python
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

def linearregression(data: pd.Series, args) -> dict:
    # Uses linear regression to verify when a variable have a high tendency
    upper_limit = data.mean() + data.std() * 3
    lower_limit = data.mean() - data.std() * 3
    y = data.to_numpy(dtype=np.float64)
    X = np.arange(0, len(data), 1, dtype=int).reshape(-1, 1)
    reg = LinearRegression().fit(X, y)
    intercept = reg.intercept_
    slope = reg.coef_[0]
    if slope in (0, 1): return {
          "result": False
        }
    X_upper_limit = (upper_limit - intercept) / slope
    X_lower_limit = (lower_limit - intercept) / slope
    val, limit = get_positive_value(X_lower_limit, X_upper_limit)
    if val < args["DistanceLimitFromThreshold"]:
        return {
            "result": True,
            "additional_result_information":
                f"DistanceToTheThreshold:{str(val)}({limit})/Intercept:{str(intercept)}/Slope:{str(slope)}",
            "analyzed_JSN": data.index.to_list()
        }
    return {
        "result": False,
        "additional_result_information":
            f"DistanceToTheThreshold:{str(val)}({limit})/Intercept:{str(intercept)}/Slope:{str(slope)}"
    }

def get_positive_value(value1, value2):
    if value1 > 0: return round(value1), "Lower"
    return round(value2), "Upper"
```

Figure 4.8: The code file example for a linear regression to generate alerts for unexpected deviation trends.

```json
{
  "Name": "LinearRegression",
  "Description": "Uses linear regression to verify if the deviation has a strange prediction tendency.",
  "RelatedProblem": "Based on previous data an alerting trend exists.",
  "Function": "linearregression",
  "FunctionKwargs": {
    "DistanceLimitFromThreshold": 10
  },
  "FunctionRequirements": {
    "DataFormat": "COL",
    "MinimunInstances": 20,
    "LastNInstances": 45
  }
}
```

Figure 4.9: The configuration file example for a linear regression to generate alerts for unexpected deviation trends.

The dynamic approach in developing rules and setting additional arguments to simplify on the fly adjustments, has proven to be more efficient, particularly when working with Machine Learning models and other kinds of rules that often require specific data formats and preprocessing steps that can be easily configured by the configuration file.

4.2.3 Experiments

To evaluate the micro-service efficiency in processing and producing results two different approaches were developed:

- The first one aimed to evaluate the micro-service in an isolated and controlled environment to test the worst case scenarios the system can handle such as the number of rules, or columns being monitored without the interference of the infrastructure.
- The second experiment aimed to test the micro-service performance in relation to execution time and reliability with the hole infrastructure setup.

Both experiment were devised in a development local machine provided by the IPB cluster that replicates the production resources with the following specifications: (CPU) AMD EPYC 7351 16-Core Processor (Working with 4 cores at 2.4 Ghz 45W), and (RAM) 64GB DDR4 2400 Mhz, both tests ensures that the rule-based monitoring tool will behave as expected before deployment in the industry shop-floor machines.

Both experiments were evaluated by the worst-case scenario, being conducted using a test dataset of instances from Measurement Station 1, which records 720 measurements per car [8], with all documents being dropped at once to the file watcher (Reference figure 4.1). This station represents the highest number of monitored points, making it the most complex for evaluation.

To effectively test the monitoring tool, nine rules were implemented within the framework: The linear regression rule, already devised in section 4.2.2 and the eight Nelson Rules [30] (Devised in section 2.4) were added to the knowledge base as a non-context specific rule package. The Nelson Rules are a set of rules recognized for their statistical

accuracy, being used to detect out-of-control conditions in the process, helping in identifying deviations early on and addressing them before they escalate into major quality concerns.

4.2.4 Experiment Results

First Experiment

The first experiment was applied without any interference's from the infrastructure. Time measurements were recorded after producing 10, 20, 40, 70, 110, and 160 outputs, respectively and the results are devised in table 4.1 below.

Amount of Cars Processed	10	20	40	70	110	160
Time Elapsed (in Seconds)	17.06	35.96	66.4	111.44	173.47	253.92
Per Instance Operation (Time Elapsed / Cars Processed)	1.706	1.798	1.66	1.592	1.577	1.587

Table 4.1: Experiment 1: Amount of Cars Processed vs Time Elapsed and Per Instance Operation, isolated.

Second Experiment

The second experiment was conducted using the benchmark micro-service described in Section 4.1 and evaluates the monitoring_rules_results queue where the output is published by the micro-service. Similarly to experiment 1 time were recorded after producing 10, 20, 40, 70, 110, and 160 outputs, respectively and the results are devised in table 4.2 below.

Amount of Cars Processed	10	20	40	70	110	160
Time Elapsed (in Seconds)	0.09	67.9	102.16	137.76	244.42	401.44
Per Instance Operation (Time Elapsed / Cars Processed)	0.009	3.395	2.554	1.968	2.222	2.509

Table 4.2: Experiment 2: Amount of Cars Processed vs Time Elapsed and Per Instance Operation within the openZDM infrastructure.

4.2.5 Output Generated

After the rule execution, if any rule are triggered, the system generates output data in a normalized JSON format. This data is ready for use by downstream systems, such as the front-end visualization tools and the diagnosis micro-service. Figure 4.10 exemplify the output generated by the rule-based monitoring micro-service for all data sources.


```json
{
   "TIMESTAMP": "03_08_2024_19.20.20",
   "DT_SOURCE": "measurement_station_1",
   "RULES_RESULTS": [
      {
         "NAME": "Nelson_Rule_1",
         "DESC": "One point is more than 3 standard deviations from the mean.",
         "RELATED_PROBLEM": "One sample is grossly out of control.",
         "RESULTS": {
            "Pilar_A_LH_Gap": {
               "result": true,
               "additional_result_information": "Point triggered below lower thereshold.",
               "current_car": 7000005,
               "analyzed_cars": [
                  7000005, 7000004, 7000003, 7000002, 7000001
               ]
            },
            "Pilar_B_LH_Gap": {
               "result": true,
               "additional_result_information": "Point triggered above upper thereshold.",
               "current_car": 7000000,
               "analyzed_cars": [
                  7000005, 7000004, 7000003, 7000002, 7000001
               ]
            }
         }
      }
   ]
}
```


Figure 4.10: Output JSON generated by the rule-based monitoring micro-service.

The JSON data contains the data source queue, a list with the rules triggered with their respectively information's, and in which measurements the rule triggering was detected, followed with additional information's around the detections, the current car, and in which cars the analysis were done.

4.3 Diagnosis Tool

The second analytical tool developed is the diagnosis tool, designed to diagnose defects in automotive body shop and assembly lines. Also aligned with the RAMI4.0 [13], into

supporting the Zero Defect Manufacturing strategies. This tool compromises two micro-services: one for labeling and evaluation, and another for classification and diagnosis. These micro-services assess car and measurement system quality, offering insights into defect causes, consequences, and possible solutions. The system is vital in daily production and engineering, as it not only identifies root causes of defects but also recommends corrective actions.

The decision to split the diagnosis tool into two micro-services allows for broader use. The accepted methodology for evaluating car and measurement quality can be used by other micro-services and also enables predictive analysis.

4.3.1 Architecture

Figure 4.11 shows how both micro-services from the diagnosis tool interact with the data sources and other micro-services queues in the openZDM infrastructure.

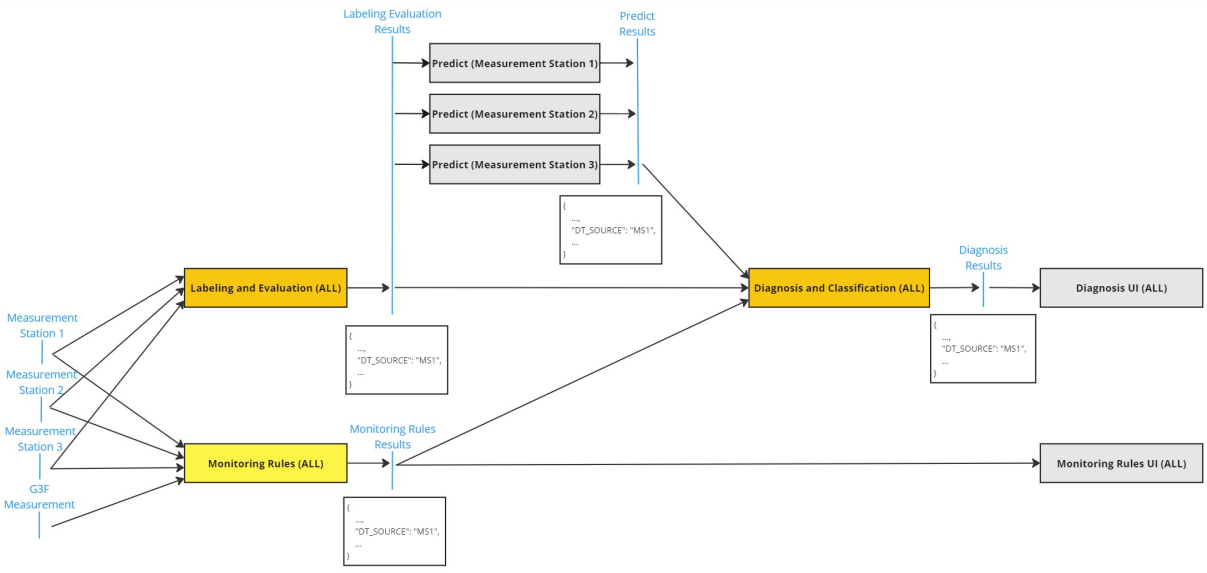


Figure 4.11: Diagnosis Tool dataflow in the openZDM project.

Labeling and Evaluation

The first micro-service from the diagnosis tool is the Labeling and Evaluation system this service processes the information and calculates the car and measurement quality.

Figure 4.12 illustrates the micro-service architecture. This tool subscribes the data sources from the measurement stations processes the incoming data and generates the car and measurement system quality indicators, which are then consumed by the prediction and diagnosis micro-service.

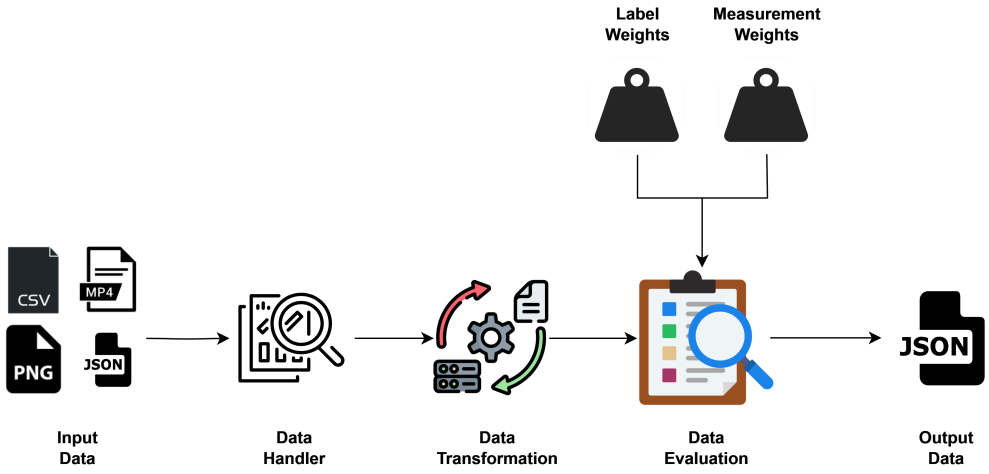


Figure 4.12: Labeling and Evaluation Architecture.

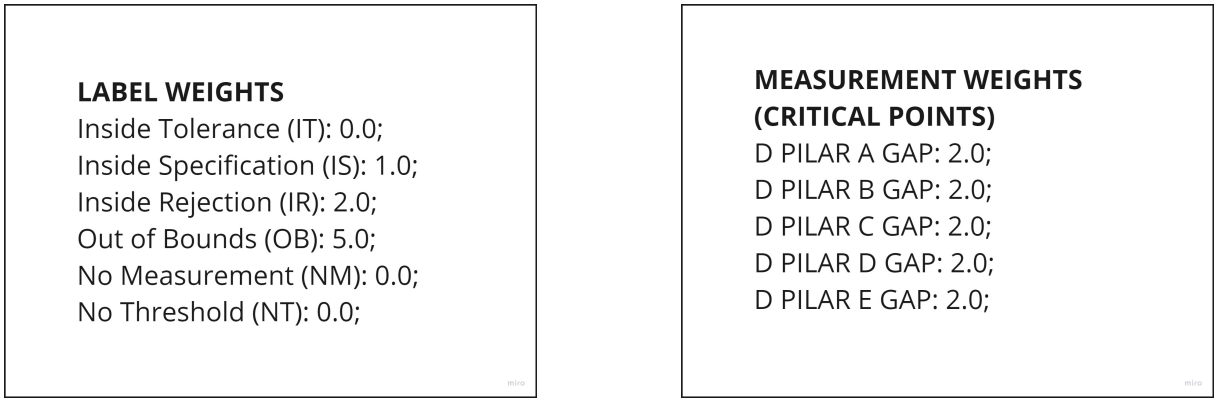
Each component of the architecture was specifically designed for the body shop and assembly line use cases. Below is a breakdown of each component:

- **Input Data:** Input data must follow the measurement and threshold format. Since only measurement stations 1, 2, and 3 meet this requirement, the service subscribes exclusively to these queues.
- **Data Handler:** The Data Handler interprets, analyzes, and processes incoming data, ensuring it includes the necessary measurement and threshold values.
- **Data Transformation:** This step applies the transformation methodology from Section 3.4.3, converting deviations and thresholds into meaningful categorical variables for further analysis.
- **Label Weights:** Label weights represent defect indicators for each class of measurement in the car, based on the threshold limits shown in Figure 3.1.

- **Measurement Weights:** Measurement weights quantify the importance of defects in critical car measurements, as defined by industry leaders.
- **Data Evaluation:** This object calculates the car and measurement quality by summing and weighting the label and measurement values.
- **Output Data:** After evaluation, the system outputs data in a normalized JSON format, ready for use by downstream systems like the prediction and diagnosis micro-service.

Car and Measurement Quality Calculation

Before understanding the calculation of the quality indicators is important to properly understand the label and measurement weights, and how they affect the defect indicator, figure 4.13 represent respectively the weights developed for the body shop and assembly lines use case. Where in both cases semantically lower values means less defect in the car.



(a) Label Weights (b) Measurement Weights

Figure 4.13: Body shop and Assembly Defect Weights

The car and measurement quality indicators are calculated based on a sequence of equations, explanations and conclusions about the Labels and Measurements defects. Equation 4.1 represents the calculation of the defect indicator from an entire car where the defect weights are summarized, equation 4.2 otherwise represents the worst case scenario

for the current car, taking into consideration the missing measurement and threshold values in the car. Equation 4.3 calculate the worst possible scenario were hypothetically all measurement are collected, but all measurement have the worst label. This indicator enables the possibility to evaluate not only the car quality, but the measurement system quality, correctly pounding critical points for the body shop and assembly process.

$$D^i = \sum_{i=0}^{\text{len}(m)} L_w * M_w \tag{4.1}$$

Where:

- $\sum_{i=0}^{\text{len}(m)}$: The defect summation from each deviation labeled in the car.
- L^w: Label weight of the current label.
- M^w: Measurement specific weight for weighting critical points.

$$C.W.D^i = \sum_{i=0}^{len(m)} max(L_w) * M_w \forall Label \neq \text{NT or NM} \tag{4.2}$$

Where:

- $\sum_{i=0}^{\text{len}(m)}$: The defect summation from each deviation labeled in the car.
- $max(L^w)$: The worst label weight from the weights in 4.13 (Current OB having a label weight of 5).
- M^w: Measurement weight for weighting critical points.
- $\forall Label \neq$ NT or NM: For all deviations not labeled as No Measurement or No Threshold.

$$G.W.D^i = \sum_{i=0}^{len(m)} max(L_w) * M_w \tag{4.3}$$

Where:

- $\sum_{i=0}^{\text{len}(m)}$: The defect summation from each deviation labeled in the car.
- $max(L^w)$: The worst label weight from the weights in 4.13 (Current OB having a label weight of 5).
- M^w: Measurement weight for weighting critical points.

Finally equations 4.4 and 4.5 calculate car quality as the inverse proportion between the defect indicator and the worst current defect possible (Note: The inverse proportion is crucial, as lower values indicate fewer defects in the car.), and measurement quality as the proportion between the worst current defect and the global worst defect. Since these values are proportions, they range between 0 and 1, with 1 indicating perfect quality and 0 representing the worst possible quality (Figure A.1 exemplify the car and measurement quality calculation diagram.).

$$CarQuality = 1 - \frac{D^i}{C.W.D^i} \tag{4.4}$$

Where:

- D^i: The car defect indicator.
- $C.W.D^i$: The current worst defect indicator possible, if all collect deviations (discarding the NM and NT values) were labeled as OB.

$$MeasurementQuality = \frac{C.W.D^i}{G.W.D^i} \tag{4.5}$$

Where:

- $C.W.D^i$: The current worst defect indicator possible, if all collect deviations (discarding the NM and NT values) were labeled as OB.
- $G.W.D^i$: The global worst defect indicator possible, if all deviations were collected without any NM and NT values and all deviations labeled as OB.

This equations introduce the data evaluation methodology, presented at figure 4.12 into quantifying the car and measurement qualities for further diagnosis and predictions for the respective micro-services (As figure 4.11 introduces).

Diagnosis and Classification

The second micro-service in the diagnosis tool is the Classification and Diagnosis system, responsible for classifying and diagnosing the car based on rule-triggering results, car and measurement quality indicators, and predicted quality metrics. It processes incoming data and generates diagnoses, outlining the cause, consequence, and possible solutions for deviations and defects, which are then passed to the user interface. Figure 4.14 represents the architecture from the diagnosis micro-service.

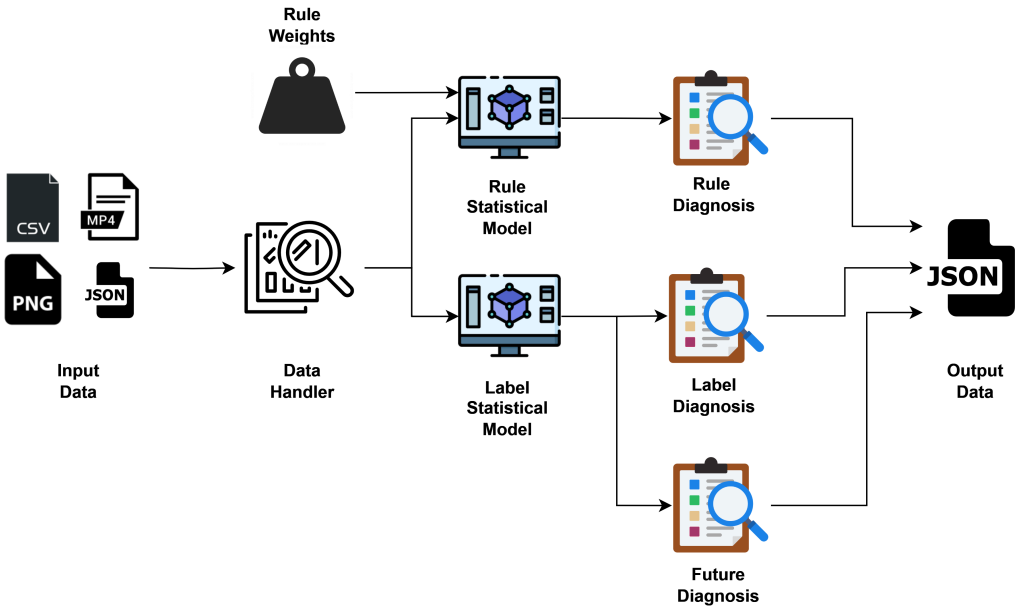


Figure 4.14: Diagnosis and Classification Architecture.

Similarly to the labeling and evaluation micro-service each component of the architecture was specifically designed for the body shop and assembly line use cases. Below is a breakdown of each component:

- **Input Data:** This service consumes data from 3 different micro-services: rule-triggering results, car and measurement quality indicators, and the predicted quality metrics.

- **Data Handler:** The Data Handler interprets, analyzes, and processes incoming data, ensuring it includes the necessary information and schema for each diagnosis type.

- **Rule Weights:** Rule weights quantify the importance of the rule triggering results in critical car measurements, as defined by industry leaders.

- **Statistical Models:** The statistical models classifies the current car based on the progression of the last N cars, measurement and rule triggering quality values, defining threshold to classify the cars into APPROVED, ATTENTION and REJECTED.

- **Rule Diagnosis:** The rule diagnosis system classifies the rule triggering results based in the statistical model history, the rule descriptions and related problems.

- **Label Diagnosis:** The label diagnosis system classifies the car and measurement quality results based in the statistical model history, adding more information about the specific measurement that are the causes of the defects, the process consequence of this defects, and the solution.

- **Future Diagnosis:** The future diagnosis system classifies the car and measurement quality from the predicted next N cars, based on the current statistical model, and they giving insights about trends in the car and measurement quality indicating miscalibration and problems in the measurements.

- **Output Data:** After finishing the diagnosis, the system outputs data in a normalized JSON format, ready for use by the front-end visualization tools.

Statistical Models

The statistical models are designed to classify the current car using a traffic light system. There are two types of models: one for car and measurement qualities, and another for rule defect indicators. Both follows the same statistical calculation devised in equation 4.6 and classify results into three categories: APPROVED, ATTENTION, and REJECTED.

$$Class = \begin{cases} APPROVED & if \quad quality \geq \bar{x} - \sigma \\ ATTENTION & if \quad \bar{x} - \sigma > quality > \bar{x} - 3 * \sigma \\ REJECTED & if \quad quality \leq \bar{x} - 3 * \sigma \end{cases} \tag{4.6}$$

Where:

- *quality*: The quality from the current car or measurement.
- $\bar{x}$: The quality mean in the last N cars.
- σ: The quality standard deviation in the last N cars.

The simplified traffic light classification was chosen to help end-users on the shop floor quickly identify defects, signaling the need for further investigation. This approach makes it easier to verify problem occurrences and prompts deeper analysis of root causes and consequences, as identified by the diagnostic tool.

4.3.2 Implementation

Similarly to the rule-based monitoring tool, both micro-service from the diagnosis tool were implemented entirely in Python, utilizing two key libraries: NumPy [16] (Version: 1.26.4), specifically for coding the evaluation calculations and Pandas [41] (Version: 2.2.2), for data manipulation, making tasks like data transformation and grouping more efficient during the evaluation and statistical model classification steps. An SQLite database was also added in the classification and diagnosis micro-service using the SQLAlchemy [5] (Version: 2.0.30), allowing for a more flexible and intuitive interaction with the database

in order to save the historical quality measurement for the statistical model to evaluate and classify the cars.

Figure 4.15 and 4.16, represents along with the following items, the class diagram and a detailed explanation of each class component in the code, describing their interactions and identifying which components are configurable for each micro-service.

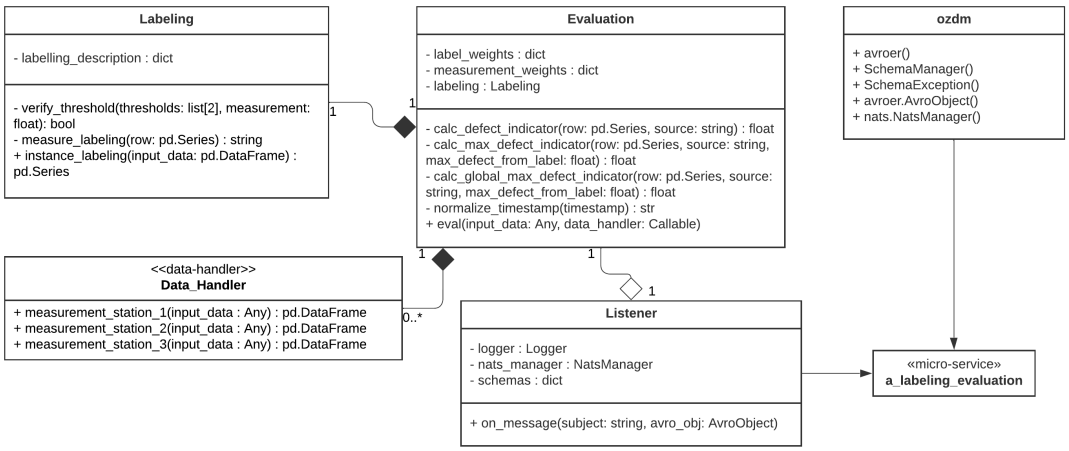


Figure 4.15: Labeling and Evaluation class diagram.

- **Data_Handler:** The Data Handler is a file that contains the use case-specific functions to interpret and normalize each data source ensuring that the data contains the deviations and thresholds before processing. These handler functions are passed as parameters to the evaluation object dynamically, by the listener class.

- **Labeling:** The Labeling class contains all the data transformation functionalities devised in section 3.4.3.

- **Evaluation:** The Evaluation class is responsible for implementing the defect indicators functionalities applying the label and critical measurement weights and calculating the car and measurement system qualities.

- **Listener:** The Listener class, is responsible for receiving the new messages from the subscribed queues and apply the labeling and evaluation objects functionalities to the incoming data.

- **ozdm:** The ozdm library implements various functionalities and classes for managing schemas and integrating the micro-services within the openZDM infrastructure.
- **a_labeling_evaluation:** This python program serves as the interface between the labeling and evaluation micro-service and the ozdm infrastructure.

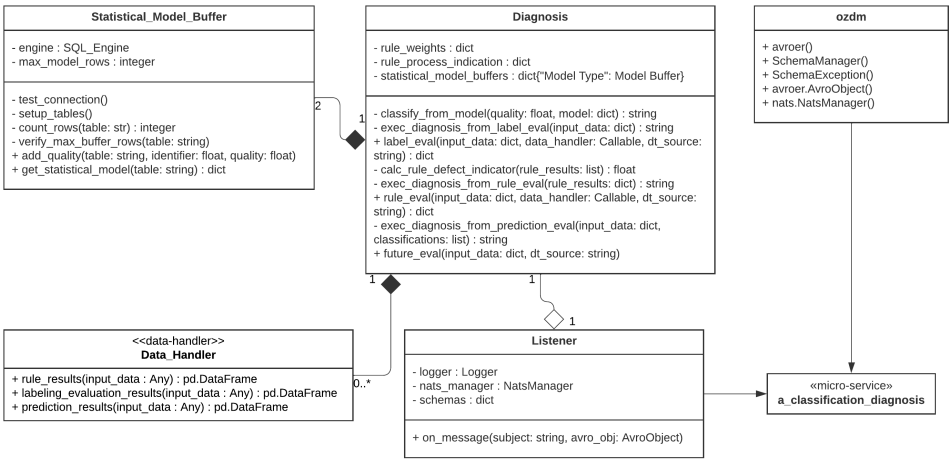


Figure 4.16: Classification and Diagnosis class diagram.

- **Data_Handler:** The Data Handler is a file that contains the use case-specific functions to interpret and normalize each data source ensuring data format and schema from the rule-monitoring, labeling and evaluation, and predictions queues are in correct format before further diagnosis. These handler functions are passed as parameters to the diagnosis class dynamically, by the listener class.
- **Statistical_Model_Buffer:** The Statistical Model Buffer class implements the database functionalities. Saving the quality results and retrieving the current updated statistical model.
- **Diagnosis:** The Diagnosis class is responsible for applying the statistical model dynamically to the incoming data and also to analyze the cause, consequence and solution of the detected defects for each data source. This object is instantiated in the a_classification_diagnosis interface file.

- **Listener:** The Listener class, is responsible for receiving the new messages from the subscribed queues and apply the monitoring engine object functionalities to the incoming data.
- **ozdm:** The ozdm library implements various functionalities and classes for managing schemas and integrating the micro-services within the openZDM infrastructure.
- **a_classification_diagnosis:** This python program serves as the interface between the classification and diagnosis micro-service and the ozdm infrastructure.

4.3.3 Experiments

Similarly to the monitoring rules micro-service, both micro-services from the diagnosis tool were tested and evaluated in terms of efficiency and reliability in processing and producing the expected outputs with the hole infrastructure setup. But not isolated, this in turn was defined because the micro-services from the diagnosis tool are more related and dependable of the body shop and assembly lines use case making the isolated analysis have no intrinsic purpose.

The micro-services were tested in a development local machine provided by the IPB cluster that replicates the production resources with the following specifications: (CPU) AMD EPYC 7351 16-Core Processor (Working with 4 cores at 2.4 Ghz 45W), and (RAM) 64GB DDR4 2400 Mhz, both tests ensures that the diagnosis tool will behave as expected before deployment in the industry shop-floor machines.

The experiments were evaluated by the worst-case scenario, being conducted using a test dataset of instances from Measurement Station 1, which records 720 measurements per car [8], with all documents being dropped at once to the file watcher (Reference figure 4.1). This station represents the highest number of monitored points, making it the most complex for evaluation.

Specifically for the classification and diagnosis micro-service, the data input depends on the output frequency of other micro-services, such as the rule and prediction micro-services, which do not have a 1-to-1 relationship between the data source and output

generation. In this context to ensure a worst-case scenario, the testing data was slightly modified to guarantee that at least one rule is triggered for each measurement processed by the monitoring rules micro-service, thereby creating a 1-to-1 relationship. The prediction micro-service (Note: This service was not implemented by the authors of this document and has not yet been published, but its contributions are referenced in figure 4.11, and the diagnosis systems uses it information for one of its diagnosis.) was configured to generate outputs every 20 cars, as it involves intensive mathematical operations due to being one Machine Learning model per quality results, and the training step being intensively activated.

4.3.4 Experiment Results

Labeling and Evaluation

This experiment was conducted using the benchmark micro-service described in Section 4.1 and evaluates the labeling_evaluation_results queue where the output is published by the micro-service. Time were recorded after producing 10, 20, 40, 70, 110, and 160 outputs, respectively and the results are devised in table 4.3 below.

Amount of Cars Processed	10	20	40	70	110	160
Time Elapsed (in Seconds)	10,23	15,98	30,6	62,86	103,29	145,44
Per Instance Operation (Time Elapsed / Cars Processed)	1,023	0,799	0,765	0,898	0,939	0,909

Table 4.3: Experiment: Amount of Cars Processed vs Time Elapsed and Per Instance Operation from the Labeling and Evaluation Micro-service with the hole infrastructure.

Classification and Diagnosis

The second experiment was conducted using the benchmark micro-service described in Section 4.1 and evaluates the diagnosis_results queue where the output is published by the micro-service. Similarly to experiment 1 time were recorded after producing 10, 20, 40, 70, 110, and 160 outputs, respectively and the results are devised in table 4.4 below.

Amount of Cars Processed	10	20	40	70	110	160
Time Elapsed (in Seconds)	12,24	21,08	41,0	80,71	119,35	172
Per Instance Operation (Time Elapsed / Cars Processed)	1,224	1,054	1,025	1,153	1,085	1,075

Table 4.4: Experiment: Amount of Cars Processed vs Time Elapsed and Per Instance Operation from the Classification and Diagnosis Micro-service with the hole infrastructure.

4.3.5 Output Generated

Labeling and Evaluation

After evaluation, the system outputs data in a normalized JSON format, ready for use by downstream systems like the prediction and classification-diagnosis micro-services. Figure 4.17 exemplify the output generated by the labeling and evaluation micro-service for all measurement stations.


```json
{
  "TIMESTAMP": "2024-04-04T04:05:30.000Z",
  "ID": 1000000,
  "DT_SOURCE": "measurement_station_2",
  "CAR_QUALITY": 0.6799999999999999,
  "MEASUREMENT_QUALITY": 0.8333333333333334,
  "MISSING_VALUES": [
    "PILAR_D_TEST_Flush",
    "PILAR_F_TEST_Flush"
  ],
  "OUT_OF_BOUNDS_MEASUREMENTS": [
    "PILAR_C_TEST_Flush"
  ]
}
```


Figure 4.17: Output JSON generated by the labeling and evaluation micro-service.

Classification and Diagnosis

After finishing the diagnosis, the system outputs data in a normalized JSON format, ready to be used by the front-end visualization tool. Figure 4.18 exemplify the output generated by the diagnosis tool for each subscribed queue, exemplifying the diagnosis for the current

labeled car, the car and measurement predicted qualities, and also the diagnosis based on the rule triggering results.


```
{
    "TIMESTAMP": "03_08_2024_16.08.00",
    "DT_SOURCE": "measurement_station_2",
    "ID": 1000000,
    "DIAGNOSIS_TYPE": "LABEL",
    "DIAGNOSIS_OVERALL_RESULT": "APPROVED",
    "DIAGNOSIS": {
        "PROBLEM_CAUSES": "Because of measurements ['D PILAR D', 'D PILAR E'] being Out of Bounds.
            And Measurements ['D PILAR A', 'D PILAR B', 'D PILAR C'] being missing.",
        "CONSEQUENCE": "The car 1000000 quality is 66.7%, and the measurement process quality is 50.0%.",
        "PROCESS_INDICATION": "Based on car 1000000 information, please adjust the measurement ['D PILAR D', 'D PILAR E'],
            because it is highly deviating from the thresholds. Also measurement process should be evaluated due to
            measurements ['D PILAR A', 'D PILAR B', 'D PILAR C'] being missing.",
        "RESULT": "Based on the last 40 cars, the statistical model classified car 1000000 as APPROVED
            (Model mean: 0.667, Model standard deviation: 0.0)."
    }
}
```


(a) Output JSON generated by diagnosis of the current labeled car diagnosis.


```
{
   "TIMESTAMP": "03_08_2024_19.20.20",
   "DT_SOURCE": "measurement_station_2",
   "ID": 7000000,
   "DIAGNOSIS_TYPE": "RULE",
   "DIAGNOSIS_OVERALL_RESULT": "APPROVED",
   "DIAGNOSIS": {
      "PROBLEM_CAUSES": "Rule Nelson_Rule_1 triggered in ['Pilar_A_Cx_Correio_LH_Gap', 'Pilar_A_Delta_Gap_LHS_Gap']
          meaning that (One sample is grossly out of control.), and with the following caveats: Pilar_A_Cx_Correio_LH_Gap
          has Point triggered below lower threshold. Pilar_A_Delta_Gap_LHS_Gap has Point triggered above upper threshold.
          Rule Nelson_Rule_2 triggered in ['Pilar_A_Cx_Correio_LH_Gap', 'Pilar_A_Delta_Gap_LHS_Gap'] meaning that (Some prolonged bias exists.),
          and with the following caveats: Pilar_A_Cx_Correio_LH_Gap has Tendency above mean. Pilar_A_Delta_Gap_LHS_Gap has Tendency below mean.
          Rule Nelson_Rule_7 triggered in ['Pilar_A_Cx_Correio_LH_Gap', 'Pilar_A_Delta_Gap_LHS_Gap'] meaning that (With 1 standard deviation,
          greater variation would be expected.), and with the following caveats: None for Now.",
      "CONSEQUENCE": "The car 7000000 has an accumulated defect indicator of 18.0 from the rule triggering (0.0 means that no rule triggered).",
      "PROCESS_INDICATION": "Measurements ['Pilar_A_Cx_Correio_LH_Gap', 'Pilar_A_Delta_Gap_LHS_Gap'] are grossly out of control try to
          re-calibrate the measurement system. And Measurements ['Pilar_A_Cx_Correio_LH_Gap', 'Pilar_A_Delta_Gap_LHS_Gap'] need a re-adjustment,
          re-calibrate the measurement system.",
      "RESULT": "Based on the last 32 cars, the statistical model classified car 7000000 as APPROVED
          (Model mean: 18.0, Model standard deviation: 0.0)."
   }
}
```


(b) Output JSON generated by diagnosis of the rule triggering results diagnosis.


```
{
   "TIMESTAMP": "2024-04-04T04:05:30",
   "DT_SOURCE": "measurement_station_2",
   "ID": 7000000,
   "DIAGNOSIS_TYPE": "PREDICT",
   "DIAGNOSIS_OVERALL_RESULT": "APPROVED",
   "DIAGNOSIS": {
      "PROBLEM_CAUSES": "The prediction has no problem cause.",
      "CONSEQUENCE": "Based on the last 5 cars the prediction model states that: The car quality will raise 10.0%
          and the measurement quality will drop 10.0% in 2 cars.",
      "PROCESS_INDICATION": "The 1st car quality predicted was REJECTED by the current statistical model, please
          keep attention on car 7000001.",
      "RESULT": "From the prediction the statistical model classified 1 as APPROVED, 0 as ATTENTION, 1 as REJECTED.
          (Model mean: 0.667, Model standard deviation: 0.0)."
   }
}
```


(c) Output JSON generated by diagnosis of the car and measurement quality predictions.

Figure 4.18: Output JSON generated by the classification and diagnosis micro-service.

While the current car diagnosis focused in producing information around the specific causes that were represented by the missing measurements and those measurements out of the the industry specifications, presenting the quality classified by the statistical model, the prediction diagnosis focus in presenting information around the evolution of the quality predicted, presenting a prognostic evaluation for end users to take proactive actions. Finally the diagnosis based on the rule triggering results merge the specific rule descriptions and problem related from the rules that were triggered, with the more accurate information from the additional parameters, and related the rules with a table of process indications for solving the problems found.

Chapter 5

Results Evaluation and Discussions

In this chapter, the evaluation and analysis of the results will be presented, with a focus on highlighting the most significant findings. The key aspects of the data will be examined in detail, emphasizing the performance of the implemented methodologies and the impact of specific parameters on the outcomes. Additionally, this section will discuss the broader implications of the results, assessing their alignment with the original objectives of the study and their potential contributions to the field. Any observed limitations or challenges encountered during the evaluation process will also be addressed, providing insights for future work and possible improvements.

5.1 Rule-Based Monitoring Performance Analysis

To evaluate the results presented in Tables 4.1 and 4.2, Figures 5.1 and 5.2 show the relationship between the time elapsed during testing and the data processed and generated. A linear regression was applied to predict the behavior of this relationship, confirming its linear nature in both isolated and within the openZDM infrastructure environments. Additionally, by analyzing the mean time elapsed per car processed at each collection point, the time to generate one output was 1,6533 seconds in the isolated environment and 2,1095 seconds when the tool was used within the infrastructure.

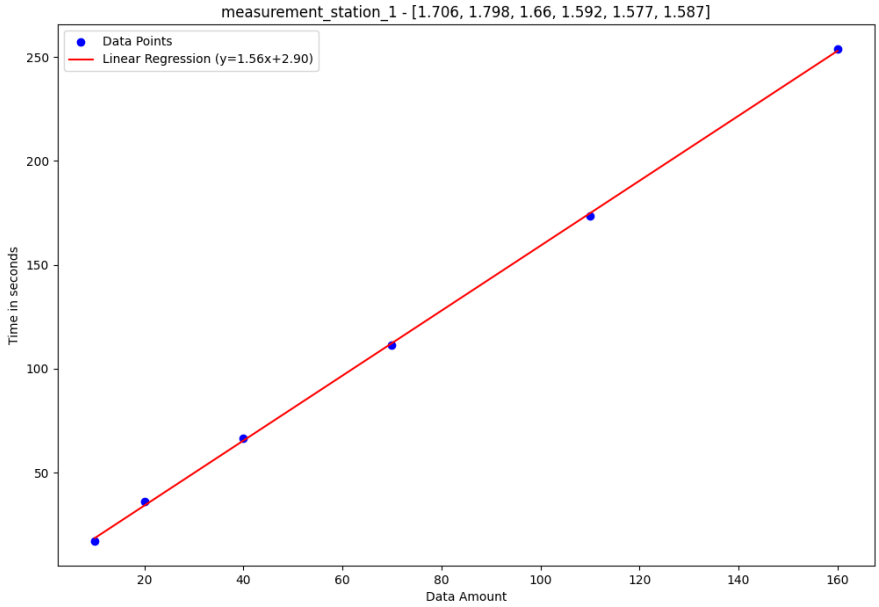


Figure 5.1: Rule-based Monitoring Tool isolated evaluation.

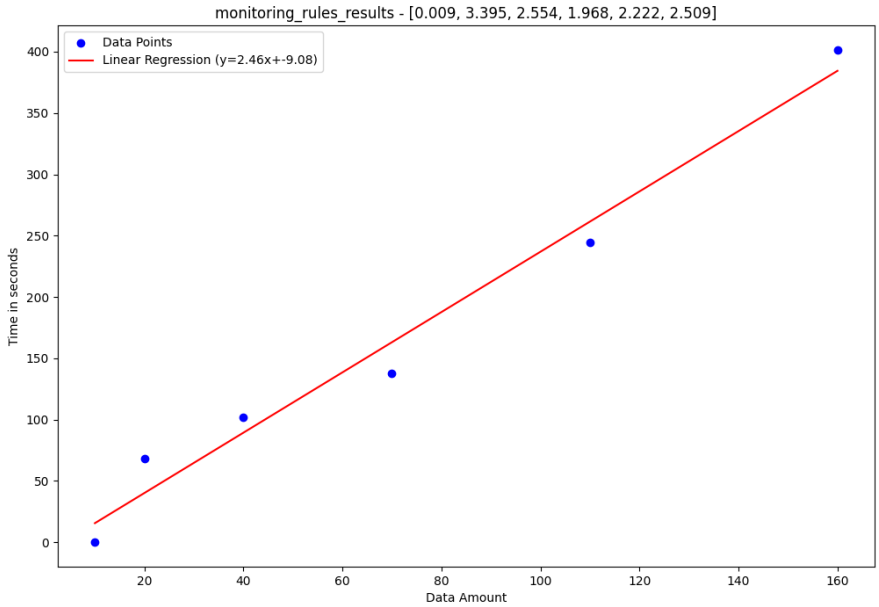


Figure 5.2: Rule-based Monitoring Tool evaluation in the openZDM infrastructure.

The slight variation in linear behavior observed when running the system within the openZDM infrastructure can be attributed to the fact that other micro-services running in parallel may impact resource usage, particularly CPU utilization. This interference from the shared infrastructure is likely the cause of the observed differences.

5.2 Diagnosis Performance Analysis

To evaluate the results presented in Tables 4.3 and 4.4, figures 5.3 and 5.4 show the relationship between the time elapsed during testing and the data processed and generated. A linear regression was applied to predict the behavior of this relationship, confirming its linear nature in both labeling-evaluation and classification-diagnosis micro-services within the openZDM infrastructure. Additionally, by analyzing the mean time elapsed per car processed at each collection point, the mean time to generate one output was 0,8888 seconds in the labeling and evaluation micro-service and 1,1026 seconds in the classification and diagnosis micro-service.

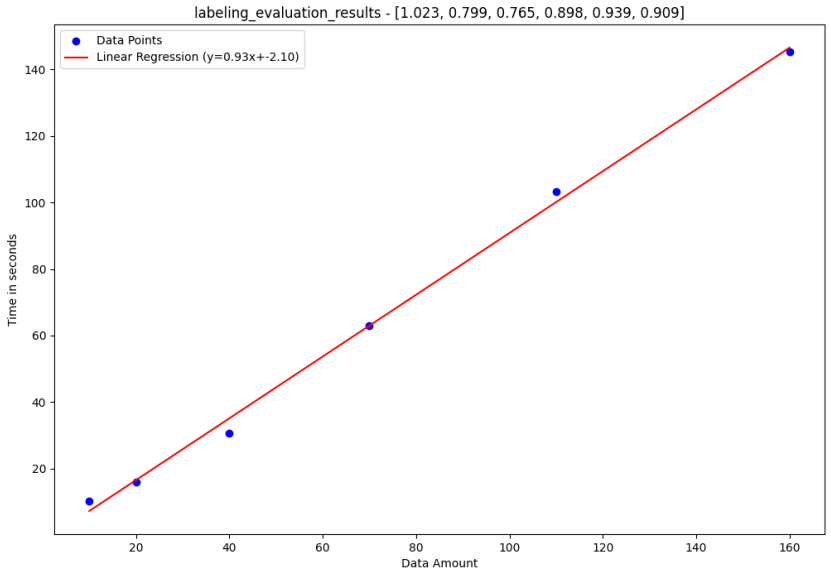


Figure 5.3: Labeling and Evaluation micro-service evaluation in the openZDM infrastructure.

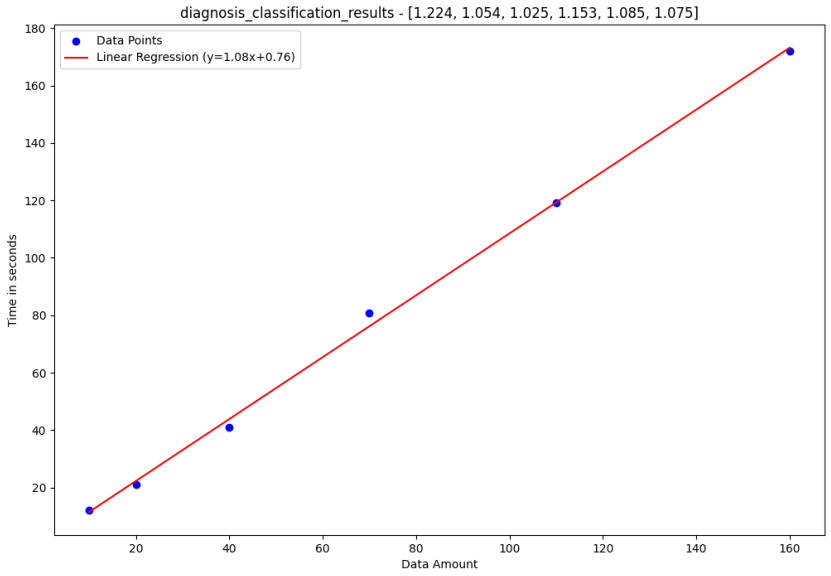


Figure 5.4: Diagnosis and Classification micro-service evaluation in the openZDM infrastructure.

5.3 Results Analysis in Achieving Zero Defect Manufacturing

During the development of the monitoring and diagnosis tools, follow-up meetings were held with industry leaders to test and gain qualitative approval from both end users and those responsible for the visualization tool. This process respected the data source limitations and result required for the industry use case and the ZDM principles.

To state the realization of the developed solutions in this work the appendices A contains the tool front-end developed by the openZDM team for the rule-based monitoring tool, while the visualization for the diagnosis tool is still a working in process until the publishing of this document.

For the quantitative analysis, both tools aimed to generate useful information respecting the data frequency for all measurement stations from the body shop. Finally it was concluded that based on the car frequency for each measurement station as described in

Section 3.3, showed mean collecting times of 208, 124, and 117 seconds for measurement stations 1, 2, and 3, respectively, while from the tools the worst time elapsed to process one car was 3,395 seconds, but with means not higher than 2,109 seconds demonstrating the efficiency of the proposed solutions in delivering results 30 times faster than the expected for the worst case scenarios across all body shop and assembly stations.

Chapter 6

Conclusion and Future Work

This work dealt with the exploratory analysis and development of two analytical solutions in monitoring and diagnosing measurement stations from a vehicular body shop and assembly industry lines. This was based on the importance and benefits that achieving zero defect manufacturing can provide for industries who aims to not only lower waste of materials, but of money and investments and also increase the productivity and product quality. The analysis and solution parts designed in functional and non-functional requirements delimited analysis objectives and what, how and where the system should work.

Moreover, both services were designed to effectively handle various manufacturing constraints, such as high-frequency data streams, missing or incomplete information, and non-normalized datasets. This is particularly relevant in achieving a Zero Defect Manufacturing environment, where the goal is to minimize errors and improve product quality through advanced monitoring and diagnosis enabling predictive maintenance techniques to be applied.

For the rule-based monitoring micro-service presented in this work it stands out for its innovative ability to adapt to emerging technologies and diverse use cases, but also to fit well to the proposed use case. This flexible and generalist approach aims to address the shortcomings identified in the related work. When applied to the industrial case study, the approach delivered promising outcomes, demonstrating its potential effectiveness in

tackling complex data monitoring and rule triggering within industrial and manufacturing environments.

A key strength of this tool lies in its support for the dynamic definition and configuration of contextualized, multipurpose rules. It can integrate Machine Learning, optimization techniques, and statistical algorithms into a unified platform, which allows for real-time adaptability and enhanced decision-making and other diagnosis tools.

The diagnostic tool excels in its methodology and calculation techniques based on industrial standards, using a defect indicator to quantify measurement errors and assess car and measurement quality. Designed for data with deviation and threshold formats, the system is also flexible in defining case-specific weights. Applied to an industrial case study, it showed promising results, effectively classifying cars by three different criteria (Label, Prediction and Rule) while identifying causes, consequences, and potential solutions for rapid adjustments in manufacturing environments.

Both the rule-based monitoring and diagnostic tools successfully met the qualitative goals by integrating with the openZDM platform, generating industry-approved insights, and adhering to Zero Defect Manufacturing standards. Quantitatively, the systems demonstrated their effectiveness by delivering results faster than the station frequency in the proposed use case.

Future work will focus on enhancing the rule-based tool capabilities by allowing the configuration of rule pipelines and a concurrency mode in which each rule will run in a different thread maximizing resource usage and performance. Additionally, efforts will be directed towards developing a configurator to simplify the rule definition, empowering high-level users to effortlessly create and manage rule sets. These improvements aim to further streamline the system's functionality and broaden its applicability across various domains. For the diagnosis tool focus will be targeted on incorporating and testing Machine Learning algorithms to analyze car and measurement quality, and on discovering patterns in the new supervised dataset to enhance predictive solutions.

The general objectives of the work were achieved and its development met all the requirements. The data analysis were able to retrieve useful insights, and the analytical

tools developed were able to achieve the requests from the industry, bypassing its limitations. The analyzed data, micro-service developed and tests devised generated value for industries and its contexts.

Bibliography

[1] Patricia Abelairas-Etxebarria and Inma Astorkiza. From exploratory data analysis to exploratory spatial data analysis. *Mathematics and Statistics*, 8:82–86, 2020.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994. VLDB Endowment.

[3] Mohammed Al-Zeyadi, Javier Andreu-Perez, Hani Hagras, Chris Royce, Darren Smith, Piotr Rzonsowski, and Ali Malik. Deep learning towards intelligent vehicle fault diagnosis. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020.

[4] S. Asadzadeh, A. Aghaie, H. Shahriari, and S. T. A. Niaki. Improving reliability in multistage processes with autocorrelated observations. *Quality Technology & Quantitative Management*, 12(2):143—-157, 2015.

[5] Michael Bayer. Sqlalchemy. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org, 2012.

[6] D. Bertsimas and J. Dunn. *Machine Learning Under a Modern Optimization Lens*. Dynamic Ideas LLC, 2019.

[7] Derek Collison. Nats.io. `https://nats.io`, 12 2011.

[8] Julio Costa, Alexandre O. Júnior, Jose Barbosa, Gleifer Alves, Andre P. Borges, Gisela Garcia, Rui Pires, and Paulo Leitao. Real-time rule-based monitoring tool to achieve zero defect manufacturing. Published to Springer "Studies in Computational Intelligence", 2024.

[9] P.B. Crosby. *Quality is Free: The Art of Making Quality Certain*. Mentor book. McGraw-Hill, 1979.

[10] Angel Dacal-Nieto, Juan José Areal, Miguel García-Fernández, and Marcos Lluch. Use cases and success stories of a data analytics system in an automotive paint shop. In *2020 Eighth International Symposium on Computing and Networking (CANDAR)*, pages 95–100, 2020.

[11] Usama M. Fayyad. Data mining and knowledge discovery: Making sense out of data. *IEEE Expert: Intelligent Systems and Their Applications*, 11(5):20–25, October 1996.

[12] The Apache Software Foundation. Apache avro™ 1.12.0 specification. `https://avro.apache.org/docs/current/spec.html`, 01 2012.

[13] Deutsches Institut für Normung (DIN). *DIN SPEC 91345: Reference Architecture Model Industrie 4.0 (RAMI4.0)*, 2016.

[14] Shradha Ghansiyal, Li Yi, Peter M. Simon, Matthias Klar, Marius Marvin Müller, Moritz Glatt, and Jan C. Aurich. Anomaly detection towards zero defect manufacturing using generative adversarial networks. *Procedia CIRP*, 120:1457–1462, 2023.

[15] D.R. Gunasegaram, A.S. Barnard, M.J. Matthews, B.H. Jared, A.M. Andreaco, K. Bartsch, and A.B. Murphy. Machine learning-assisted in-situ adaptive strategies for the control of defects and anomalies in metal additive manufacturing. *Additive Manufacturing*, 81, 2024.

[16] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg,

Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[17] Nian-Ze Hu, Chih-Hui Simon Su, Cihun-Siyong Alex Gong, Cheng-Jung Lee, Yong-Sheng Chen, Ching-Hsiang Yang, Ching-Ying Yeh, Zheng-Han Shi, and Jieh-Tsyr Chuang. Machine learning approach for robot diagnostic system. In *Machine learning approach for robot diagnostic system*, page 5 – 7, 2019. Cited by: 0.

[18] S. Ian. *Engenharia de Software, 9st.* BRASIL: Pearson Education, 2011.

[19] Prathap Irudayaraj and Saravanan P. Adoption advantages of micro-service architecture in software industries. *International Journal of Scientific & Technology Research*, 8:183–186, 09 2019.

[20] Sai Biao Jiang, Pak Kin Wong, Renchu Guan, Yanchun Liang, and Jia Li. An efficient fault diagnostic method for three-phase induction motors based on incremental broad learning and non-negative matrix factorization. *IEEE Access*, 7:17780–17790, 2019.

[21] Ioannis Kavakiotis, Olga Tsave, Athanasios Salifoglou, Nicos Maglaveras, Ioannis Vlahavas, and Ioanna Chouvarda. Machine learning and data mining methods in diabetes research. *Computational and Structural Biotechnology Journal*, 15:104–116, 2017.

[22] Natalia Koteleva and Nikolai Korolev. A diagnostic curve for online fault detection in ac drives. *Energies*, 17(5), 2024. Cited by: 3; All Open Access, Gold Open Access.

[23] Xiaolin Liu, Guanghua Li, Qiang Gao, Shanshan Gao, and Yanan Yu. An intelligent quality prediction and autonomous decision system for zero defect manufacturing in natural product manufacturing. *Computers & Industrial Engineering*, 191, 2024.

[24] Rui Pedro Lopes, Ahmed Ibrahim, Jose Barbosa, and Paulo Leitao. Microservices architecture to enable an open platform for realising zero defects in cyber-physical manufacturing. *Submitted to Logic Journal of the IGPL*, 2024.

[25] Hideyuki Maki and Yuko Teranishi. Development of automated data mining system for quality control in manufacturing. In Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, editors, *Data Warehousing and Knowledge Discovery*, pages 93–100, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg".

[26] Gökan May and Dimitris Kiritsis. Zero defect manufacturing strategies and platform for smart factories of industry 4.0. In Laszlo Monostori, Vidosav D. Majstorovic, S. Jack Hu, and Dragan Djurdjanovic, editors, *Proceedings of the 4th International Conference on the Industry 4.0 Model for Advanced Manufacturing*, pages 142–152, Cham, 2019. Springer International Publishing.

[27] Elisa Minnetti, Paolo Chiariotti, Nicola Paone, Gisela Garcia, Helder Vicente, Luca Violini, and Paolo Castellini. A smartphone integrated hand-held gap and flush measurement system for in line quality control of car body assembly. *Sensors*, 20(11), 2020.

[28] Thanga Murugan. Empowering industry 4.0: The vital role of microservice architecture. `https://www.linkedin.com/pulse/empowering-industry-40-vital-role-microservice-thanga-murugan-hxfoc/`, 01 2024.

[29] Arkadiusz Mystkowski, Adam Wolniakowski, Adam Idzkowski, Maciej Ciężkowski, Michał Ostaszewski, Rafał Kociszewski, Adam Kotowski, Zbigniew Kulesza, Sławomir Dobrzański, and Krzysztof Miastkowski. Measurement and diagnostic system for detecting and classifying faults in the rotary hay tedder using multilayer perceptron neural networks. *Engineering Applications of Artificial Intelligence*, 133:108513, 2024.

[30] Lloyd S. Nelson. The shewhart control chart—tests for special causes. *Journal of Quality Technology*, 16(4):237–239, 1984.

[31] NIST. Monitoring, diagnostics and prognostics for manufacturing operations. `https://www.nist.gov/programs-projects/monitoring-diagnostics-and-prognostics-manufacturing-operations`, October 2021.

[32] Sunhwa Park, Mijeong Shin, and Byoungha Ahn. Fault diagnostic of drain pump based on ai svm. In *Proceedings of 2020 International Congress on Noise Control Engineering, INTER-NOISE 2020*, 2020. Cited by: 0.

[33] RedHat. O que é arquitetura de microsserviços? `https://www.redhat.com/pt-br/topics/microservices/what-are-microservices`, May 2023.

[34] John Smith, Jane Doe, and Emily Brown. Domain-knowledge-informed functional outlier detection for line quality control systems. *Computers & Industrial Engineering*, 189, 2024.

[35] D. Spiegelhalter, G. Schlesinger, H. Hennemann, and F. Design. *A arte da estatística: Como aprender a partir de dados*. Zahar, 2022.

[36] S. A. Spiewak. *Automatic Supervision of Machine Tools*. Springer London, London, 1994.

[37] S.A. Spiewak, R. Duggirala, and K. Barnett. Predictive monitoring and control of the cold extrusion process. *CIRP Annals*, 49(1):383–386, 2000.

[38] Peter Trebuna, Miriam Pekarcikova, and Michal Dic. Comparing modern manufacturing tools and their effect on zero-defect manufacturing strategies. *Applied Sciences*, 12(22), 2022.

[39] Iury Michel Treuk, Alexandre O. Júnior, Rui Pedro Lopes, Gonçalo Mota, J. Joaquim Mira, and Paulo Leitao. Digitalization of industrial inspection assets

through the asset administration shell. In *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–6, 2024.

[40] John W. Tukey. Introduction to styles of data analysis techniques. In ROBERT L. LAUNER and ANDREW F. SIEGEL, editors, *Modern Data Analysis*, pages 1–11. Academic Press, 1982.

[41] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[42] Pingle Yang, Yalei Yang, and Yunfeng Lou. A business activity real-time monitoring platform based on rule engine. *Procedia Engineering*, 15:3744–3748, 2011.

[43] Ye Yuan, Guijun Ma, Cheng Cheng, Beitong Zhou, Huan Zhao, Hai-Tao Zhang, and Han Ding. A general end-to-end diagnosis framework for manufacturing systems. *National Science Review*, 7(2):418–429, 11 2019.

Appendix A

Code and Reports

This appendices contains all documents and implementations developed to support the conclusions from this document, and as all these documents are protected by a Non Disclosure Agreement, only accredited users are allowed to access these information.

Thesis Reports that support the data exploration and KDD methodology of section 3: `OpenZDM/data_analytics_tools/tree/main/JulioThesisReports`

Benchmark micro-service code: `https://github.com/OpenZDM/a_benchmark`

Rule-base monitoring micro-service code: `https://github.com/OpenZDM/a_monitoring_rules`

Diagnosis Tool - labeling and evaluation, classification and diagnosis micro-services code: `https://github.com/OpenZDM/a_diagnosis`

Front-end Visualization for the rule-based monitoring tool: `Julio%20Thesis%20Reports/Rule-Based%20Monitoring%20Front-end.png`

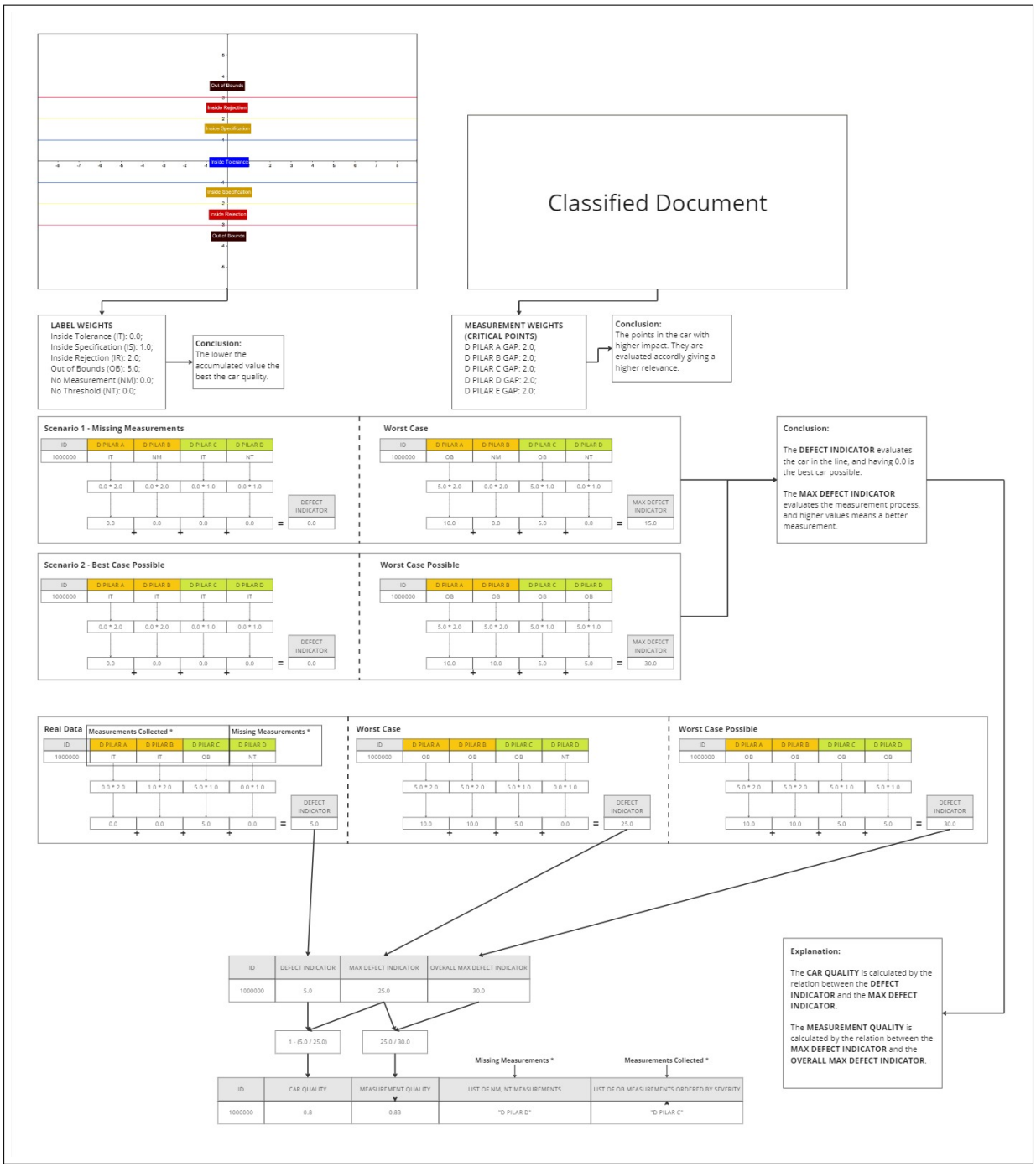


Figure A.1: Diagnosis Quality Evaluation Diagram.