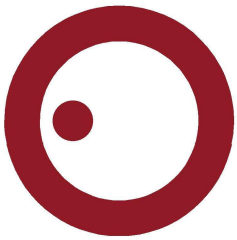


IPL

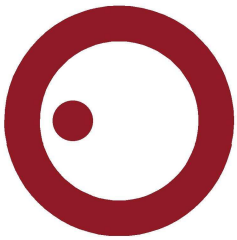
escola superior de tecnologia e gestão
instituto politécnico de leiria

Polytechnic Institute of Leiria
School of Technology and Management
Department of Computer Engineering
Master in Data Science

INJECTION MOLDING PROCESS MONITORING BASED ON MACHINE LEARNING ALGORITHMS

PEDRO ALEXANDRE DA PONTE COSTA

Leiria, September 2025



escola superior de tecnologia e gestão
instituto politécnico de leiria

Polytechnic Institute of Leiria
School of Technology and Management
Department of Computer Engineering
Master in Data Science

INJECTION MOLDING PROCESS MONITORING BASED ON MACHINE LEARNING ALGORITHMS

PEDRO ALEXANDRE DA PONTE COSTA
Number: 2220129

Dissertation under the supervision of:

Professor Sílvio Priem Mendes, PhD (smendes@ipleiria.pt)

Professor Paulo Jorge Gonçalves Loureiro, PhD (paulo.loureiro@ipleiria.pt)

João Patrício, MSc (joao.patricio@aspoeck.pt)

Leiria, September 2025

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to Professors Paulo Loureiro and Sílvio Mendes for their invaluable collaboration throughout this thesis. Their supervision, insightful guidance, and thoughtful counsel were fundamental in shaping this work and ensuring that I could derive meaningful value from the entire process.

I am equally grateful to Aspöck for their essential contribution, without which this work would not have been possible. Their provision of data, full collaboration, and generous invitation to visit their factory exemplify the spirit of academic–industrial partnership, fostering a relationship of mutual benefit. I am especially indebted to João Patrício, whose vision and dedication were central to the success of this thesis. From conceiving the project idea and initiating its proposal to facilitating data acquisition and offering continuous support, João played a pivotal role at every stage. His ability to bridge engineering practice with data-driven thinking guided key decisions and added exceptional depth to this work.

I would also like to thank my colleagues in the Master's program, whose support and companionship made both the coursework and this thesis an enriching and enjoyable experience. In particular, I wish to acknowledge Telmo Gregório, João Dinis, and Francisco Teixeira for their continuous encouragement, collaboration, and friendship throughout these past two years.

Finally, and most importantly, I am profoundly grateful to my family. To my mother and brother, thank you for your constant belief in me, your encouragement to pursue growth, and your steadfast support in striving toward ever greater goals.

ABSTRACT

This thesis addresses the detection of non-conformities in an injection moulding process using unsupervised and semi-supervised learning techniques, with the dual objective of identifying faulty production time zones that prompt the creation of non-conforming parts and enhancing process explainability for machine operators. The early detection of machine parameter deviations, such as abnormal temperature, pressure, torque, and cycle time variations, is essential to minimize economic losses, material waste, and sub-optimal product quality, while maintaining the machine in an optimal production state for longer periods.

A diverse array of methods, including Local Outlier Factor (LOF), Isolation Forest (IF), One-Class Support Vector Machine (One-Class SVM), OPTICS, Mean Shift, and Long Short-Term Memory (LSTM) networks, were experimented with to achieve this goal. Extensive feature engineering and selection were performed to balance dimensionality reduction with domain-relevant interpretability, enabling actionable feedback for process optimization.

The model pipeline was developed using Python, PyTorch, and Scikit-learn, containerized with Docker and accelerated using CUDA. Real-world sensor data spanning six months of continuous 24/7 operation were used for training and evaluation. The proposed LSTM-based approach, designed for time series modeling, achieved a weighted average F1-score of 0.94 on test data in predicting faulty production time zones of approximately seven-minute intervals. Evaluation metrics included accuracy, precision, and recall, with particular emphasis on the F1-score due to the imbalanced nature of the dataset and the critical need to minimize both false positives and false negatives.

A key aspect of this work lies in its commitment to data-driven development, grounded in the use of real, unfiltered sensor data from live industrial production. While raw data introduces noise and operational variability, it also provides a more faithful representation of the production environment, revealing edge cases and failure modes often absent in curated datasets. Addressing these challenges required robust preprocessing, careful validation, and a strong understanding of the process domain, ultimately enabling the development of a model better suited for real-world deployment and generalization.

The results demonstrate that this methodology provides a robust baseline for anomaly detection and process monitoring in injection moulding. Contributions include a reproducible framework for explainable unsupervised anomaly detection, validation of LSTM's effectiveness over static models, and novel feature reduction strategies, such as applying PCA to sensor groups while preserving domain interpretability. This work serves both as a practical tool for deployment and as a methodological reference and compendium of techniques for future research in data-driven industrial process optimization.

Keywords: Injection Molding, Defect Prediction, Data-Driven Solution, Industrial Sensor Data

RESUMO

Esta tese aborda a deteção de não-conformidades num processo de injeção plástica recorrendo a técnicas de aprendizagem não supervisionada e semissupervisionadas, com o duplo objetivo de identificar intervalos temporais de produção propícios a defeitos que conduzem à geração de peças não conformes e de reforçar a explicabilidade do processo para os operadores de máquina. A deteção precoce de desvios nos parâmetros da máquina, tais como variações anómalas de temperatura, pressão, binário e tempo de ciclo, é essencial para minimizar perdas económicas, desperdício de material e degradação da qualidade do produto, mantendo simultaneamente a máquina num estado ideal de produção durante períodos mais prolongados.

Um conjunto diversificado de métodos, incluindo *Local Outlier Factor (LOF)*, *Isolation Forest (IF)*, *One-Class Support Vector Machine (One-Class SVM)*, *OPTICS*, *MeanShift* e redes *Long Short-Term Memory (LSTM)*, foram explorados com este propósito. Foi realizada engenharia e seleção de features de forma extensiva, procurando equilibrar a redução de dimensionalidade com a interpretabilidade relevante para o domínio, permitindo fornecer informação acionável para a otimização do processo.

O pipeline do modelo foi desenvolvido em Python, PyTorch e Scikit-learn, num container de Docker e acelerado com CUDA. Dados reais de sensores, cobrindo seis meses de operação contínua 24/7, foram utilizados para treino e avaliação dos modelos. A abordagem de LSTM proposta, concebida para modelação de séries temporais, alcançou um *F1-Score* médio ponderada de 0,94 em dados de teste na previsão de intervalos de produção defeituosos de aproximadamente sete minutos. As métricas de avaliação incluíram *accuracy*, *precision* e *recall*, com particular ênfase no *F1-Score* derivado do caráter não balanceado do dataset e à necessidade crítica de minimizar tanto falsos positivos como falsos negativos.

Um aspeto central deste trabalho reside no seu compromisso com um desenvolvimento data-driven, sustentado pela utilização de dados reais e não filtrados provenientes de produção industrial em operação. Embora dados brutos introduzam ruído e variabilidade operacional, oferecem também uma representação mais fiel do ambiente produtivo, revelando casos limite e modos de falha frequentemente ausentes em datasets refinados. A abordagem a estes desafios exigiu pré-processamento robusto, validação cuidadosa e um conhecimento sólido do domínio do processo, permitindo, em última análise, o desenvolvimento de um modelo mais adequado à implementação no mundo real e à sua generalização.

Os resultados demonstram que esta metodologia fornece uma base robusta para a deteção de anomalias e monitorização do processo de injeção. As contribuições incluem um framework reprodutível para deteção de anomalias não supervisionada preservando explicabilidade de features, a validação da eficácia das LSTM face a modelos estáticos e estratégias inovadoras de redução de features, como a aplicação de PCA a grupos de

sensores mantendo a interpretabilidade do domínio. Este trabalho serve simultaneamente como ferramenta prática para implementação e como referência metodológica e compêndio de técnicas para investigação futura na otimização de processos industriais orientados por dados.

Keywords: Injecção Plástica, Previsão de Defeitos, Solução *Data-Driven*, Dados de Sensores Industriais

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CSV	Comma-Separated Values.
CUDA	Compute Unified Device Architecture.
GANs	Generative Adversarial Networks.
IF	Isolation Forest.
IQR	Interquartile Range.
KDE	Kernerl Density Estimate.
LOF	Local Outlier Factor.
LSTM	Long Short-Term Memory.
MDI	Mean Decrease Inpurity.
MI	Mutual Information.
MSE	Mean Squared Error.
One-Class SVM	One-Class Support Vector Machine.
OPTICS	Ordering Points To Identify the Clustering Structure.
PCA	Principal Component Analysis.
PCC	Pearson Correlation Coefficient.
PDF	probability density function.
RBF	Radial Basis Function.
RF	Random Forest.
RNN	Recurrent Neural Networks.
SPC	Statistical Process Control.

SVDD Support Vector Data Description.

XLSX Microsoft Excel Spreadsheet.

1

INTRODUCTION

1.1 CONTEXT AND APPLICATION

In recent years, Data Science as a field has steadily grown, leaving its footprint across various industries and establishing itself as a pivotal element for the success of many enterprises [1]. Many factors contribute to its widespread impact, from the development of advanced data analysis techniques to more commercially available hardware and an increased accessibility of technology to its users. More and more conditions are being met to cement this field's importance. In modern manufacturing, the integration of data analytics techniques has revolutionised traditional production processes, ushering in an era where data-driven insights drive enhanced efficiency and quality, this change in industrial approach as since been cemented as the "Fourth Industrial Revolution" or "Industry 4.0" [2].

Aspöck Portugal S.A. operates at the forefront of automotive component manufacturing, specialising in the production of rear lights, where precision and reliability are paramount. The production of technical plastic parts involves intricate processes, one of which is the injection moulding of thermoplastics. Despite robust equipment that predominantly meets quality standards, the occasional production of non-conforming parts is still a reality that seeks to be minimised, even more so in the automotive industry, a competitive market where customers require highly optimised and quality-assured processes.

During the injection moulding process, an abundance of data is generated from variables such as temperature, pressure, torque, time, and other mechanical properties. This data serves as a rich resource for uncovering insights into production dynamics and identifying deviations from optimal conditions. Like most real-world scenarios, not all data is in ideal condition for processing, and it might not be possible to extract the perfect data or adjust it to meet our needs.

This thesis explores the critical domain of injection moulding process monitoring, aiming to comprehend the data and find suitable methodologies for feature selection, consequently detecting outliers and relating them to product defects. To this end, a wide range of unsupervised and semi-supervised tools are employed, some of which are outlier detection, clustering and LSTM models. The findings from this project seek to enable early intervention to prevent defects and improve operational efficiency by reducing downtime and material waste.

1.2 OBJECTIVES

This work has three main objectives:

- Detection of Events/Anomalies: Employ machine learning techniques to identify deviations in process variables during the mass production of plastic parts.
- Model Benchmarking: Conduct a comparative evaluation of various machine learning models to determine their effectiveness in detecting events and anomalies, as well as their limitations and requirements.
- Explainability: Utilise the explainability features of the machine learning models to identify the most relevant variables that contribute to the occurrence of events and anomalies.

1.3 SCOPE AND LIMITATIONS

- **Scope:**
  - Focus on the injection moulding process of thermoplastics from a rear lighting housing mould at Aspöck Portugal S.A..
  - Analysis of data variables extracted during the mould-injection process.
  - Application of machine learning algorithms for anomaly detection and process monitoring.
  - Evaluation of the relationship between detected anomalies and product defects.
  - Extraction of explainabily and process insight from model results.
- **Limitations:**
  - Potential challenges in data quality and availability.
  - Unlabelled data.
  - Complexity of establishing direct causal relationships between process variables and product defects.
  - Limitations in generalising findings to other manufacturing processes or industries.

1.4 SIGNIFICANCE OF THE STUDY

This study's significance lies in its potential to enhance the quality control process in the manufacturing of technical plastic parts. By identifying and understanding process anomalies and their impact on product quality, this research aims to provide actionable insights that can lead to more efficient production processes, reduced defect rates, and

improved decision-making on the shop floor. By integrating explainable machine learning models into daily operations, this research seeks to empower decision-makers with the tools necessary to optimise production processes proactively.

Beyond academic exploration, this thesis aims to provide actionable insights directly applicable to Aspöck Portugal S.A. and similar manufacturing environments.

Additionally, it develops an extensive toolkit of methods and approaches for addressing unsupervised and semi-supervised problems where standard validation metrics might not be suitable.

1.5 Introduction to Plastic Injection Moulding

Plastic injection moulding is a widely used manufacturing process for producing various technical plastic parts. This process involves injecting molten plastic material into a mould cavity, where it cools and solidifies to form the desired part.

As a production process, it is a highly efficient method for mass production and allows for the creation of complex shapes with high precision and repeatability. In Figure 1 a very simplified diagram view of the process can be seen.

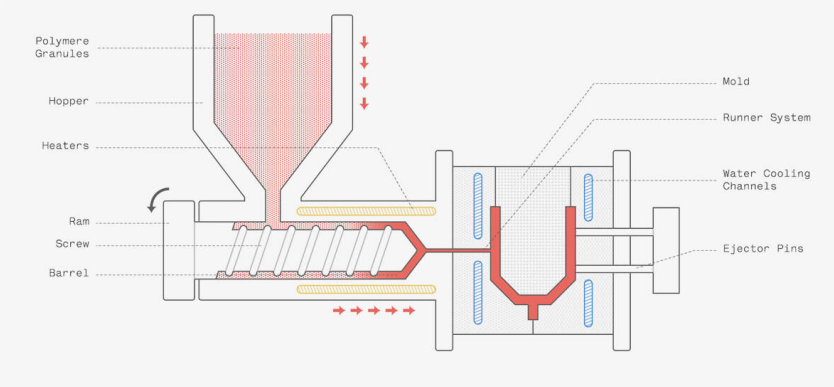


Figure 1: Injection Process Simplified [3]

For the most part the injection process can be broken down into 7 key stages:

1. **Clamping**: The initial step involves clamping the mould, which is typically composed of two or more metal plates. These plates are pressed together with the assistance of hydraulics in preparation for the injection.

2. **Injection**: Once the mould is securely clamped, the injection phase begins. Plastic granules or pellets are melted down into a molten state and injected into the mould.

3. **Packing**: Pressure is applied during a specific time frame to ensure that the molten plastic fills every area of the mould, creating an accurate replica of the mould's negative shape.

4. **Cooling**: In the cooling stage, the mould is left undisturbed so that the hot plastic can cool and solidify into a usable product. This process is usually aided by built-in cooling on the mould itself to accelerate cycle times and promote a more controllable and uniform cooling.

5. **Dosing**: In paralleled to the cooling of the part previously injected, dosing of a new portion of melted material is prepared for the next injection cycle.

6. **Mould Opening**: After the part has cooled, the mould slowly opens, separating the two plates and making the final product accessible.

7. **Ejection**: Finally, an ejector bar pushes the solidified product out of the mould core. Any waste material is trimmed off, and the product is finalised.

1.5.1 *Data Generation in Plastic Injection moulding*

The plastic injection moulding process generates a vast amount of data that is critical for monitoring, controlling, and optimising the process. The data can be broadly categorised into three types:

1. **Input Process Parameters**: These include data related to the injection moulding machine itself, such as injection pressure, injection time, screw position, and hydraulic pressure.

2. **Output Process Parameters**: These involve the conditions under which the moulding process occurs, including mould temperature, cycle time and screw rot time.

3. **Quality Metrics**: These refer to the characteristics of the final product, such as dimensions, weight, surface finish, and mechanical properties.

As far as this project goes, we focus on tackling the first two categories since those are the ones we have reliable access to. Out of the various features, some of them are inputted by the operator in the machine, while others are directly derived from the process, something we cover later on.

1.6 METHODOLOGY

1.6.1 *Project Phases*

The project was structured into several well-defined and interdependent phases, each contributing to the systematic development of a solution to the problem. The first phase focused on acquiring a deep understanding of the production line data, including its structure, semantics, and potential sources of variability and noise. This was achieved through close collaboration with domain experts and thorough exploratory data analysis.

Following this, an extensive research phase was conducted that included a review of the state of the art in anomaly detection, time series analysis, and unsupervised learning techniques. This phase served to ground the project in current scientific knowledge and to identify appropriate methodological directions.

Based on insights from the research phase, an approach was conceptualized, from an initial phase of tackling data issues and experimenting with various techniques for data cleaning and feature selection to later running various models, finding suitable parameters and validating results.

Simultaneously, the documentation and reporting process was initiated early in the project to ensure traceability and scientific rigour. This culminated in the writing of the thesis report, which was conducted in parallel with continued refinement of the technical solution. The final phase involved cleaning, validating, and consolidating both the solution and the accompanying documentation, ensuring coherence and completeness in both the scientific and engineering deliverables.

1.6.2 *Development Methodology*

Although no formal methodology was established at the outset, the project's progression naturally aligned with agile principles due to its iterative and research-driven nature. The practical execution of the work was characterized by flexibility, incremental development, continuous re-assessment, and responsiveness to new insights.

Tasks were organized dynamically, often driven by current obstacles or opportunities discovered during experimentation. For example, unexpected behaviours in model training might lead to additional data validation efforts, or improved understanding of the domain might require changes to preprocessing or feature selection. Such responsiveness to change is emblematic of agile practices, even if the process was not managed through formal agile tools.

This adaptive, self-directed approach also proved compatible with the nature of scientific research, where exploratory paths are seldom strictly pre-planned, and solutions must evolve in tandem with understanding. As such, while not governed by a defined development framework, the project reflects a pragmatic and iterative style that aligns with modern, agile-influenced research workflows.

"Agile methods appear to be particularly well-suited to research environments, where requirements are often emergent, and solutions are discovered through iteration and experimentation." [4] This further validates the organic fit between agile principles and the workflow adopted in this thesis.

1.6.3 *Development Environments*

The technical solution was developed within a containerized environment to ensure reproducibility, consistency, and ease of deployment. Specifically, `Docker` was employed to encapsulate all necessary dependencies, including machine learning libraries, data handling packages, and GPU computing interfaces.

Development took place on a Windows machine using the Windows Subsystem for Linux 2 (WSL2), which provides a native Linux kernel with efficient performance and deep integration with the host operating system. This environment enabled seamless usage of CUDA for GPU-accelerated deep learning tasks, particularly model training using frameworks such as PyTorch and TensorFlow.

The containerized setup also ensured that experiments could be executed reliably across different machines and that all collaborators could reproduce the environment without conflicts or version mismatches. This aligns with best practices in modern machine learning research, where experiment reproducibility and environment isolation are of paramount importance [5, 6].

1.7 STRUCTURE OF THE THESIS

This thesis is structured to unfold as a coherent narrative that chronicles the research journey from inception to conclusion. We follow a systematic and chronological approach to each component, with each chapter designed to contribute to a comprehensive understanding of the study's objectives, methodologies, findings, and implications.

- **Chapter 2 - Related Work:** This section reviews the existing literature and research relevant to the study. It critically examines previous approaches, methodologies, and findings, providing the foundation for the current research. Additionally it gives a ground understanding of the models used, how they work, the scope of their applicability, and their advantages and limitations.
- **Chapter 3 - Data Processing:** Details the methods and procedures employed in exploring, cleaning, and transforming the data. This chapter illustrates the process undertaken to ensure the quality, and readiness of the data for subsequent analysis and modelling. It details a lot of insight into the dataset, its various properties, and challenges encountered during data processing. This process is discussed transparently to highlight challenges, limitations and procedures employed in the processing of the data.
- **Chapter 4 - Feature Selection:** Focuses on the methodologies employed to identify and prioritise relevant features from the dataset. Techniques such as feature correlation analysis, explainability assessments, and validation procedures are elucidated to justify the selection of features critical to the modelling phase. This chapter dissects the

many facets of the features and how we might assess their relevance and information contribution to the model.

- **Chapter 5 - Development of Models:** Presents the culmination of the research efforts through the development and evaluation of various models. This chapter encompasses outlier detection models, clustering models, and LSTM models, each discussed in terms of methodology, results, and evaluations against predefined criteria. Comparative analyses provide insights into the strengths, limitations, and implications of each model type, offering a comprehensive perspective on their applicability to real-world scenarios. Metrics and validation methods employed are also detailed to better underline how each might suit the context.

- **Chapter 6 - Feature Explainability:** This chapter briefly goes over closing remarks regarding how to approach feature explainability and its relevance for the development of this project. It validates the data and insights achieved during model experimentation through the use of information analysis methods, to deterministically verify feature correlation to defects and compare scales of importance between features.

- **Chapter 7 - Conclusion:** This chapter summarises the main findings, implications, and contributions of the study. It looks back on the research journey, revisiting the objectives outlined in Chapter 1 and discussing how they were addressed through the methodologies and findings in the following chapters. Based on the insights from the study, recommendations for future research directions are proposed to ensure continuity and relevance beyond the current scope.

2

RELATED WORK

This chapter reviews existing literature on various methodologies, technologies relevant to the goal of the project and noteworthy research that has been done in this scope. The focus is unsupervised methods that can identify anomalous patterns without prior labelling of data, which is critical in industrial settings where faulty or anomalous data points may occur infrequently and without prior pattern knowledge. The discussion covers key outlier detection techniques, clustering methodologies for the segmentation of outliers, and real-time detection strategies using time-series data.

2.1 PREVIOUS RELEVANT WORK

The resolution of data problems related to industrial processes is no new endeavour, as industries are often the primary drivers of technological advancements and process optimizations. AI solutions have long been deployed, from image-based quality control to predictive maintenance strategies.

Industrial anomaly detection has been a crucial research area due to its impact on reducing downtime, improving quality control, and increasing efficiency. Traditional supervised approaches often require labelled datasets, which are challenging to obtain in industrial settings where faults occur infrequently [7]. Therefore, unsupervised techniques have gained traction, as they allow for the identification of deviations from normal behaviour without the prior need to create supervised datasets.

Several works have explored the use of unsupervised learning for anomaly detection in manufacturing. This project was not heavily based on any other previous work having most of its process decision making derived from direct problem solving and data limitations encountered along the way. That said, previous works were relevant to understand the state of the art and how various scenarios have been handled to achieve successful results, they proved mostly useful in the acquisition of knowledge regarding available techniques, and model usage.

In 'Anomaly Detection in Industrial Processes: Supervised vs. Unsupervised Learning and the Role of Explainability' [8], an unsupervised deep learning approach was utilized to analyse industrial sensor data and identify deviations indicative of anomalies in machine operations. This methodology exhibited significant resilience and accuracy in anomaly detection tasks, despite the absence of labelled data for training. The study underscores the efficacy of unsupervised models in industrial settings, where acquiring labelled data can be both costly and time-consuming.

Furthermore, they compared supervised and unsupervised learning algorithms in the context of anomaly detection for industrial processes, with a focus on transparency and interpretability. Their findings revealed that while supervised learning models offer reliable performance under certain conditions, the unsupervised approaches hold promise due to their adaptability. The study also introduced an explainability method, MLW-XAttentIon [8], designed to enhance the interpretability of transformer-based models, thereby facilitating trust in AI-driven anomaly detection solutions. This capability further extends the range of impact of AI solutions within the context of industries, where a black-box approach to the problem even if it yields strong results, leaves us with a lack of explainability that does not provide input for the improvement of the industrial process at stake, leaving us with just a model to predict issues based on current data and no actionable way or leads on how to reduce defect inducing factors.

Similarly, [9] conducted an extensive review of deep learning techniques tailored for anomaly detection, highlighting the critical role played by feature learning in enhancing detection performance. Their work emphasized the growing importance of self-supervised approaches, which leverage the structure of the data itself to generate labels, circumventing the need for manual annotation. Such methods are particularly advantageous in industrial environments characterized by diverse and high-dimensional sensor data, much like our current scenario.

Time is another very relevant variable for this project, production chains can be very often defined by cycles, and sequential relations between features is particularly important when looking at products outputted from the same process and machine, often bound by the slow adjustment of environmental variables, machine parameters and physical properties involved. In [10] we find this factor addressed, where various clustering-based anomaly detection techniques were evaluated on manufacturing datasets. The study highlighted that density-based methods, such as DBSCAN and LOF, outperformed traditional statistical methods due to their ability to handle noise and detect rare fault occurrences.

More recently, [11] discussed the integration of deep autoencoders with Statistical Process Control (SPC) techniques for real-time anomaly detection in sensor-driven industrial systems. The study demonstrated that combining reconstruction-based anomaly detection with traditional statistical methods could significantly improve fault detection rates while reducing false positives. Similarly, [12] proposed a hybrid approach utilizing Generative Adversarial Networks (GANs) and feature engineering techniques to detect anomalies in industrial production lines. This method showed strong generalization capabilities and the potential to adapt to evolving manufacturing conditions, key properties when dealing with a slowly adjusting environment and its subset of features.

A study particularly relevant to this research is ‘Anomaly detection in injection molding process data based on unsupervised learning’ [13], which has the title states, as a very similar settings to our scenario. The authors highlight the challenges associated with high-dimensional sensor data and propose methods such as clustering, PCA, and autoencoder-based approaches for detecting deviations in process parameters. Their findings demonstrate the importance of interpretable anomaly detection models, allowing process engineers to

diagnose faults effectively. Given that injection molding processes are characterized by complex, interdependent variables, their work underscores the necessity of robust feature selection and dimensionality reduction techniques to improve detection accuracy. The study further validates the suitability of unsupervised learning in factory environments where labelled fault data is scarce, reinforcing the direction of the current research.

This study proved to be very similar to our scenario in paper, having a strong similarity in context but mostly its goals of interpretability as a key factor of success. However, dataset properties and different challenges would ultimately motivate substantially different approaches to the classification problem.

In summary, the literature review highlights the growing interest in applying unsupervised learning techniques to industrial anomaly detection. Deep learning models, clustering approaches, and hybrid techniques integrating statistical process control remain prominent research directions. The challenges of interpretability, adaptability to evolving systems, and real-time deployment continue to drive research in this domain, paving the way for more advanced and scalable anomaly detection solutions. Nevertheless, our project addresses several aspects that remain underexplored in existing studies. Unlike many works that rely on pre-processed or curated datasets, we confront the reality of raw, messy sensor data as encountered on the factory floor. A strong emphasis is placed on explainability, ensuring that the results are not only accurate but also actionable for the engineers tuning the injection-moulding machines. Our approach is deliberately broad, testing a range of models and methods while keeping the engineering process itself at the centre of the investigation, with the goal of developing both a deep understanding and a versatile technical toolset to tackle the challenges inherent to this environment. Rather than dictating a purely data-driven path, the work integrates machine-learning insights with practical utility for operators, prioritizing solutions that can guide process improvements and support day-to-day decision-making.

2.2 DISTANCE/PROTOTYPE-BASED MODELS

Distance-based methods assume that normal data forms compact groups in feature space. Anomalies are identified as points that exhibit large distances from all cluster centroids or belong to small, low-density clusters. Although originally designed for clustering, this simple prototype-based approach remains a competitive baseline for unsupervised anomaly detection tasks.

2.2.1 *K-Means*

K-Means is a widely used clustering algorithm that partitions data into `k` clusters by minimizing intra-cluster variance. The algorithm iteratively refines cluster assignments and updates centroids until convergence. While primarily a clustering technique, K-Means can also be leveraged for outlier detection by identifying data points that are far from their

assigned cluster centroids. For the context of this project it was employed in both ways having more success as the later.

2.2.1.1 *Mathematical Formulation*

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with n observations, K-Means seeks to partition these points into k clusters $C_1, C_2, \ldots, C_k$ by minimizing the within-cluster sum of squared errors (WCSS), also known as inertia:

$$J = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2, \tag{1}$$

where μ_i represents the centroid of cluster C_i, computed as:

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j. \tag{2}$$

The algorithm iterates through two key steps:

1. **Assignment step**: Each point x_j is assigned to the cluster whose centroid is closest:
2. **Update step**: Each cluster centroid μ_i is updated as the mean of all points assigned to it.

The process repeats until centroids stabilize or the change in WCSS falls below a predefined threshold.

2.2.1.2 *K-Means for Outlier Detection*

K-Means is not explicitly designed for anomaly detection, but outliers can be identified by analyzing the distances of points from their respective centroids. Points with distances exceeding a predefined threshold, such as a multiple of the standard deviation, can be flagged as anomalies:

$$O = \{x_j : \|x_j - \mu_{c(x_j)}\| > \tau\}, \tag{3}$$

where τ is a chosen threshold and $c(x_j)$ denotes the cluster assignment of x_j. This method is particularly useful for detecting global anomalies, where an outlier significantly deviates from all clusters.

2.2.1.3 *Advantages of K-Means*

- **Efficiency**: K-Means is computationally efficient, with a time complexity of $O(nkd)$, where d is the dimensionality of the data [14].
- **Scalability**: It can handle large datasets efficiently, making it suitable for real-time industrial applications.

- **Interpretability**: The algorithm provides clear cluster structures, aiding process engineers in understanding process variations.
- **Versatility**: It is useful for both clustering and outlier detection in structured datasets.

2.2.1.4 *Drawbacks of K-Means*

- **Assumption of Spherical Clusters**: K-Means assumes clusters are convex and isotropic, making it ineffective for arbitrarily shaped clusters.
- **Dependency on** k: The need to specify the number of clusters k in advance can be challenging in unsupervised settings. Forcing us to test with brute force with various parameters and compare results.
- **Sensitivity to Initialization**: Poor initialization can lead to suboptimal clustering solutions.
- **Ineffective in High-Dimensional Spaces**: The Euclidean distance metric becomes less meaningful as dimensionality increases, affecting clustering accuracy.

2.3 DENSITY-BASED MODELS

Density-based methods detect anomalies by examining variations in the local density of data points. The underlying assumption is that normal instances occur in dense neighbourhoods, whereas anomalies appear in regions of significantly lower density. These techniques are particularly effective when the data contains clusters of varying shapes and sizes.

2.3.1 *Local Outlier Factor*

Local Outlier Factor (LOF) is a density-based anomaly detection algorithm that identifies outliers by comparing the local density of a data point to the densities of its neighbors. Unlike global anomaly detection methods, LOF is particularly effective in detecting local anomalies, where deviations are relative to surrounding data points rather than the entire dataset [15].

2.3.1.1 *Mathematical Formulation*

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with n observations, LOF quantifies the degree of outlierness of a point x_j based on the local density of its neighbors.

STEP 1: DISTANCE-BASED REACHABILITY LOF relies on the concept of *k-distance*, which is the distance of a point x_j to its k-th nearest neighbor. The *reachability distance* between two points x_j and x_i is defined as:

$$\text{reach-dist}_k(x_j, x_i) = \max(\text{k-distance}(x_i), d(x_j, x_i)), \tag{4}$$

where $d(x_j, x_i)$ represents the Euclidean distance between x_j and x_i.

STEP 2: LOCAL REACHABILITY DENSITY (LRD) The local reachability density (LRD) of a point x_j is computed as:

$$\text{LRD}_k(x_j) = \frac{|N_k(x_j)|}{\sum_{x_i \in N_k(x_j)} \text{reach-dist}_k(x_j, x_i)}, \tag{5}$$

where $N_k(x_j)$ represents the set of k nearest neighbors of x_j.

STEP 3: COMPUTING THE LOF SCORE The Local Outlier Factor score for a point x_j is defined as the ratio of the average LRD of its neighbors to its own LRD:

$$\text{LOF}_k(x_j) = \frac{\sum_{x_i \in N_k(x_j)} \text{LRD}_k(x_i)}{|N_k(x_j)| \cdot \text{LRD}_k(x_j)}. \tag{6}$$

A LOF score significantly greater than 1 indicates that x_j has a substantially lower local density than its neighbors, suggesting that it is an outlier.

2.3.1.2 *Advantages of LOF*

- **Local Sensitivity**: Unlike global methods, LOF detects anomalies relative to local density variations, making it effective in non-uniform datasets.
- **No Assumption on Data Distribution**: LOF does not assume a predefined cluster shape or distribution, making it robust for real-world datasets.
- **Effective for Multi-Density Data**: It can handle datasets with varying densities, where global thresholding techniques would fail.
- **Parameter Flexibility**: The parameter k can be adjusted to control sensitivity to anomalies.

2.3.1.3 *Drawbacks of LOF*

- **Computational Complexity**: LOF has a computational complexity of $O(n^2)$ in its naive implementation, making it costly for large datasets.
- **Sensitivity to Parameter** k: The choice of k can significantly impact performance, requiring domain-specific tuning.

- **Not Ideal for High-Dimensional Data**: In high-dimensional spaces, distance-based methods suffer from the curse of dimensionality, reducing LOF's effectiveness.

2.3.2 *OPTICS*

Ordering Points To Identify the Clustering Structure (OPTICS) is a density-based clustering algorithm that extends DBSCAN by providing a more flexible approach to detecting clusters of varying densities [16]. It is particularly useful for anomaly detection as it identifies points that do not fit well into any high-density region.

2.3.2.1 *Mathematical Formulation*

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with n observations, OPTICS processes the dataset by computing two key quantities: the *core distance* and the *reachability*, which together define the clustering structure based on density.

CORE DISTANCE: A given point x_i, is a core point if at least MinPts points exist within its ϵ-neighbourhood defined as $N_\epsilon(x_i)$. All points are considered, even those belonging to other clusters, so to each is assigned a core distance defined as:

$$\text{core-distance}_{\epsilon,\text{MinPts}}(x_i) = \begin{cases} \text{UNDEFINED} & \text{if } |N_\epsilon(x_i)| \leqslant \text{MinPts}, \\ \text{MinPts-dist}(x_i) & \text{otherwise.} \end{cases} \tag{7}$$

where $N_\epsilon(x_i) = \{x_j \in X \mid d(x_i, x_j) < \epsilon_{max}\}$ denotes the ϵ-neighborhood of x_i, and $d(x_i, x_j)$ is the distance between x_i and x_j.

REACHABILITY: Given two points x_i, x_j, the reachability of x_i with respect to x_j is defined as:

$$\text{reachability}(x_i \leftarrow x_j) = \max\left\{d(x_j, x_i), \text{MinPts-dist}(x_j)\right\} \tag{8}$$

CLUSTER ORDERING: OPTICS algorithm constructs a *cluster ordering* of a dataset, which is a linear arrangement of data points based on their local density connectivity. Unlike traditional clustering algorithms that produce explicit partitions of the data space, OPTICS outputs a sequence of points annotated with their *reachability*, a density-based proximity measure. Reachability, while originally called the the "reachability distance" is not a distance as defined in mathematics, because it is not symmetric. This asymmetry often causes confusion and, as such, is referred to as simply *reachability* [17].

This ordering captures the intrinsic clustering structure of the data at multiple density levels.

2.3.2.2 *OPTICS for Outlier Detection*

Unlike DBSCAN, OPTICS does not explicitly classify outliers however, as highlighted above, it instead produces a reachability plot that highlights sparse regions. Anomalies are detected as points with significantly higher reachability compared to their neighbours. These points are located in low-density regions, making them suitable candidates for outlier identification.

2.3.2.3 *Advantages of OPTICS*

- **Handles Clusters of Varying Densities**: Unlike DBSCAN, OPTICS adapts to different density regions, making it more versatile.
- **No Need to Predefine a Fixed** ϵ : The algorithm dynamically determines appropriate cluster boundaries instead of relying on a single distance threshold.
- **Identifies Outliers Naturally**: Points with high reachability distances are automatically flagged as anomalies.
- **Robust to Noise**: OPTICS is effective in detecting noise points that do not belong to any cluster.

2.3.2.4 *Drawbacks of OPTICS*

- **Computational Complexity**: The algorithm has a time complexity of $O(n^2)$ in its basic implementation, making it less efficient for large datasets.
- **Complex Parameter Tuning**: Choosing an appropriate `MinPts` value is crucial for optimal performance.
- **Lack of Explicit Clusters**: Unlike DBSCAN, OPTICS does not return a fixed set of clusters but requires post-processing to extract meaningful groups.

2.3.2.5 *Applications in Industrial Settings*

OPTICS is well-suited for industrial anomaly detection, particularly in manufacturing processes where data exhibits clusters of varying densities. In injection molding, it can detect unusual production cycles that deviate from normal operational patterns [13]. Additionally, in predictive maintenance, OPTICS identifies regions of abnormal sensor activity that indicate potential failures.

Due to its adaptability in clustering and anomaly detection, OPTICS can be considered a powerful tool for analysing complex industrial datasets where patterns are not easily separable using traditional methods. In contrast, it is often not easy to implement, having a high computational demand to process data, and parameter tuning not always being straightforward.

2.3.3 *Mean Shift*

The *Mean Shift* algorithm is a non-parametric, density-based clustering technique that identifies clusters by iteratively shifting each data point toward the nearest peak of the underlying data distribution, commonly called modes. This is achieved by estimating the gradient of the probability density function (PDF) using a *kernel function*, which determines the influence of neighbouring data points in the update step [18].

2.3.3.1 *Mathematical Formulation*

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$, the shift vector for a point x_i is computed as:

$$m(x_i) = \frac{\sum_{x_j \in N(x_i,h)} K(x_j - x_i)\, x_j}{\sum_{x_j \in N(x_i,h)} K(x_j - x_i)} \tag{9}$$

where:

- $N(x_i, h)$ represents the neighborhood of x_i within a bandwidth h.
- $K(x_j - x_i)$ is a kernel function that weights nearby points.

Each point is iteratively shifted according to $m(x_i)$ until convergence, forming clusters around local density maximums.

2.3.3.2 *Kernel Choice in the Mean Shift Algorithm*

FLAT KERNEL The flat kernel assigns equal weight to all points within a fixed radius h. It is defined as:

$$K(x) = \begin{cases} 1 & \text{if } \|x\| \leqslant h, \\ 0 & \text{if } \|x\| > h. \end{cases} \tag{10}$$

This results in a mean shift vector pointing to the centre of mass of the local neighbourhood. The uniform kernel is computationally efficient but results in discontinuous convergence paths and is sensitive to the choice of h.

GAUSSIAN KERNEL The Gaussian kernel assigns weights to neighbouring points based on their distance from x, decaying exponentially. It is defined as:

$$K(x) = e^{-\|x\|^2} \tag{11}$$

In this case, nearby points exert more influence, producing a smooth approximation of the PDF and smoother convergence. The Gaussian kernel is more robust to noise and is

better suited for complex data distributions, although it incurs a higher computational cost due to the use of exponential functions.

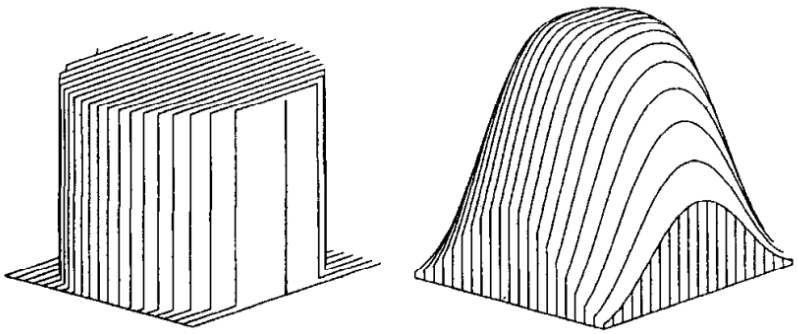


Figure 2: Respectively flat and Gaussian kernels [19]

Aspect	Flat Kernel	Gaussian Kernel
Weighting	Equal within radius	Distance-based (decaying)
Smoothness	Discrete jumps	Smooth convergence
Computational Cost	Low	High
Robustness to Noise	Lower	Higher
Best Use Case	Fast approximation, uniform data	Precise clustering, complex densities

Table 1: Comparison of flat and Gaussian kernels in Mean Shift.

2.3.3.3 *Mean Shift for Outlier Detection*

Outliers in Mean Shift are identified as points that do not converge to a meaningful cluster or require excessive iterations to stabilize. After clustering the dataset using the Mean Shift algorithm, each data point is assigned to a cluster defined by its corresponding cluster centre. To identify potential outliers within this clustering structure, we compute the Euclidean distance between each point and its associated cluster centre.

Formally, for a data point $\mathbf{x}_i$ assigned to cluster $c(i)$ with centre $\boldsymbol{\mu}_{c(i)}$, the distance is given by:

$$d_i = \sqrt{\sum_{j=1}^{d} \left(x_{ij} - \mu_{c(i),j} \right)^2}$$

These distances are used to define an outlier score for each data point. Points located farther from the centre of their assigned cluster are more likely to be anomalous. These outlier scores can then be sorted and ranked (retrieving the top-k most outlying samples) based direly on their distance to their respective cluster.

2.3.3.4 *Advantages of Mean Shift*

- **No Need to Predefine Number of Clusters**: Unlike K-Means, Mean Shift automatically determines the number of clusters based on data density.
- **Effective for Arbitrary Cluster Shapes**: It can detect non-linear and non-convex cluster structures.
- **Suitable for Density-Based Outlier Detection**: Points in low-density areas can be easily flagged as anomalies.

2.3.3.5 *Drawbacks of Mean Shift*

- **Possible Convergence to Poor Local Maxima**: Some data points may converge to suboptimal modes, leading to inconsistent clustering.
- **Computational Complexity**: The algorithm has a complexity of $O(n^2)$ for the Gaussian kernel, making it inefficient for large datasets.
- **Sensitivity to Bandwidth Parameter**: The choice of h for flat kernel heavily influences clustering results and must be carefully tuned.

2.3.3.6 *Applications in Industrial Settings*

Mean Shift is valuable in industrial applications where clustering structures are unknown or vary dynamically. Due to its ability to adapt to data distributions without requiring predefined clusters, Mean Shift is particularly useful for analysing complex manufacturing and process control datasets, and as such has been consistently used in various settings despite its simplistic approach.

2.4 TREE-BASED MODELS

Tree-based approaches to anomaly detection rely on recursive data partitioning to isolate individual observations. By randomly splitting the feature space into hierarchical structures, these methods exploit the principle that anomalies require fewer splits to be separated from the rest of the data. This makes tree-based techniques efficient, scalable, and naturally suited to high-dimensional datasets.

2.4.1 *Isolation Forest*

Isolation Forest is an anomaly detection algorithm based on decision trees. Unlike density-based methods, Isolation Forest explicitly isolates anomalies by constructing random decision trees [20]. The key idea is that anomalies are easier to isolate because they exist in sparse regions of the dataset.

2.4.1.1 *Mathematical Formulation*

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with n observations, Isolation Forest constructs an ensemble of isolation trees (iTrees), where each tree recursively partitions the data using randomly selected features and split values.

STEP 1: RECURSIVE PARTITIONING Each isolation tree is built by selecting a feature f and a split point s randomly within the range of f. The partitioning continues recursively until:

- A single data point remains in the node.
- A predefined depth limit is reached.

STEP 2: PATH LENGTH COMPUTATION The length of the path from the root to a given point x_j in the tree represents the number of splits required to isolate it. The average path length across multiple trees is used as a measure of normality. The expected path length for a dataset of size n is given by:

$$E(h(n)) = 2H(n-1) - \frac{2(n-1)}{n}, \tag{12}$$

where $H(n)$ is the harmonic number:

$$H(n) = \sum_{i=1}^{n} \frac{1}{i}. \tag{13}$$

Anomalies are expected to have shorter average path lengths, as they are isolated quickly due to their sparsity.

STEP 3: ANOMALY SCORE CALCULATION The anomaly score for a point x_j is computed as:

$$S(x_j, n) = 2^{-\frac{E(h(x_j))}{E(h(n))}}. \tag{14}$$

A score close to 1 indicates an anomaly, while a lower score indicates normal behavior.

2.4.1.2 *Advantages of Isolation Forest*

- **Efficiency**: With a time complexity of $O(n \log n)$, Isolation Forest is significantly faster than distance-based and density-based methods.
- **Scalability**: It is well-suited for large datasets due to its tree-based structure.
- **No Assumptions on Data Distribution**: Unlike statistical methods, Isolation Forest does not require prior knowledge about the data distribution.

- **Effective in High-Dimensional Spaces**: Unlike distance-based methods, it remains effective in high-dimensional datasets.

2.4.1.3 *Drawbacks of Isolation Forest*

- **Randomness Sensitivity**: The algorithm relies on randomly chosen splits, which may lead to variability in results.
- **Not Ideal for Small Datasets**: Performance degrades when applied to very small datasets, as the random partitions may not capture meaningful anomalies.
- **Limited Explainability**: Unlike clustering or density-based methods, Isolation Forest does not provide intuitive cluster assignments or density estimations.

2.4.1.4 *Application in Industrial Settings*

Isolation Forest is widely used in industrial anomaly detection due to its efficiency and ability to handle high-dimensional sensor data. In manufacturing, it is applied to detect defective production cycles by analysing deviations in machine sensor readings [13]. In predictive maintenance, it identifies unusual patterns in machine operation before failures occur.

2.5 BOUNDARY-BASED MODELS

Boundary-based methods aim to learn an enclosing surface that captures the region where the majority of data resides. By modelling the support of the normal data distribution, these techniques identify as anomalies any observations that fall outside the learned boundary.

2.5.1 *One-Class SVM*

One-Class Support Vector Machine (One-Class SVM) is a boundary-based anomaly detection algorithm that learns a decision function to separate normal data from potential anomalies. It is particularly useful when only normal data is available for training, making it well-suited for unsupervised anomaly detection tasks.

2.5.1.1 *Mathematical Formulation*

Given a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with n observations, the One-Class SVM aims to find a hyperplane that best separates the data from the origin in a high-dimensional feature space [21]. This is achieved by solving the following optimization problem:

$$\begin{aligned} \min_{w,\xi,\rho} \quad & \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \\ \text{subject to} \quad & w \cdot \phi(x_i) \geqslant \rho - \xi_i, \quad \xi_i \geqslant 0, \quad i = 1, \ldots, n, \end{aligned} \tag{15}$$

where:

- $w \in \mathcal{H}$ is the normal vector to the separating hyperplane in the feature space $\mathcal{H}$,
- $\phi(x)$ is a feature map that transforms the input data into $\mathcal{H}$,
- $\rho \in \mathbb{R}$ is the offset (bias term) of the hyperplane from the origin,
- $\xi_i \geqslant 0$ are slack variables that allow for soft violations of the margin,
- $\nu \in (0, 1]$ is a parameter controlling the trade-off between the fraction of outliers and the number of support vectors.

The decision function for a new observation x_j is defined as:

$$f(x_j) = \text{sign}(w \cdot \phi(x_j) - \rho). \tag{16}$$

A value $f(x_j) < 0$ indicates that x_j lies outside the learned support and is classified as an outlier.

2.5.1.2 *Kernel for Non-Linear Decision Boundaries*

To handle non-linearly separable data, the One-Class SVM employs the kernel trick to implicitly map data into a high-dimensional space without computing $\phi(x)$ explicitly [22]. A commonly used kernel is the Radial Basis Function (RBF) kernel:

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right), \tag{17}$$

where $\gamma > 0$ is a kernel parameter that defines the influence of individual training examples. The choice of kernel and its parameters critically affects the tightness of the data description and the sensitivity to anomalies.

2.5.1.3 *Support Vector Data Description (SVDD)*

Support Vector Data Description (SVDD), introduced by Tax and Duin [23], is a kernel-based method for one-class classification and anomaly detection. Unlike One-Class SVM, which separates data from the origin in a high-dimensional feature space, SVDD constructs a minimal-volume hypersphere that encloses the majority of the data points. The objective is to find the center a and radius R of this sphere while allowing for some outlier points via slack variables ξ_i. The optimization problem is formulated as:

$$\min_{R,a,\xi} R^2 + C \sum_{i=1}^{n} \xi_i, \tag{18}$$

subject to:

$$\|\phi(x_i) - a\|^2 \leqslant R^2 + \xi_i, \quad \xi_i \geqslant 0, \quad i = 1, \ldots, n, \tag{19}$$

where $\phi(x_i)$ maps the input into a feature space, and $C > 0$ is a regularization parameter controlling the trade-off between sphere size and tolerance for outliers.

Aspect	One-Class SVM	SVDD
Geometric Interpretation	Separates data from the origin using a hyperplane	Encloses data in a hypersphere
Optimization Goal	$\min_{w,\xi,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n} \xi_i - \rho$	$\min R^2 + C \sum \xi_i$
Margin Definition	Distance from origin in feature space	Radius in input (or feature) space
Mapping Function	Uses kernel $\phi(x)$ or kernel trick	Also supports kernelization
Historical	Schölkopf et al., 2001	Tax & Duin, 2004

Table 2: Comparison between One-Class SVM and Support Vector Data Description (SVDD).

2.5.1.4 *Advantages of One-Class SVM*

- **Effective in High-Dimensional Spaces**: Unlike distance-based methods, One-Class SVM handles high-dimensional data efficiently.
- **Flexible Decision Boundaries**: Kernel functions enable the modeling of complex decision surfaces for detecting anomalies.
- **Useful for Novelty Detection**: It is well-suited for scenarios where only normal data is available for training.

2.5.1.5 *Drawbacks of One-Class SVM*

- **Computationally Expensive**: The algorithm has a complexity of $O(n^2)$ in its basic implementation, making it less scalable for large datasets.
- **Sensitive to Kernel Choice**: Performance heavily depends on the selected kernel and hyperparameters.
- **Poor Performance on Imbalanced Data**: If anomalies are not well-separated from normal data, the model may struggle to distinguish them.

2.5.1.6 *Application in Industrial Settings*

One-Class SVM is commonly used in industrial settings for detecting abnormal machine behaviour, process deviations, and quality control defects [13].

Due to its ability to learn from normal data alone, One-Class SVM is particularly effective in environments where anomalies are rare and labeled examples of faults are unavailable. On the other hand, a non-normal distribution of data proves difficult to handle quite considerably lowers the effectiveness of this type of model, as such its applicability is limited in many settings but can be key in some scenarios.

2.6 TIME-SERIES / SEQUENTIAL MODELS

Time-series models focus on capturing temporal dependencies and sequential patterns in data. Instead of relying on static feature distributions, these approaches learn the normal evolution of a signal over time and flag deviations from the expected sequence as anomalies.

2.6.1 *Long Short-Term Memory (LSTM)*

Long Short-Term Memory networks are a class of Recurrent Neural Networks (RNN) designed to model sequential data. They are particularly effective for time-series anomaly detection due to their ability to capture long-range dependencies. By learning normal temporal patterns in sensor data, LSTM-based models can detect deviations indicative of anomalies or system failures while accounting for slow and normal deviations of the cycle, not treating all time lapse as the same instance of data [24].

2.6.1.1 *Mathematical Formulation*

LSTM networks mitigate the vanishing and exploding gradient problem [25] inherent in standard RNNs by introducing gating mechanisms that regulate information flow. It achieves this by using different feedback loops for long-term and short-term memories.

The long-term memory, represented by the cell state c_t, can be modified, but is not directly influenced by weights or biases. This structural characteristic allows the cell state to preserve gradients over long time intervals, thus alleviating the vanishing gradient issue.

The short-term memory, represented by the hidden state h_t, is directly modulated by trainable weights, allowing it to adapt rapidly to short-term dynamics in the input sequence.

The mathematical operations within an LSTM cell at time step t are as follows:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{20}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{21}$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{22}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{23}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{24}$$

$$h_t = o_t \odot \tanh(c_t) \tag{25}$$

Where:

- x_t is the input vector at time t.
- h_{t-1} is the hidden state vector from the previous time step.
- c_{t-1} is the cell state vector from the previous time step.
- f_t, i_t, and o_t are the forget, input, and output gate vectors, respectively.
- $\tilde{c}_t$ is the candidate cell state vector.
- c_t and h_t are the updated cell state and hidden state vectors, respectively.
- W and U are weight matrices, and b are bias vectors for their respective gates.
- σ denotes the sigmoid activation function, and $\tanh$ denotes the hyperbolic tangent function.
- $\odot$ represents element-wise multiplication.

The LSTM model can be deconstructed into three key stages that work together to regulate memory:

1. FORGET GATE (MEMORY DISCARDING STAGE):

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

This gate decides the fraction of the previous cell state c_{t-1} to retain.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid activation function outputs values between 0 and 1. A value close to 0 implies forgetting most of the previous value in memory, whereas a value close to 1 implies near full retention.

2. INPUT GATE (MEMORY UPDATE STAGE):

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

This stage determines which parts of the new candidate information $\tilde{c}_t$ is to be added to the cell state. While $\tilde{c}_t$ combines the short-term memory and the input to create a potential long-term memory, i_t determinates what percentage of that potential memory is added to the long-term memory.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

The $\tanh$ function squashes inputs to the range $(-1, 1)$ that is then multiplied by the percentage factor retrieved from i_t. Using a $(-1, 1)$ range in the activation function provides two key advantages compared to sigmoid activation function:

- Helps keep the cell state c_t centred around zero, improving gradient flow and stability.
- Allows for negative updates, if i_t is high and $\tilde{c}_t < 0$, the model intentionally forgets or reduces aspects of current memory, tuning better to the data.

On this processing stage we also have what can be called as the *Cell State Update*. The new cell state c_t (long-term memory) integrates the effects of the forget and input gates and is updated:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

3. OUTPUT GATE (MEMORY EXPOSURE STAGE):

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

This stage governs which parts of the cell state contribute to the output at the current time step. Once again a sigmoid activation function is used to determine what percentage the LSTM remembers from the current input and h_t value (short-term memory) achieving a new value of h_t.

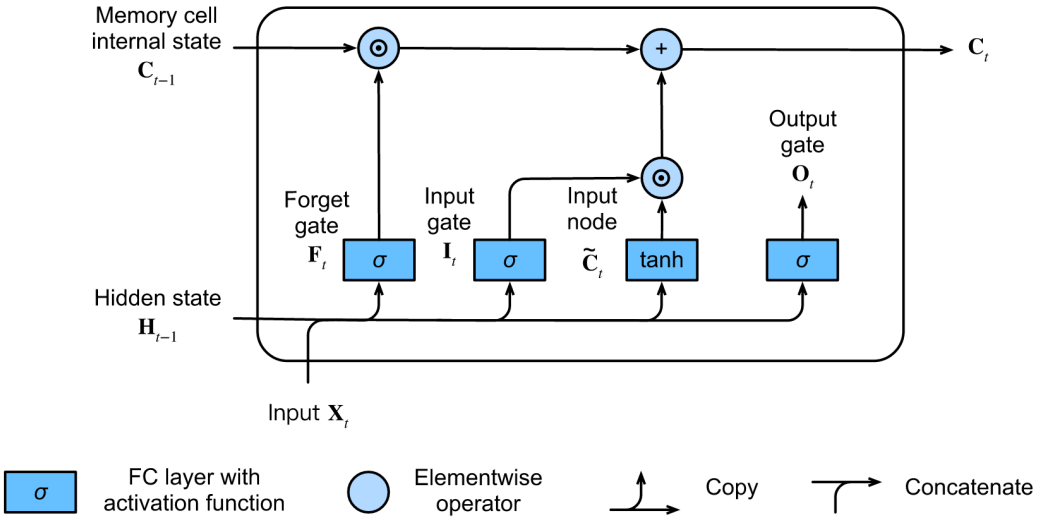


Figure 3: Illustration of an LSTM cell architecture with gating mechanisms [26].

2.6.1.2 *LSTM for Outlier Detection*

A trained LSTM model captures the normal behaviour of a process by learning to predict future time steps based on past observations. Anomalies can then be identified by measuring the prediction error, typically via MSE, and comparing it to a predefined or dynamically learned threshold [27], [28].

Let $\hat{x}_{t+1}$ be the predicted next value and x_{t+1} be the actual observed value. The prediction error is computed as:

$$\text{MSE}_t = \frac{1}{n}\sum_{i=1}^{n}(x_{t+1}^{(i)} - \hat{x}_{t+1}^{(i)})^2$$

An anomaly is flagged if MSE_t exceeds a threshold θ:

$$\text{Anomaly}_t = \begin{cases} 1, & \text{if } \text{MSE}_t > \theta \\ 0, & \text{otherwise} \end{cases}$$

Alternatively, instead of providing a fixed threshold for the deviation of the data that would lable an outlier, MSE values can be sorted to filter for the top N-percent of values as outliers,

2.6.1.3 *Advantages of LSTM*

- **Captures Temporal Dependencies**: LSTM effectively models sequential patterns, making it ideal for time-series anomaly detection.
- **Handles Long-Term Dependencies**: Unlike traditional RNNs, LSTM retains relevant past information over long sequences.

- **Robust to Noise**: The gating mechanism mitigates sensitivity to noisy data.
- **Versatile for Various Anomaly Detection Tasks**: It can be applied to predictive maintenance, quality monitoring, and process control [29].

2.6.1.4 *Drawbacks of LSTM*

- **Computationally Expensive**: Training LSTMs requires significant computational resources.
- **Sensitive to Hyperparameters**: Performance is highly dependent on proper tuning of sequence length, number of layers, and learning rate.
- **Black-Box Nature**: LSTMs lack interpretability, making anomaly explanations difficult if not impossible in industrial settings [30].

2.6.1.5 *Applications in Industrial Settings*

As previously stated, such models are particularly suitable for detecting subtle or context-aware anomalies in applications such as sensor monitoring in industrial systems, where deviations may not be abrupt but rather emerge gradually over several cycles. It accounts for slow shifts in data and takes in time as a key element adjusting along with the sequential data, which simulates extremely well the setting of most production process and its variations. This allows for it to far outperform static methods for this kind of scenarios, however at the cost of high computational demand and a need for field expertise to fine tune hyperparameters. LSTM as since been used in various similar settings to detect irregular cycles in sensor data [13], and identifying abnormal machine vibrations or temperature fluctuations [28].

2.7 COMPARATIVE OVERVIEW OF MODELS

Model Type	Label Need	Model Complexity	Explainability	Computational Cost	Outlier Sensitivity
K-Means	No	Low	High	Low	Moderate
LOF, OPTICS, Mean Shift	No	Moderate	Moderate	Moderate	High
Isolation Forest	No	Moderate	Moderate	Moderate	High
One-Class SVM	No	Moderate	Low	Moderate	Moderate
LSTM	Yes	High	Low	High	High

Table 3: Comparison of anomaly detection model families across key practical criteria.

To complement the individual model overviews, Table 3 summarizes the key characteristics of the presented approaches. The comparison highlights practical considerations such

as data requirements, computational burden, and interpretability that often influence model choice in industrial settings, where sensor data is high-volume, noisy, and rarely labeled. Despite these challenges, all models were thoroughly evaluated, and the final selection was guided by their ability to best support the overarching project goals.

2.8 SUMMARY

This chapter surveyed the principal approaches to unsupervised anomaly detection that are relevant to industrial manufacturing. We reviewed distance-based and density-based methods (e.g., K-Means, LOF, OPTICS, Mean Shift), tree-based and boundary-based models (Isolation Forest, One-Class SVM/SVDD), and sequential models (LSTM) that exploit temporal dependencies. Prior work highlights the trade-offs between scalability, interpretability, and sensitivity to high-dimensional on non-stationary data, as well as the importance of combining statistical techniques with machine-learning models for robust detection.

While previous studies provide valuable methodological insights, most rely on curated datasets or emphasize pure predictive accuracy. In contrast, the present research builds on these foundations by working directly with raw sensor data and prioritizing explainability, aiming to deliver actionable insights that can guide real-world process improvements rather than only accurate anomaly flags. This chapter also examined the inner workings of each model type, outlining their fundamental principles, application requirements, and comparative performance across key aspects.

3

DATA PROCESSING

In this chapter, we tackle the original dataset, going over its properties, highlights and state of quality. We approach the steps taken to import the data, deepen our understanding of the material we are working with, and make some preliminary adjustments to handle the data better. Following that, we go into a whole sector of data exploration, where we undertake a lengthy process of various methods to assess the data and its features, to evaluate what route to take and how to proceed with the project goals.

3.1 DATA COLLECTION

The data used for this study was provided by Aspöck. It comprises three months of sensor data extracted from one of their many injection moulding machines. Every injection cycle, the machine records data from up to 264 sensors, which are by-products of various physical properties generated in the injection process, such as temperature, pressure, and time, among others. Each injection produces two plastic pieces, which are then conveyed via automats to an operator who verifies their condition and evaluates if they are conforming, setting aside the non-conforming pieces to be scrapped.

This translates to us having access to a time span during which that specific machine made 71016 injection cycles, resulting in double that number of pieces produced.

3.2 DATA PROPERTIES AND FEATURE METADATA

The data is found in CSV format, and with it, we also have access to an additional Microsoft Excel Spreadsheet (XLSX) file containing metadata info regarding each feature, namely, name, description and category as per Table 4.

We make full use of this information and even add additional columns to the Metadata table, as it provides great insight into understanding each feature and how they might relate to or contribute to the model. Given our project context, explainability is paramount. As such, during the development of the various sections of the project, we have a very considerate approach to how to handle features and how to retain as much as possible their information to be able to trace back their importance and impact on the actual injection, as a means to provide tools for the business to extract value out of the insight gained during the project.

Short Name	Description	Category
SZx	Effective value of cycle counter	All
H2x	Real value of temperature zone 2	Injection
Pakt 2	Heating power in zone 2	Injection
...	...	...
tgf09	Flow thermoregulator zone 9	-
tgf01	Flow	-
StZx	Piece counter real value	-

Table 4: Table of features Metadata

As previously stated, the original dataset is in a format of 71016 rows and 264 columns. Additionally, there is a first row of labels in the format of the feature name, joined by its measuring unit in brackets. All the data is formatted as floats aside from the feature `"Tem"` being a string that we later convert to a date-time format.

3.3 DATA CLEANING

The changes made to the dataset as cleaning procedures were minimal since the original dataset was found to be in very good shape. Out of the changes made, it is important to highlight the following:

- **Labels:** Removal of the units of measure from the feature label and their insertion on the Metadata file. Original labels were formatted as Name[Unit of Measure], for example, *tgf02[l/min]*
- **Handling Missing Values:** A function was built to check for missing values, but none were found.
- **Removing Duplicates:** Only one instance of duplicate value was found. There were two features tracking injection cycles counter.
- **Removal of non-relevant data:** From the initial 264 features available in the data, we narrowed it down to 47 features. This selection was based on a few metrics:
  - Features that are not being used by that specific mold, such as temperature sensors that are inactive and are present in the dataset always with a value of 0.
  - Constant features that, as such, have no contribution to the model's outcome.
  - Unreliable features. For instance, instant heating power measurements. Having inconsistent reading values and only working to keep the heating zone at the designed temperature, which is a feature we do use.
  - Features measuring cinematic properties of the machine movement outside cycles that have no relation to the outcome of the produced piece.

- **Inadequate Data Correction:** Inconsistencies were reviewed and corrected. Due to the nature of the process, every time there is a stop in production, we can account for some changes in the data derived from its resting state. To diminish the noise of outlier data pertaining to these instances, we go over the dataset and for every 2-minute gap in sensor data, we remove the following 50 cases of data, a range considered adequate for the machine to return to ideal working conditions (Listing 1). This interval can be adjusted, and the values chosen were based on process knowledge and testing.

Listing 1: Filtering data instances from post-cycle stops

```
def drop_rows_after_reset(df: DataFrame, num_rows_to_drop=50, mins_threshold=2) -> DataFrame:

    rows_to_drop = []
    threshold_breach_counter = 0

    for index, row in df.iterrows():
        prev_row_time = df.iloc[index-1]['DateTime']
        row_time = row['DateTime']
        time_diff = (row_time - prev_row_time)
        if time_diff.total_seconds() / 60 > mins_threshold:
            threshold_breach_counter = threshold_breach_counter + 1

            for i in range(num_rows_to_drop):
                rows_to_drop.append(index - i)

    df.drop(index=rows_to_drop, inplace=True)
    df.reset_index(drop=True, inplace=True)
    return df
```

3.4 DATA UNDERSTANDING AND EXPLORATION

It is paramount that we have a good understanding of the data and its underlying properties to make informed decisions about how to process and refine it for model implementation. In this chapter, we undertake a comprehensive exploratory analysis to deepen our knowledge of the various features and their relationships within the dataset.

3.4.1 *Data Distribution*

As a first approach to evaluating the data distribution and attaining valid insight, a Shapiro-Wilk test on normality was made for each feature for a significance level of $\alpha = 0.05$. Unlike intuition would have led us to believe none of our features has a normal distribution for the significance level attributed. Upon further inspection, with the support from frequency plots, we are able to a certain extent to group the features based on how they are generated to understand their distributions better:

- **Assigned Values Features:** There are features with values inputted by the process engineers in the machine computer that are assigned to stay at that value. The constant ones that are not prone to change due to the injection process have previously been filtered out. However, there are features like zone temperatures that are set for a

specific value and still oscillate as a response to the injection process. They are tuned in real-time by the machine to try to stay adjusted to the desired value. As a result, we get a distribution that is mostly constant to the same value but still has minor oscillations. However, for our data set, there are different values to which the feature has been assigned, this is especially noticeable for the feature H17x, Figure 4, that most frequently has had the value of temperature assigned to 285 ºC, but if we limit the frequency variable of the plot, we are able to observe other values to which it has been assigned for some injection cycles, Figure 5.

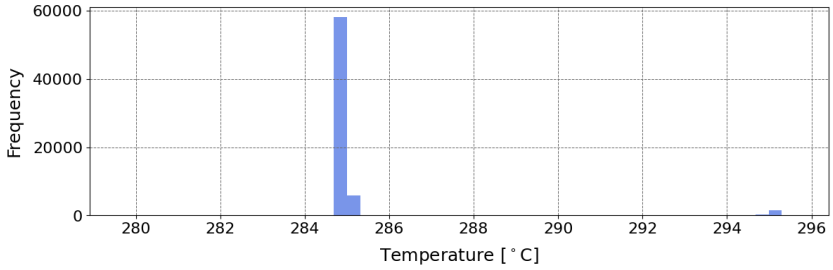


Figure 4: Distribution of feature H17x - Real Temperature value in zone 17

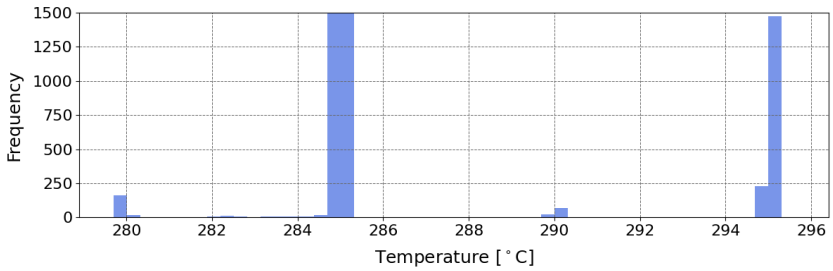


Figure 5: Distribution of feature H17x: Plot Frequency limited to 1500

As previously mentioned, this feature value still oscillates, however minimally, which can be seen in Figure 6, where we frame the plot only around the values of 295 and its variance.

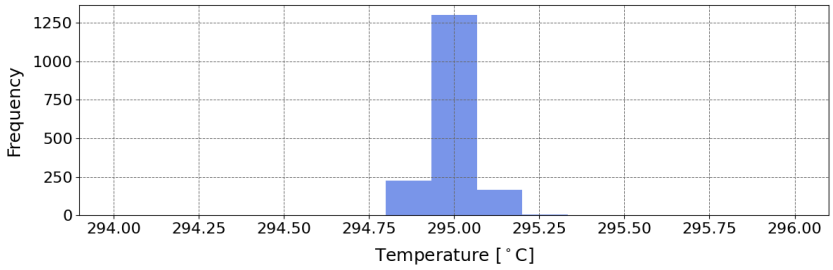


Figure 6: Distribution of feature H17x: Plot Framed around value 295

- **Process Derived Features:** Unlike the previously mentioned set of features these take their values derived from the process itself and often reflect directly the physical demands of the process. Some have a very similar distribution to the Gaussian

distribution with the various spikes, due to their correlation to some assigned feature that also changes, Figure 7.

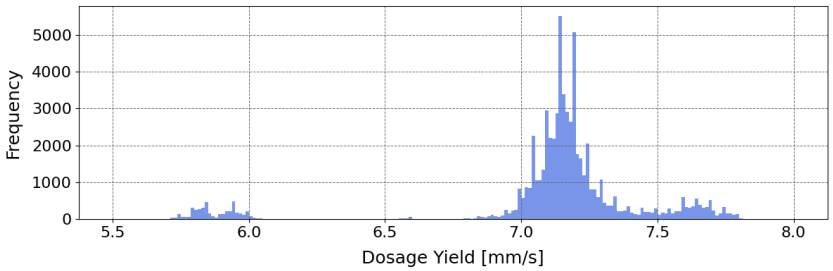


Figure 7: Distribution of feature iwdls - Dosage yield

Other features like the feature "Ms" that represents the peak value of torque during the dosage process still derive their values from the process, but in contrast, due to the physical nature of the property where it represents the maximum value achieved during the entire injection, its values are less prone to reflect directly any changes to the assigned features and more closely resembles the Gaussian Distribution with less to no noise, Figure 8. Even so, none of these features achieve what we would consider a Gaussian distribution for the α value previously considered.

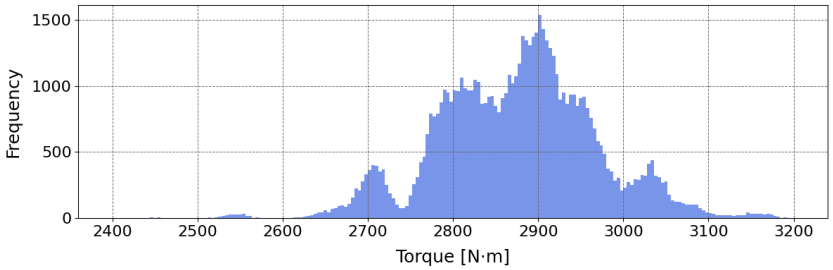


Figure 8: Distribution of feature Ms - Peak Torque Value During Dosage

3.4.2 *Quantifying Outliers*

Despite the feature distributions not being adequately represented by Gaussian distribution, the first approach made to measure outliers was using Interquartile Range (IQR). Even though its appropriateness for the context may be questionable, IQR still functions as a great gauging tool to quantify outliers.

IQR works as a measure of statistical dispersion. It is calculated as the difference between the third quartile (Q3) and the first quartile (Q1):

$$\mathrm{IQR} = \mathrm{Q3} - \mathrm{Q1} \tag{26}$$

To identify outliers, we use the IQR in conjunction with a factor, typically 1.5 for outliers or 3 for extreme outliers, which determines the threshold for considering a data point as an outlier. The thresholds for outliers are calculated as follows:

$$\text{Lower Bound} = Q1 - k \times IQR \tag{27}$$

$$\text{Upper Bound} = Q3 + k \times IQR \tag{28}$$

where k is the chosen factor (1.5 or 3).

In Figure 9, we can intuitively observe how this factor impacts the selection of the ranges of data considered outliers and how using 2 different factors can help us understand and differentiate properties, such as the evenness of the distribution.

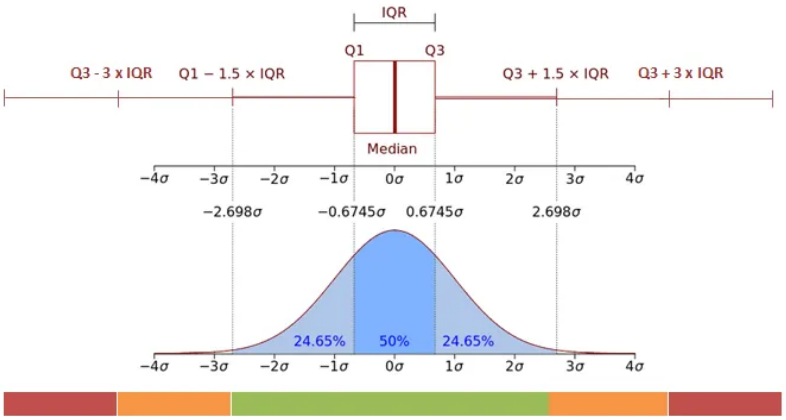


Figure 9: IQR in conjunction with factors of 1.5 and 3 [31]

Upon inspecting the obtained values from testing IQR for both factors, we compare results with the aid of plots, Figure 10. As observable, for the top-outlier dense features both factors give us very comparable if not the same result, reinforcing the poor normal distribution of the data, meaning that either a feature is comprised of mostly the same value (constant) or the extreme opposite of having extreme evenness in that specific feature distribution. This endeavour contributed poorly to our immediate understanding of the distributions at hand and how we might deal with outliers for models, despite that, as we touch on ahead, this information proved very relevant when devising procedures to correlate non-conform production pieces to relevant outliers.

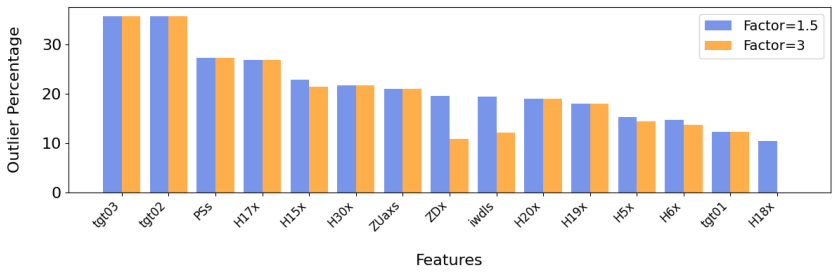


Figure 10: Frequency Plot: Outliers via IQR for 15 most outlier dense features

Other initiatives were taken to better grasp the state of the data distribution for the various features, none contributing particularly to our understanding or that would prove useful for the project's development. Most of the knowledge attained regarding this property was acquired during feature selection and later in model testing.

3.4.3 *Data correlation*

Regarding standard approaches to data exploration and understanding, correlation comes as a great tool. Understanding feature correlation proves very relevant for our context, allowing us to either join features using Principal Component Analysis (PCA) or even remove them to avoid adding additional and unnecessary noise to the models.

With the help of a correlation plot such as the default one built into the Seaborn Python library, we get easy info on how some features might be groupable and are able to highlight and contrast both strong positive and negative correlations between features that could be handy going forward. In Figure 11, we get a glance at our correlation matrix and quite immediately are able to discern high-density zones entailing very similar feature behaviour. For cleanliness's sake, the figure has had its feature names removed so as not to hindrace visualisation.

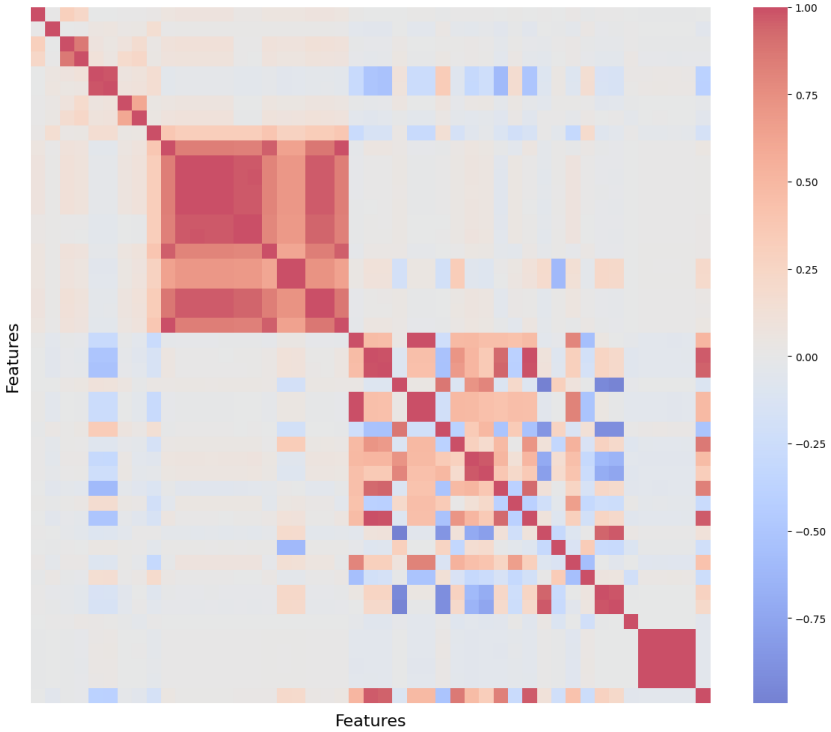


Figure 11: Feature Correlation Plot

With the figure guidance and on inspecting the correlation matrix, we quickly relate higher correlation features to temperature sensors within the same zone of the mold or thermoregulators assigned to stay at the same value, also displaying similar oscillations in

value and therefore having extremely strong correlation values. On the opposite end, we get features like iwdls (Dosage yield) and Zdx (Dosage time) having very high negative correlation, since the lower the dosage time, the higher the dosage yield is.

3.4.4 *Frequency Analysis of Principal Components*

To further explore the temporal structure of our dataset, a periodogram was computed for the first principal component obtained from PCA. This approach provides insight into the dominant frequencies present in the data, helping us distinguish between slow trends and high-frequency variations.

The periodogram essentially decomposes the time series into its frequency components, quantifying how much each frequency contributes to the overall variance. In Figure 12, we visualize the power spectral density on a logarithmic scale to ensure even subtle contributions at higher frequencies are visible.

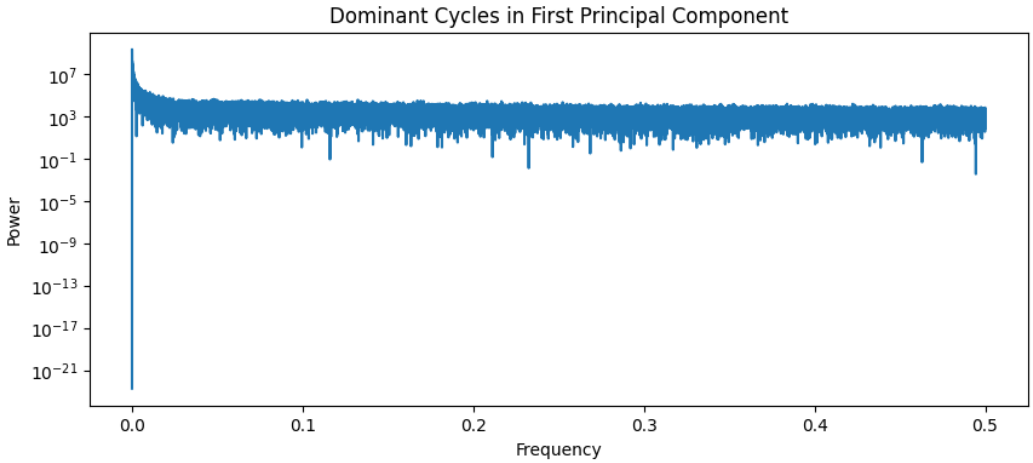


Figure 12: Periodogram of the first principal component

From the figure, two key insights emerge:

1. **Dominance of Low-Frequency Trends:** The strong peak near zero frequency indicates that the first principal component is largely driven by slow-changing patterns. These could correspond to gradual process drifts, environmental influences, or cumulative operational effects over time.

2. **Limited High-Frequency Cycles:** The spectrum at higher frequencies is relatively flat, suggesting no pronounced periodic or cyclic behaviour. High-frequency noise is present, which is expected in real-world industrial sensor data, but it does not dominate the signal.

3.5 DATA TRANSFORMATION

Despite many changes to the data being made during the project, they are usually related to specific processes and are relevant to be brought up on that specific step so as to better substantiate the explanation process. As far as substantial data changes made in these early stages only two situations can be highlighted:

- **Feature Engineering:** With the help of Aspöck it was possible to add a feature relating to the relative Viscosity of the plastic during injection. As far as technical plastic production goes, this is a very relevant feature for the process engineer's understanding and possibly the same could apply to our model.

  The Viscosity is calculated by multiplying the peak injection pressure ($''PVs''$), the filling time ($''ZSx''$), and the intensification ratio of constant 8.67. The intensification ratio is the relation between plastic specific pressure to hydraulic pressure and is specific to the machine build, meaning different machines have different intensification ratios calculated from the contact area of both sides.

$$\text{Relative Viscosity} = \text{"PVs"} \times \text{“ZSx”} \times 8.67 \tag{29}$$

- **Data Scaling:** Functions in the code were built to appropriately scale the data via normalisation or standardisation according to our needs. This ultimately allowed us to test both options extensively and, later on, even go as far as experimenting with normalising subsections of features using the same range to preserve their meaning and relation when all the selected features were meant to represent the same scale of physical property, for example, temperatures or pressures. In the end standardization often performed better than normalization due to not squishing ranges and preserving the properties of the distribution that are key to outlier detection.

3.6 SUMMARY

To briefly overview, this chapter allowed us to gain a solid understanding of the data and, in the process, develop tools, namely functions and graphs that help us extract relevant information over the long term. Not all of the information acquired was immediately recognised as valuable. Still, throughout the project, we frequently revisited the exploration sections of the code to revise and uncover previously unforeseen relationships and properties within the data.

Following our designed objectives, we extensively covered the steps from data collection to transformation. This process highlighted significant information regarding data identity and properties while adequately cleaning and preparing the relevant data for upcoming stages.

With the addition of a new feature and the scaling of the data, we now possess the tools necessary to proceed to the next stage and undertake an extensive feature analysis and selection process.

From the steps taken during this phase, we can highlight the following in relation to changes made to the dataset and their respective impacts.

Chapter	**Process**	**Changes Made**	**Dataset Impact**
Data Cleaning	Missing Values	-	0%
Data Cleaning	Duplicate Features	-1 Feature	-0,38%
Data Cleaning	Removal of non-relevant data	-216 Features	-81,81%
Data Cleaning	Inadequate Data Correction	-4917 Rows	-6,92%
Data Transformation	Feature Engineering	+1 Feature	+2,13%

Table 5: Table of Dataset Changes

The data percentage relative change mentioned concerns the current dataset size according to the previous processes.

4

FEATURE SELECTION

The initial go-to solution devised for identifying non-conform parts was to proceed with either outlier or novelty detection methods. This line of thought was logical, as non-conform parts would deviate from normal production data and, therefore, be identifiable through what we would mathematically describe as outliers or novelties in the data.

Regardless of the various approaches experimented with, this line of reasoning does not ultimately work. Extremely poor results lead us to a few conclusions that force us to adapt our approach to solving this project and fundamentally change our understanding of the data and some of its properties in relation to the production parts.

From the various models tested, which are covered in the models chapters, the lack of positive results had to be derived from one or multiple of the following issues:

- **Usage of Inadequate Models:** This was one of the early possible reasons that motivated a lengthy experimentation process of various models and a whole range of hyper-parameters for each model. Seen as results from the multiple models on significantly different spectrums were equally bad at identifying outliers that would correlate to real non-conform parts and would always drift towards merely numerical outliers. It prompted us to pinpoint the limitation in the data and not the model usage.
- **Poor Selection of Features and Features Noise:** As previously stated, having extreme deviations in one feature values would not directly imply a non-conform piece since the feature itself could have a low or close to inexistent relation to the production piece quality outcome. To that extent, one possible way out would be to improve our feature selection process, diminishing the presence of low-importance features while trying to bring to light features that would, in the end, visibly reflect deviations on low-quality injection.
- **Not Detecteable Correlation:** It is perfectly conceivable that the correlation might not be possible to identify due to it not existing or being extremely low, in our view this could stem from two reasons:
  - **Extremely Refined Process:** Aspöck's production process is not hands-off, quite the opposite is true. The company invests heavily into perfecting and optimising its injection process and in doing so could have refined the process to a point where detected deviations might not be significant compared to other outside factors, such as atmospheric temperature, humidity in the factory or material quality, in that sense, those would be the driving factors for the non-conform parts, and being information that we do not have reliable access to, there would be no way to create a correlation.

– **Material Quality or Other Unknown Factors:** As stated, our having no information on material quality going into the process could be one of the reasons for not correctly detecting outliers since they could be a direct result of faulty materials injected on those specific parts. Other than material quality, other external factors could also be driving forces for the non-conformity of the parts.

Given this standpoint, it was logical for us to take the following steps related to features, acknowledging the need for better selection and devising metrics and methodologies for it.

4.1 INTRODUCTION OF AUXILIARY DATA

One of the key elements that can be attributed to the value of this project lies in the state of its validation data. As it was initially presented, the project aimed at a purely unsupervised solution. Despite that, thinking in advance, the process engineer responsible for the production took upon himself the responsibility of devising a way to acquire data pertaining to the condition and the acceptability of the injected parts to support the project. To that end, a fill-in form was created to allow machine operators to report defective parts from an aesthetic point of view, Figure 13.

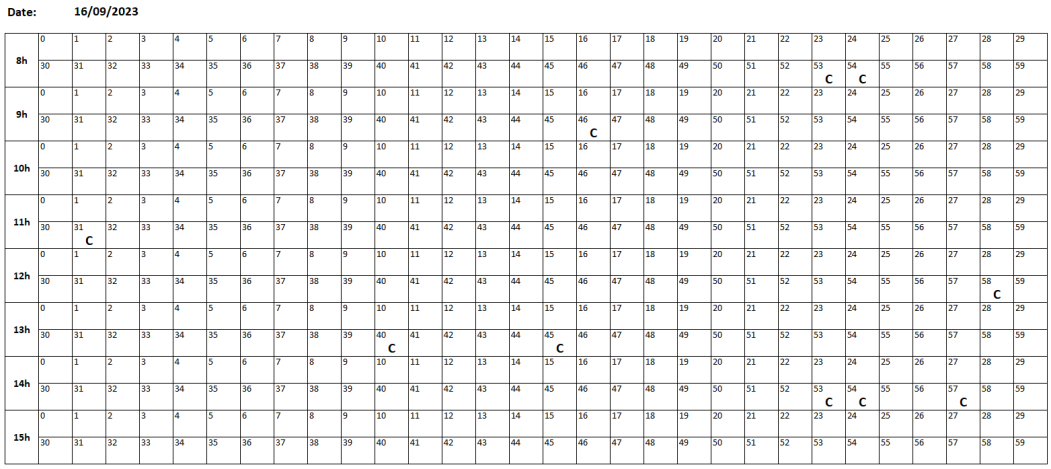


Figure 13: Fill-In form for part compliance and defect report, where the "C" stands for a specific type of visual defect in the plastic part. Other defects were existing but in a significantly smaller frequency hence the study focus on this particular issue.

The form covered the 3 shifts according to the company's labour rotation, each filled by the operator during their respective shift while checking the parts for defects. In Figure 14, we get a closer look at how it is built. The form has 1-minute intervals and covers the full 8 hours of the shift. Each time the operator spots a piece that needs to be left out due to quality issues, he writes an abbreviation of the type of defect encountered and when it was encountered.

One important point to highlight, which highly influences the project going forward, is that, as mentioned, the time registered for the sighting and exclusion of the piece due to its nonconformity is inconsistently distanced from when it was injected. This happens because

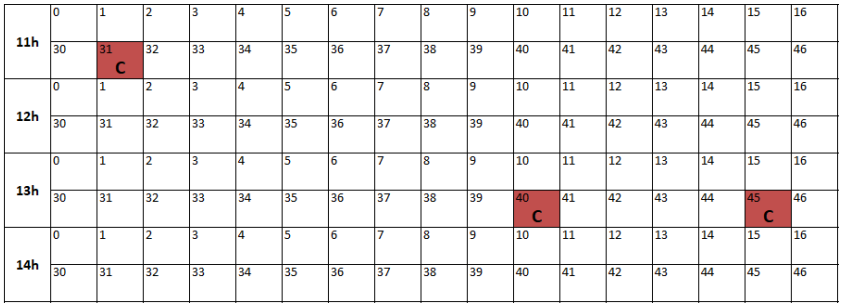


Figure 14: Zoomed in section of the fill-in form

the operator receives multiple parts in the conveyor belt and does not always orderly analyse each while also taking varying times between inspections. With that in mind, the time frame expected between piece injection and defect spotting was considered to be anywhere between 1 and 7 minutes, based on procedure knowledge.

What this briefly represents is that for non-conformity detection, or, in other words, as our objective variable, we consider the highlighted range in search for volatile data and not the detection time alone, Figure 15.

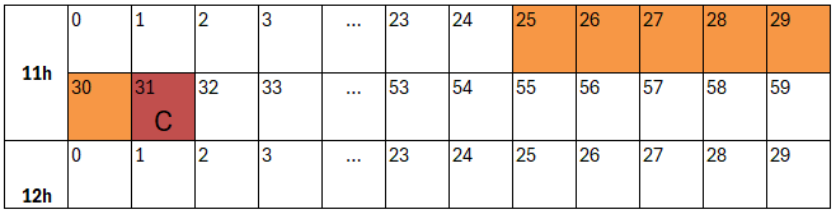


Figure 15: Range of considered volatile data on non-conformity detection

This is a consideration we take forward and does substantially impact many steps and metrics used.

4.1.1 *Enabling the Auxiliary Data*

At its current state, this additional data was of very little use to us. In an effort to extract value from it, 10% of the manual fill-in forms written from the 3 month time span of extracted data were inserted into a CSV format that could then be imported and used to aid us in the project development.

Special attention was taken on selecting the data used, making an intentional effort not to create biases using the same number of forms from each shift while also keeping them decently spaced along the 3 months of data. Some were also selected as extreme data extracts for feature selection and comparison, meaning we had forms with extremely high and low density of defects to better assess how this might relate to feature outliers on the extremes, since extreme scenarios are usually when the correlations are most visible.

The CSV file was composed of only 2 columns of data, the date and time in a combined *"DateTime"* format and the defect count for that specific time frame.

4.1.2 *Visualisation and Interpretability*

To quickly gauge how the added information might help us select features and provide us a solid basis to compare good and bad data, plots were created comparing sub-sets of extremely good and bad data.

Histograms were made for every feature in the dataset, with different colours for each type of data, allowing us to visualise whether it would be possible to segregate the data points and later implement methodologies for that end. In doing so, at first glance, it was possible to immediately distinguish some features where the differences were substantially more observable. In Figure 16 we can see that for the feature *"H2x"*, the distribution of values present is the same for both samples of data. In contrast, there is less overlap for the feature *"Viscosity"*, and bad data appears to be more strongly related to higher values of Viscosity.

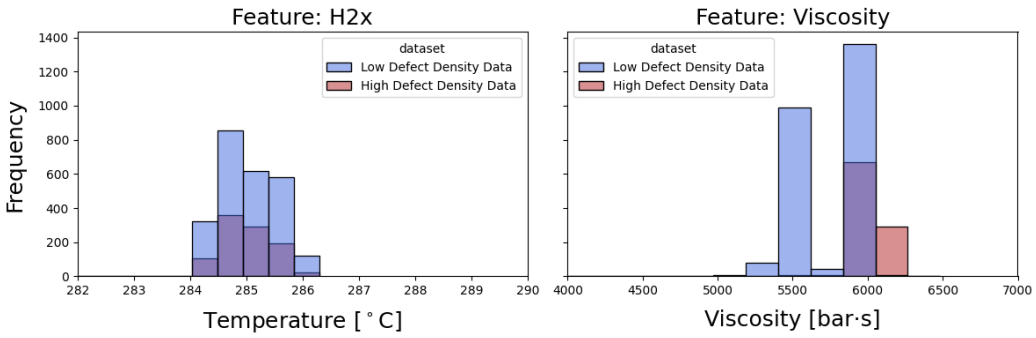


Figure 16: Frequency Plot - Comparing Feature distribution for each type of sample data

This prompted us to look for other Features that might be of higher relevance based on the distinctiveness of the data. In Figure, 17, two Features, *"Ms"* and *"Mm"*, are present where the data distribution difference is the most prevalent. These features respectively represent the peak and medium torque in N · m achieved during the dosage by the conveyor screw.

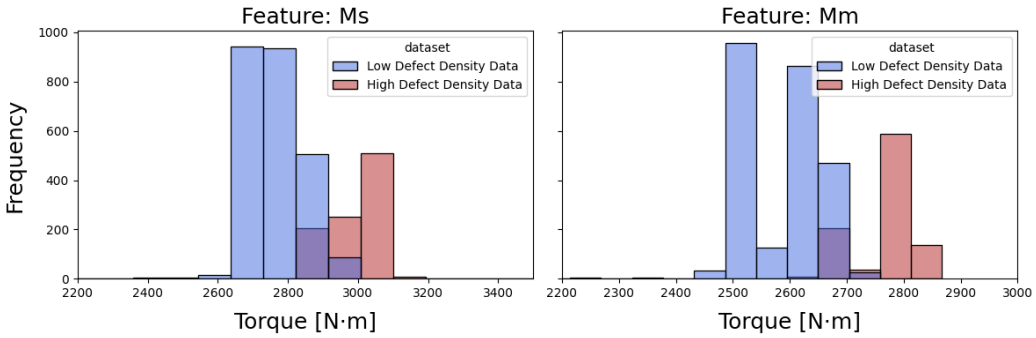


Figure 17: Frequency Plot - Features "Mm" and "Ms"

Visualisations such as Kernel Density Estimate (KDE) plots allow us to grasp the overlapping data and distribution distinctiveness better. In Figure 18, we can see yet another great data comparison for the highlighted features.

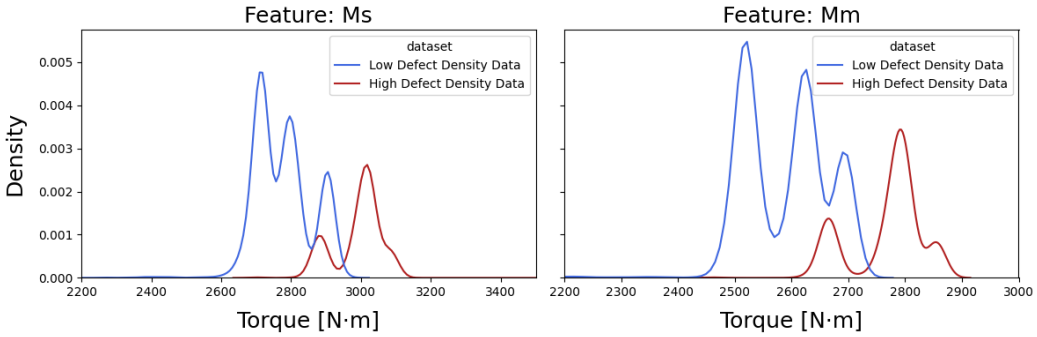


Figure 18: KDE Plot - Features "Mm" and "Ms"

4.2 METHODOLOGIES FOR FEATURE SELECTION

The previous steps provided us with enough insight to proceed with selecting relevant features. The approaches chosen were strongly based on the capability of distinguishing the two data types within the same feature and on how that specific feature would contribute information and variance to the model. This selection was made using the dataset previously achieved through data processing in Chapter 3, where key changes were the removal of inconsistent data due to machine maintenance, removal of 216 features due to various factors detailed in Section 3.3, addition of a new feature and standardization of the dataset.

4.2.1 *Retaining Feature Explainability*

In the premise of selecting and processing features, something that is very important to emphasise is the importance of explainability for this project. More than a functional model, it is of the utmost importance that the insights acquired during this project can be provided as input to the process engineers to bring clarity to the mold injection process, highlighting less intuitive factors and possibly uncovering unknown relations or problems.

With that in mind, most of the steps taken during the project take heavily into consideration information integrity and traceability of the features.

Before selecting a definitive subset or subsets of features, the features are tested for importance in terms of information contribution and relation to the non-conformities.

4.2.2 *PCA: Cumulative Explained Variance*

Cumulative explained variance refers to the amount of information (variance) retained by adding more principal components. As a common metric used for the unsupervised processing of data, it helps us understand the percentage of information that each feature contributes to the variance of the model and the amount of information that can be represented by a reduced number of features via PCA.

Even though we could realistically reduce the features used in the models from 47 to 20 and retain 98.38% of the information, we would not be aware if the information lost was relevant, but also lose all sense of the meaning behind each feature and their importance to the model, therefore having no way to retrace the problem back to the injection process itself. Figure 19 highlights the addition of information retained per increment of Principal Component Feature.

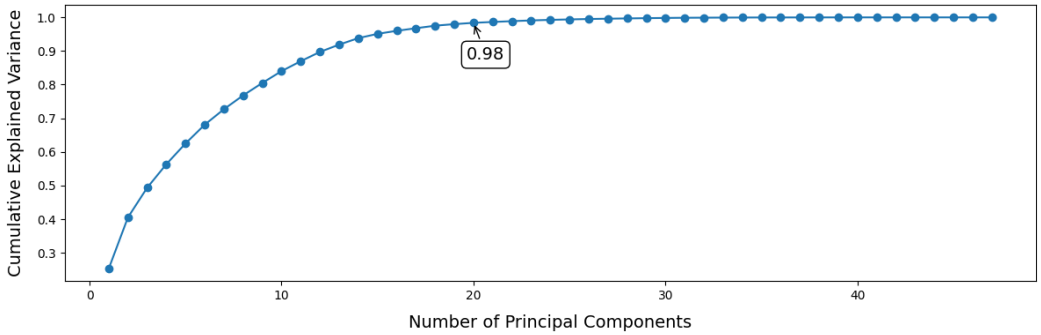


Figure 19: Cumulative Explained Variance by Principal Components

4.2.2.1 *PCA: Property Isolation*

Alternatively, a way was devised to work around the information lost from the meaning of the feature while still being able to use PCA to reduce the number of features inputted into the models.

Since the features used are data from real-world physical properties and many of them, in regards to the injection process, represent the same properties on the same physical scale measured along different zones, we can reduce them while preserving their meaning. In other words, 24 temperature sensors all measuring in Celcius degrees could be reduced to 10 while retaining nearly 100% of the explained variance and from our standpoint, issues or insights related to that set of PCA features would still be traceable to the temperature of the hot runner system as the contributing factor to the model outcome.

In Figure 20 we can visualize this possible solution to feature reduction in which the temperature sensors that are located inside the mold are all gathered under the same subset of PCA components.

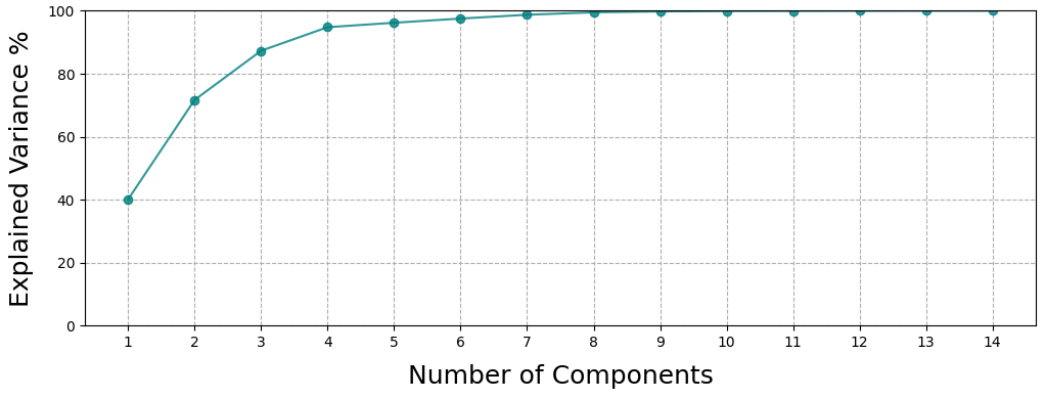


Figure 20: Temperature: Sensors in the Mold

The same approach could then be applied to other groups of sensors, such as the temperature sensors in the injection unit, Figure 21 or the four thermoregulators placed outside to in real-time adjust the temperature of various sections of the mold, Figure 22.

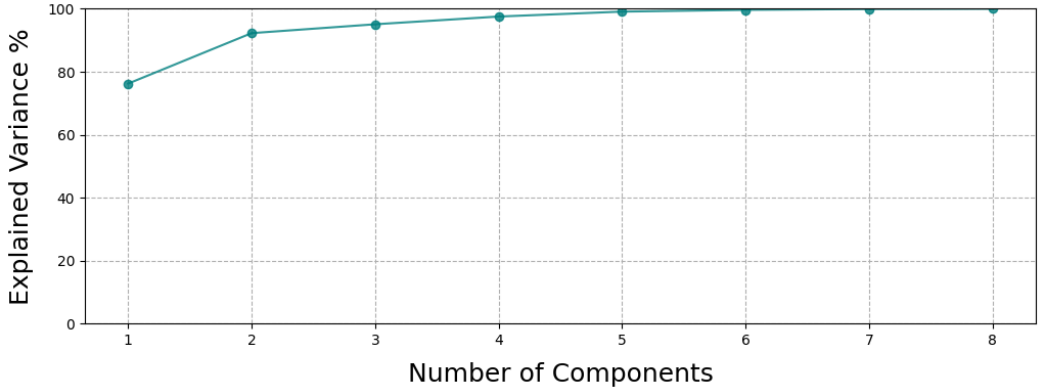


Figure 21: Temperature: Sensors in the Injection Unit

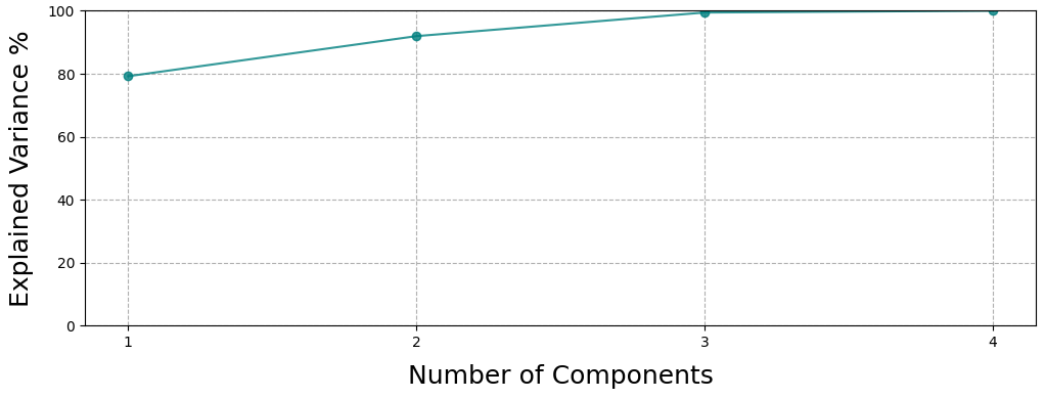


Figure 22: Temperature: Sensors in the Thermoregulators

This approach works great even though there is poor correlation between the temperature features outliers and the non-conform parts, which is discussed later as we delve into feature importance analysis.

4.2.3 *Mann-Whitney Wilcoxon*

The Mann-Whitney test was employed to deterministically validate whether features could be separated by their capacity to identify differences between density of conform and non conform parts in a given data subset. If the difference is distinguishable for the given feature, it would mean that the feature had a noticeable distribution difference for both data extracts and, therefore, is a good selection for the model.

The Mann–Whitney test was employed as a non-parametric statistical tool to assess the discriminative power of individual features. Specifically, the test was applied to compare the distributions of a given feature across data subsets containing varying proportions of conforming and non-conforming parts. A statistically significant result indicates that the feature's distribution differs between the two groups, suggesting that it carries information relevant to distinguishing conforming from non-conforming production outcomes. Features identified in this manner would therefore be considered suitable for inclusion in the feature set used to train the model.

The Mann-Whitney is adaptable, providing a solution for both normal-distributed variables using its mean in the form of the T-test and alternatively using the rank sum for non-normal distributions as the U-test. In this instance, we use the U-test, as we had previously confirmed that every feature had a non-normal distribution.

Nevertheless, the Shapiro-Wilk test was again employed for the data subset should those samples provide different results.

4.2.3.1 *Shapiro-Wilk Test for Normality*

In Section 3.4, the dataset was tested for normality in its entirety for each feature, and it was confirmed that none could be considered a normal distribution for an $\alpha = 0.05$. Despite that, this process was repeated for both data subsets, which were tested feature by feature for their normality for the same significance level of $\alpha = 0.05$.

The results remained the same. Allowing us to proceed and do the U-test for the 47 current features.

4.2.3.2 *U-Test*

The Mann-Whitney U-test, also known as the Wilcoxon rank-sum test, is a non-parametric test used to evaluate whether two independent groups come from the same distribution. [32]

The test assumes that all observations from both groups are independent, and that the response values are at least ordinal, meaning that it is possible to rank them. Under the null hypothesis H_0, the distributions of both populations are assumed to be identical. The alternative hypothesis H_1 posits that the distributions are not identical, allowing us to detect differences in central tendencies without specifying a particular distributional form. [33]

The Mann-Whitney U statistic, U, is calculated by first ranking all observations from both groups together, from smallest to largest. Let $X_1, \ldots, X_{n_1}$ represent the observations in group 1 (with sample size n_1), and $Y_1, \ldots, Y_{n_2}$ represent the observations in group 2 (with sample size n_2). The U statistic can then be computed using the following formulas:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1$$
$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2$$

where R_1 and R_2 are the sums of the ranks for groups 1 and 2, respectively, after ranking all samples from both groups combined such that the smallest observation receives rank 1 and the largest receives rank $n_1 + n_2$.

The test statistic U is the smaller of U_1 and U_2, which allows us to consult significance tables for the U distribution. For larger samples, when $n_1, n_2 > 20$, U can be approximated by the normal distribution, enhancing computational efficiency.

The U-test's probabilistic formulation also allows interpretation in terms of the Area Under the Curve (AUC) in receiver operating characteristic (ROC) analysis, where:

$$\text{AUC} = \frac{U}{n_1 n_2}$$

In this approach, the Mann-Whitney U-test serves as a distribution-free method to identify whether specific features significantly differentiate between high-quality and low-quality data extracts.

The test was done using the library SciPy [34], being the reduction in features from 47 to 39. The 8 features removed were all features relating temperature readings on various parts of the injection process, namely 'H2x', 'H3x', 'H4x', 'H9x', 'H24x', 'H25x', 'H29x', 'H30x'. This subset of feature selection was then saved for model running and comparison of results to other subsets of selected features.

4.2.4 *Pearson Correlation*

Following up on the Mann-Whitney U-test approach, another highly relevant feature selection methodology to consider was the Pearson Correlation Coefficient (PCC). Having previously employed a correlation matrix to explore the relationships between features, this step leverages the matrix as a systematic approach for selecting features based on the similarity of their numerical information contribution. In Figure 11 we previously visualized the current status of the dataset, identifying extreme correlation between sets of features.

The Pearson Correlation Coefficient quantifies the linear relationship between two variables, providing values between -1 (perfect negative correlation) and $+1$ (perfect positive correlation), with 0 indicating no linear relationship. The absolute value of the coefficient is used to assess the strength of the correlation.

Features exhibiting high correlation are likely to provide redundant information to the model, which can inflate its complexity without significantly improving performance. To that end we aim at filtering the Features via a correlation threshold of 0.90.

To attain that result pairwise correlation matrix was computed for all features in the standardized dataset, and a threshold of 0.90 was applied to identify pairs of features with high absolute correlation. For each pair exceeding this threshold, features were flagged for removal.

For each pair of flagged features, the one with the least variance was removed to ensure no two remaining features shared a strong linear relationship, while keeping the one with the highest information spread.

Upon applying this methodology to the dataset, a total of 18 features were identified and removed, Table 6.

Feature	**Max Corr.**	**Feature**	**Max Corr.**
CPv	0.999967	H29x	0.996578
tgt04	0.999725	H27x	0.996144
PVs	0.999486	CPx	0.985282
tgt02	0.998760	PNs	0.983875
H28x	0.996959	H6x	0.981434
H26x	0.996959	Ms	0.962643
H23x	0.996857	EA	0.945125
H30x	0.996578	iwdls	0.922721

Table 6: Dropped features and the highest correlation they exhibited with another feature in the dataset.

This reduction in features from 47 to 29 streamlined the dataset, reducing dimensionality while preserving the integrity of the information available to the model. Figure 23 illustrates the resulting correlation matrix, demonstrating the lower levels of inter-feature correlation after applying the threshold. This is especially noticeable when comparing the two figures and seeing the diminish in similar correlation areas, and instead finding a more random assortment of correlation distributions between features.

4.3 SELECTION OF FEATURE SUBSET

Out of the previously stated methodologies, initially the chosen method was to proceed with methodology discussed in Section 4.2.2.1, reducing the initial pre selection of 47 features to 33. This allowed us to keep all the features relevant to the injection process while reducing features with the same physical property contributions, namely temperature zones, to a fewer number of features, with little compromise on the explained variance of the features sub group.

This approach would ultimately not be enough for the purposes of the project. The drive to test various methodologies and compare results, motivated us to revisit feature

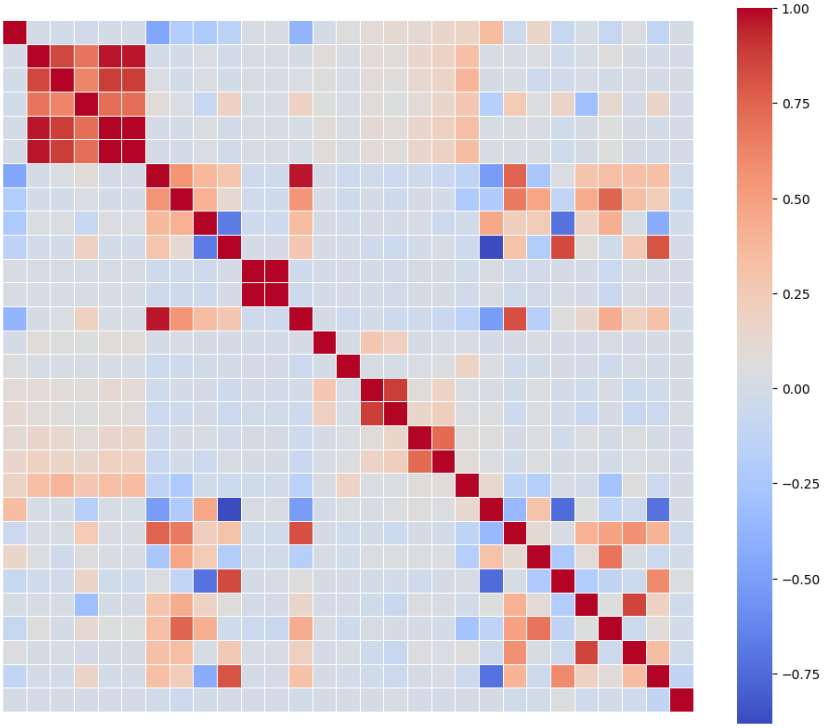


Figure 23: Feature Correlation Plot After Threshold-Based Filtering

selection and save datasets for model training comprised of different subsets of features and features derivations. Not only would this allow us to benchmark models for the dataset, but inversely also benchmark how the same dataset presented under different conditions and dimensionality would compare between models and feature selection methods.

To briefly summarize the subsets used for the sake of model training:

- **Isolated Explained Variance** (Section 4.2.2.1)
  - **Feature Reduction:** 47 to 33 features.
  - **Notes:** All injection process relevant features were kept.
- **Double Feature MM & MS**
  - **Feature Reduction:** 47 to 2 features.
  - **Notes:** As highlighted in Section 4.1.2, this two features in many instances are the best descriptors of good and bad data and more so, are derived from the same physical property. This creates a great opportunity to test linear duo feature model solutions.
- **Pearson Correlation** (Section 4.2.4)
  - **Feature Reduction:** 47 to 29 features.
  - **Notes:** The dataset outcome from feature selection that more deterministically described an ideal feature reduction on information level. Additionally all temperature related features were dropped, this decision is covered in Section 6.3

where we dive in on feature explainability with the aid semi-supervised solutions to understand feature contribution to the model objective variable outcome.

4.4 SUMMARY

This chapter detailed the challenges of selecting meaningful features for non-conformity detection in a high-dimensional, weakly labelled industrial dataset. Initial attempts at outlier and novelty detection highlighted fundamental data limitations and motivated a shift toward more rigorous feature evaluation. Multiple complementary methods were applied, including principal component analysis (PCA) for variance retention, Mann–Whitney U-tests for distributional separation, and Pearson correlation filtering for redundancy reduction. These approaches progressively reduced dimensionality while preserving interpretability, enabling the construction of feature subsets tailored to different modelling strategies.

In addition to presenting the results of each method, the chapter explored the underlying principles, application requirements, and comparative strengths of these techniques. This dual focus on methodology and process understanding ensured that feature selection not only improved model performance but also generated actionable insights into the injection process itself.

5

MODEL DEVELOPMENT AND EVALUATION

This chapter discusses the development and evaluation of three types of approaches: Outlier Detection Models, Clustering Models, and at last Long Short-Term Memory Networks. Each approach has its own merits and limitations, and this chapter provides a detailed account of the methodologies, results, and comparisons.

As previously addressed, the evolving nature of this project resulted in a very thorough initial investment into unsupervised outlier detection and later on a shifting towards Long Short-Term Memory models in order to fill in current limitations and gaps in static model capabilities. This chapter follows that chronological progression.

5.1 OUTLIER & CLUSTERING DETECTION MODELS

Given the underlying context of the project and available data labelling, the initial approach was done using unsupervised outlier detection. From the first iteration of models a lot of improvements were achieved even if in later stages a natural progression was made towards Long Short-Term Memory models. These improvements in model results were supported by minor improvements on the level of hyperparameters tuning, but mainly due to changes made in our feature selection criteria as covered in Chapter 4.

5.1.1 *Overview of Modeling Approach*

On a base layer set up, the approach to detect outliers was based on two key metrics derived from the industrial process knowledge, the rough percentage of outliers in the entire dataset, meaning value of overall non-conform parts known by the process engineers due to the amount of scrapped parts when compared to good parts, and the auxiliary data covered in Section 4.1. To that end, models that could retrieve outliers were experimented with and tuned to retrieve close to the foreseen outlier percentage, promoting us to then analyse and compare if those determined outliers were in-line with real process data.

Quickly this methodology proved insufficient and a few key changes were made:

- **Pipeline**: The code to be ran was built in a pipeline manner to allow for easier experimentation, change of parameters and sequential running of various datasets.
- **Grid search**: Even though GridSearchCV from Sklearn was not a viable option due to the models being unsupervised and there being no metrics for selection,

hyperparameters were inputted into dictionary structures and models were built to run around those combination of parameters saving all the result combinations.

- **Evaluation Metrics**: Similar to the Grid Search issue, unsupervised models have no inherent direct metric to the objective class that would allow for the validation and later tuning of hyperparameters. To this extent functions were defined to calculate precision, recall and F1-score based on the auxiliary data input and ran side by side with the models to automatically and systemically output grounded metrics that allow us to quantify model performance when matched with expected results from our non-conform part production 2.

Listing 2: Custom Evaluation Metrics

```
def calculate_f1_score(tp, fp, fn):
    if tp + fp == 0:
        precision = 0
    else:
        precision = tp / (tp + fp)

    recall = tp / (tp + fn)

    if precision + recall == 0:
        return 0  # to handle cases where precision + recall is 0
    else:
        f1_score = 2 * (precision * recall) / (precision + recall)
        return f1_score

def calculate_precision(tp, fp):
    if tp + fp == 0:
        return 0
    else:
        return tp / (tp + fp)

def calculate_recall(tp, fn):
    if tp + fn == 0:
        return 0
    else:
        return tp / (tp + fn)

def calculate_f1_inputs(real_outliers_indices, range_outliers_indices,
↪  sample_outlier_indices, detected_outliers_indices):
    true_positives = len(set(range_outliers_indices) & set(detected_outliers_indices))
    false_positives = len((set(detected_outliers_indices) & set(sample_outlier_indices)) -
    ↪  set(range_outliers_indices))
    false_negatives = len(set(real_outliers_indices) - (set(detected_outliers_indices) &
    ↪  set(sample_outlier_indices)))

    return true_positives, false_positives, false_negatives
```

- **Visualization**: A visualization tool was created to match subsets of data on a grid, allowing for the visual understating of the outliers highlighted by the models overlapped with the real outliers and outliers zones previously defined in Section 4.1. In Figure 24 overlapping can be observed in black, real outliers in red and model outliers in green.

From this point onwards an extensive pipeline grid search was executed for all relevant models from an outlier detection standpoint. For models which support multithreading, parameter *n_jobs* was used with value -1 to allow for parallel processing, further accelerating the model calculations.

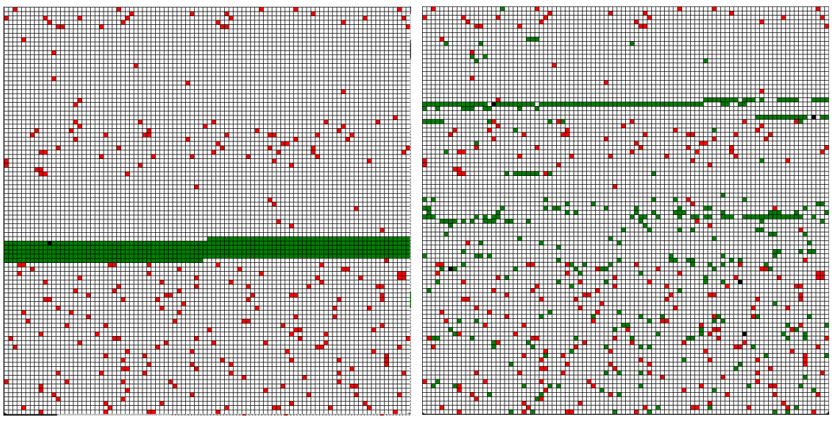


Figure 24: Visualization of a dataset subset highlighting outlier detection results. Overlapping points are shown in black, true outliers are marked in red, and model-detected outliers are indicated in green.

5.1.2 *Parameters and results*

This section overviews the sets of parameters and respective results for each model, detailing how the model was built and aspects like the outlier threshold used for each model.

5.1.2.1 *K-Means Clustering*

For the K-Means model, the following hyperparameter grid was defined for exhaustive search:

- **Number of Clusters**: {2, 3, 4, 5, 6, 7, 8, 9, 10}
- **Initialization Method**: {`k-means++`, `random`}
- **Number of Initializations**: {10, 20, 30, 40, 50}
- **Maximum Iterations**: {300, 500, 1000, 2000, 5000, 10000}

Total combinations tested: $9 \times 2 \times 5 \times 6 = 540$

Outlier Threshold Used: The top 3% of points, based on the model's outlier score distribution, were flagged as outliers. That is, all points x_i for which $S_i > \tau$, with $\tau = \text{Percentile}(S, 97)$, were included.

Results showed best alignment with our ground-truth outlier zones when using between 9 and 10 clusters and the `k-means++` initialization method.

Despite this, the overall performance of K-Means in isolating relevant outliers was limited, yielding a maximum F1-score of 0.0188. This suggests that the clustering approach, while marginally informative, was not sufficiently aligned with the non-conformity patterns present in the process data. Table 7 summarizes the best performing parameter sets.

Model Configuration	F1-Score	Precision	Recall	Matching Outliers
K-Means++ & 9 to 10 clusters	0.0188	0.0149	0.0254	7

Table 7: Selected K-Means configurations with corresponding evaluation metrics

5.1.2.2 *Local Outlier Factor (LOF)*

For the LOF model, the following hyperparameter grid was defined for exhaustive search:

- **Number of Neighbors**: {5, 10, 15, 20, 25, 30, 40, 50}
- **Contamination**: {0.01, 0.02, 0.03, 0.05}
- **Algorithm**: {`auto`, `ball_tree`, `kd_tree`, `brute`}
- **Metric**: {`euclidean`, `manhattan`, `minkowski`}

Total combinations tested: $8 \times 4 \times 4 \times 3 = 384$

Outlier Threshold Used: The top 3% of points, based on the model's outlier score distribution, were flagged as outliers. That is, all points x_i for which $S_i > \tau$, with $\tau = \text{Percentile}(S, 97)$, were included.

Results showed that the best-performing configuration used the `manhattan` distance metric, 5 neighbours and 3 algorithm's: "auto", "kd_tree", "ball_tree", were equally present. This configuration consistently identified the same 90 matching outliers across contamination levels, achieving the highest F1-score of 0.2826.

These results indicate that LOF is substantially more effective at capturing the underlying non-conformities in the data compared to clustering-based methods, particularly when leveraging local distance-based density estimation. Table 8 summarizes the best-performing configuration.

Model Configuration	F1-Score	Precision	Recall	Matching Outliers
`5 neighbors` & `auto` & `manhattan`	**0.2826**	0.3093	0.2601	90

Table 8: Best-performing LOF configuration and corresponding evaluation metrics

5.1.2.3 *Isolation Forest*

For the Isolation Forest model, the following hyperparameter grid was defined for exhaustive search:

- **Number of Estimators**: {10, 20, 30, 50, 100, 200, 300, 400, 500, 1000}
- **Contamination**: {0.01, 0.02, 0.03, 0.05}

Total combinations tested: $10 \times 4 = 40$

Outlier Threshold Used: The top 3% of points, based on the model's outlier score distribution, were flagged as outliers. That is, all points x_i for which $S_i > \tau$, with $\tau = \text{Percentile}(S, 97)$, were included.

Results showed the strongest alignment with known outlier zones when using a low number of estimators (10) and a contamination rate of 0.01, that even though contradictory to our knowledge of the dataset would align the most number of outliers to real outliers rather than a contamination of 0.03. This configuration yielded the highest F1-score of 0.422, identifying 165 matching outliers.

The performance of Isolation Forest was at first glance notably better than both clustering and local density-based approaches in this context. However when resorting to visualization aid, we concluded than the way it was achieving those results was far from ideal, making use of the fact that the dataset is extremely unbalanced and in the process creating a substantial number of false positives. Table 9 summarizes the best-performing configuration.

Model Configuration	F1-Score	Precision	Recall	Matching Outliers
`10 estimators` & `0.01 contam.`	**0.4220**	0.4496	0.3976	165

Table 9: Best-performing Isolation Forest configuration and corresponding evaluation metrics

5.1.2.4 *OPTICS*

For the OPTICS model, the following hyperparameter grid was defined:

- **Minimum Samples**: {5, 10, 20, 50}
- **Distance Metric**: {`minkowski`, `euclidean`, `manhattan`}
- **Steepness Threshold**: {0.05, 0.1, 0.2}

Total combinations tested: $4 \times 3 \times 3 = 36$

Outlier Threshold Used: Outliers were selected based on reachability distances. For valid clusterings with at least two clusters, the reachability distances were sorted in descending order. The top 3% of samples (based on these values) were flagged as outliers:

$$\text{Outliers} = \{x_i \mid \text{Reachability}_i > \tau\}, \quad \tau = \text{Top } 3\%$$

This approach leverages the insight that points with the largest reachability distances are likely to be located in sparse regions of the data space and therefore represent structural outliers.

Results indicated that OPTICS performed exceptionlly well when compared to previous methods in this study. With the `manhattan` metric, `min_samples` set to 50, and `xi` in the

range of 0.05 to 0.2, it consistently identified 411 matching outliers and yielded an F1-score of 0.5362. This demonstrates a high degree of alignment with annotated outlier regions, outperforming both clustering and density-based methods in this context. This results however good, were still far from ideal for a production scenario.

Model Configuration	F1-Score	Precision	Recall	Matching Outliers
`min_samples = 50` & `manhattan`	**0.5362**	0.4639	0.6352	411

Table 10: Best-performing OPTICS configuration and corresponding evaluation metrics

5.1.2.5 *Mean Shift Clustering*

For the Mean Shift model, the following bandwidth values were tested, corresponding to the kernel size used during clustering:

- **Bandwidth**: {0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0}

Total combinations tested: 11

Outlier Threshold Used: Outlier scores were defined as the negative Euclidean distance between each sample and the centroid of its assigned cluster:

$$S_i = -\|x_i - c_j\|_2 \quad \text{for } x_i \in \text{cluster } j$$

Outliers were then selected as the top 3% of samples with the lowest scores (i.e., farthest from their assigned cluster centres).

Surprisingly, the model exhibited highly consistent results across all bandwidth values tested. In each case, 52 known outliers were correctly identified, yielding a stable F1-score of 0.2185. Although clustering boundaries were non-disruptive and broadly consistent, the effectiveness of the Mean Shift model in detecting non-conforming patterns remained limited and significantly behind other options.

Model Configuration	F1-Score	Precision	Recall	Matching Outliers
`all bandwidths tested`	0.2185	0.3537	0.1581	52

Table 11: Mean Shift performance across different bandwidth values

5.1.2.6 *One-Class Support Vector Machine (OC-SVM)*

The One-Class SVM model was subjected to an extensive hyperparameter search, reflecting the model's flexibility across various kernels and decision boundary configurations:

- **Kernel Function**: {`rbf`, `sigmoid`, `poly`, `linear`}

- **Regularization Parameter**: {0.01, 0.02, 0.05, 0.1, 0.2, 0.5}
- **Kernel Coefficient**: {`scale`, `auto`, 0.001, 0.01, 0.1, 1}
- **Polynomial Degree**: {2, 3, 4, 5} (only applicable for `poly` kernel)
- **Independent Term in Kernel Function**: {0.0, 0.1, 0.5, 1.0}
- **Maximum Iterations**: {-1, 500, 1000, 5000}

Total combinations tested: +2000 but dependent on kernel

Outlier Threshold Used: Anomaly scores were computed as the negative log density estimates:

$$S_i = -f(x_i) = -\texttt{model.score\_samples}(x_i)$$

Outliers were defined as the bottom 3% of data points with the lowest scores, i.e., those for which $S_i \leqslant \tau$, with $\tau = \text{Percentile}(S, 3)$.

The most performant configuration involved the `rbf` kernel with $\gamma = 0.01$ and $\nu = 0.1$, while having no limited max iteration number, leading to a maximum F1-score of 0.3345.

Model Configuration	F1-Score	Precision	Recall	Matching Outliers
RBF & $\gamma = 0.01$ & $\nu = 0.1$	0.3345	0.4439	0.2684	95

Table 12: Best One-Class SVM configuration and associated detection metrics

5.1.3 *Conclusion*

Throughout the iterative experimentation with classical outlier detection and clustering models, we observed a progressive improvement in performance. These gains were primarily driven by systematic hyperparameter optimization, enhanced data preprocessing, and refined evaluation methodologies. Initial models, while indicative, lacked robustness and demonstrated limited capacity to generalize across varying patterns in the dataset. Through successive tuning and contextual re-evaluation, we achieved more consistent results, with some methods such as OPTICS approaching reasonable levels of precision and recall.

Despite these advancements, none of the evaluated methods achieved performance metrics sufficient for reliable deployment in a production environment. Several intrinsic limitations emerged. First, the models struggled to consistently capture the non-stationary and evolving nature of anomalies within the injection moulding process. Second, many of the algorithms were sensitive to initial parameters and required static assumptions (e.g., distance metrics, density thresholds, contamination levels), which inherently limited their ability to generalize across all data regimes. Additionally, the time-dependent nature of the anomalies posed a challenge, as these models are not inherently temporal and thus fail to leverage sequential dependencies.

Given the complexity of the data and the operational requirements of real-world deployment—where adaptability and temporal awareness are critical, a more dynamic and learning-based solution was deemed necessary. This motivated the transition to a sequential deep learning approach using Long Short-Term Memory networks, which is explored in detail in the next section.

5.2 LSTM

Given the progressive limitations encountered with unsupervised outlier and clustering models, particularly in their ability to generalize across varying production states, a more dynamic and temporally-aware solution was required. LSTM networks emerged as a natural next step, offering the ability to model sequential dependencies inherent in time-series data. Hence overcoming some of the limitations previously encountered.

5.2.1 *Overview of Modeling Approach*

The set up for LSTM began with minor processing of the labels. The labels had previously been set up with numbers 0 to 3, in order to ease the processing and separation of the various combinations of data as follows:

- **Label "0"**: Time instance not covered in manual defect sheet.
- **Label "1"**: No recorded defect.
- **Label "2"**: "Defective Zone", meaning range of 7 minutes before spotted defective part.
- **Label "3"**: Defective part spotted.

Since this approach would need a supervised dataset, instances of label 0 were removed leaving 8932 instance of data out of the previous 66108. Then label "1" was mapped to "0" and labels "2" and "3" mapped to "1" signifying a defective prone zone, directly covering time ranges where the data read by sensors would relate to the defective part observed within the next 7 minutes.

From this point onwards we set up what and how we want to predict. Given LSTM's operational structure, we use a **sliding window approach** to model temporal dependencies in the sensor data. The essential idea is to use a fixed-length sequence of past observations, the "window", as input, and train the model to predict whether a defect-prone region it to occur shortly after that window.

The setup for each training sample is as follows:

- **Input:** A sequence of past data points, in our scenario 50 consecutive sensor readings. This sequence acts as a snapshot of recent behaviour, and should be short enough to

be relevant for the next time step, but long enough to capture reasonable sample of data.

- **Output label:** A binary value indicating whether the next time step or range of time steps contains a labelled defect zone.

This results in a binary classification problem, that can be approached in two different ways for our scenario:

- **1 - Sequence to time step** – We use a sequence of N past points to predict if the next time step is either a defect or not.
- **2 - Sequence to *C* time steps** – We use a sequence of N past points to predict if within the next C time steps there's a defect.

To construct the dataset, we employed a sliding window mechanism, which transforms the time series into overlapping sequences. Each sequence consists of a fixed-length context window from the past, followed by a target derived from the immediately following future values. This technique ensures that temporal ordering is preserved and that the model learns from the evolving system dynamics.

Let $X = \{x_1, x_2, \ldots, x_N\}$ be the full sequence of sensor readings, and let $L = \{l_1, l_2, \ldots, l_N\}$ be the corresponding label sequence. We define:

- A fixed input **window length** T,
- A fixed **prediction horizon** k.

For each valid index t such that $1 \leqslant t \leqslant N - T - k + 1$, we construct:

- An input sequence $X_t = \{x_t, x_{t+1}, \ldots, x_{t+T-1}\}$ of length T,
- A target label y_t defined as:

$$y_t = \begin{cases} 1, & \text{if any of } \{l_{t+T}, \ldots, l_{t+T+k-1}\} \text{ is defective} \\ 0, & \text{otherwise} \end{cases}$$

This process is repeated for each t, shifting the input window forward by one time step on each iteration. As a result, the model is trained on a large number of overlapping sequences, each representing a different temporal context within the data.

The intuition behind this approach is to simulate how a real-time monitoring system would operate: at every moment, the system observes the most recent T entries and uses that context to predict whether a defect is likely to occur within the upcoming k time steps. This design maximizes temporal resolution while ensuring predictive alignment with the real-world manufacturing process.

5.2.2 *LSTM Model Architecture*

The model architecture employed throughout this study consists of a single-layer LSTM network followed by a dropout layer and a final dense output layer. The model is designed for binary classification tasks over time-series data, predicting either the next state or the presence of a defect in a future window.

- **Input layer:** Sequential input of shape (50, *n_features*), where 50 denotes the number of time steps and *n_features* varies depending on the feature selection method used.
- **LSTM layer:** One LSTM layer with 32 units and `tanh` activation.
- **Dropout layer:** Dropout rate of 0.3 to mitigate over-fitting.
- **Output layer:** Dense layer with 1 neuron and `sigmoid` activation to yield a probability score for the binary classification task.

The model was compiled with the Adam optimizer (`learning_rate=0.001`) and the binary cross-entropy loss function. In addition to accuracy, the training process monitored precision, recall, and area under the ROC curve (AUC), which are critical metrics for evaluating performance on imbalanced datasets.

5.2.2.1 *Custom Evaluation Metric: F1-Score*

Given the context of the classification problem and trying to optimize overall assertiveness over accuracy, the F1-score was prioritized as the main evaluation metric. A custom F1-score metric was implemented within the TensorFlow Keras framework to complement the built-in accuracy, precision, recall, and AUC metrics. The metric computes the harmonic mean of precision and recall over the validation and test sets to more accurately reflect anomaly detection performance.

5.2.3 *Training Procedure*

All models were trained using the same procedure to ensure consistency across experiments. Training was conducted over a maximum of 50 epochs, using a batch size of 32 and the Adam optimizer with a learning rate of 0.001. A validation split of 20% was applied to the training data in order to monitor generalization performance during training.

To mitigate over-fitting, early stopping was employed with a patience of 5 epochs. This means that training was halted if the validation loss did not improve for five consecutive epochs, and the best-performing weights (in terms of validation loss) were restored at the end of training.

- **Maximum epochs:** 50

- **Batch size:** 32
- **Validation split:** 0.2
- **Optimizer:** Adam (learning rate = 0.001)
- **Early stopping:** Patience = 5, with best weight restoration

Additionally, since the defective instances are relatively sparse compared to normal data, appropriate class weighting was applied during training to compensate for class imbalance. The class weights were inversely proportional to class frequencies, ensuring that the minority class (defective regions) contributed equally to the loss function despite being under-represented. This was achieved using library *sklearn* [35].

5.2.4 *Experimental Design*

Each model configuration was evaluated under six experimental conditions:

- **Three feature selection variants, Section 4.3:**
  - Isolated Explained Variance (33 Features).
  - Double Torque Features (2 Features).
  - Pearson Correlation (14 Features).
- **Two prediction scenarios:**
  - Predicting the label of the next time step based on a 50-step input window.
  - Predicting whether a defect appears in the next 7 steps based on a 50-step input window.

This results in a total of six distinct model runs. Each was trained with the same model architecture but on differently structured data to assess the influence of feature selection and prediction scope.

5.2.5 *Results*

Double Torque (2 Features)

Table 13: Double Torque features (2F) predicting next time step

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.96	0.96	0.96	1275
Anomaly (1)	0.91	0.91	0.91	512
Accuracy			0.95	1787
Macro Avg	0.94	0.94	0.94	1787
Weighted Avg	0.95	0.95	0.95	1787

Using only two torque-based features, the model achieves remarkably strong classification performance for predicting the immediate next cycle, with macro-averaged metrics around 0.94 and a high recall of 0.91 for the anomaly class. The strong discriminative capability suggests that torque signals alone are highly indicative of immediate anomalies, likely due to their direct relationship with mechanical resistance during the injection process. The area under the ROC curve (AUC) confirms this, with a near-optimal value of 0.974 reflecting clear separation between normal and defective cases in this short temporal horizon.

Table 14: Double Torque features (2F) predicting defect within next 7 steps

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.84	0.88	0.86	1055
Anomaly (1)	0.81	0.76	0.79	731
Accuracy			0.83	1786
Macro Avg	0.83	0.82	0.82	1786
Weighted Avg	0.83	0.83	0.83	1786

When the prediction scope extends to potential failures over the next seven time steps, performance degrades noticeably. The anomaly recall drops to 0.76, indicating more missed detections, while precision remains moderately high. The AUC remains relatively high, however, highlighting that although the model struggles with optimal thresholding over extended horizons, it retains meaningful ranking ability.

Pearson Correlation (14 Features)

Table 15: Pearson correlation features (14F) predicting next time step

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.96	0.96	0.96	1272
Anomaly (1)	0.90	0.90	0.90	505
Accuracy			0.94	1777
Macro Avg	0.93	0.93	0.93	1777
Weighted Avg	0.94	0.94	0.94	1777

Using the feature selection attained via Peason Correlation proves to slightly underperform when compared to the two-feature configuration in terms of F1-Score, though only marginally. This suggests redundancy in some of the added features or a limited contribution to the immediate classification task. The high AUC value of 0.96 supports this, confirming strong model confidence in distinguishing classes.

Table 16: Pearson correlation features (14F) predicting defect within next 7 steps

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.82	0.92	0.86	1053
Anomaly (1)	0.85	0.70	0.77	723
Accuracy			0.83	1776
Macro Avg	0.84	0.81	0.82	1776
Weighted Avg	0.83	0.83	0.83	1776

In the future-window prediction scenario, performance shifts notably, once again underperforming when compared to the set up for predicting the next immediate cycle. While still proving to achieve considerable results it pales in comparison to the alternative approach.

Explained Variance (33 Features)

Table 17: Explained variance features (33F) predicting next time step

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.96	0.96	0.96	1275
Anomaly (1)	0.91	0.91	0.91	512
Accuracy			0.95	1787
Macro Avg	0.94	0.94	0.94	1787
Weighted Avg	0.95	0.95	0.95	1787

The most comprehensive feature set, derived from explained variance across the full signal space, matches the best F1 performance in immediate predictions, being slightly better than two feature composition on the third decimal place. This result reflects the benefit of incorporating multivariate process variation, but indicates that for this scenario, a similar result could be achieved while preserving a higher degree of feature explainability. The AUC of 0.975 places in the middle between the two previous configurations of features.

Table 18: Explained variance features (33F) predicting defect within next 7 steps

Class	Precision	Recall	F1-Score	Support
Normal (0)	0.81	0.93	0.86	1055
Anomaly (1)	0.87	0.69	0.77	731
Accuracy			0.83	1786
Macro Avg	0.84	0.81	0.82	1786
Weighted Avg	0.83	0.83	0.82	1786

Much like previous attempts, the future-window approach falls short of the alternative which can be attribute to various factors.

Across all models, the F1-score on the validation set was continuously monitored to assess learning progress and ensure robust performance. As illustrated in Figure 25, a representative model exhibited steady F1-score improvement over successive epochs. Initial scores at epoch 0 ranged between approximately 0.6 and 0.8, gradually rising to 0.9–0.95 by epochs 5 to 12, after which performance plateaued. Early stopping with a patience of 5 was employed, typically halting training between epochs 10 and 20 and restoring the model to its best performing state. This consistent upward trend confirmed that the models were effectively learning from the data without signs of overfitting.

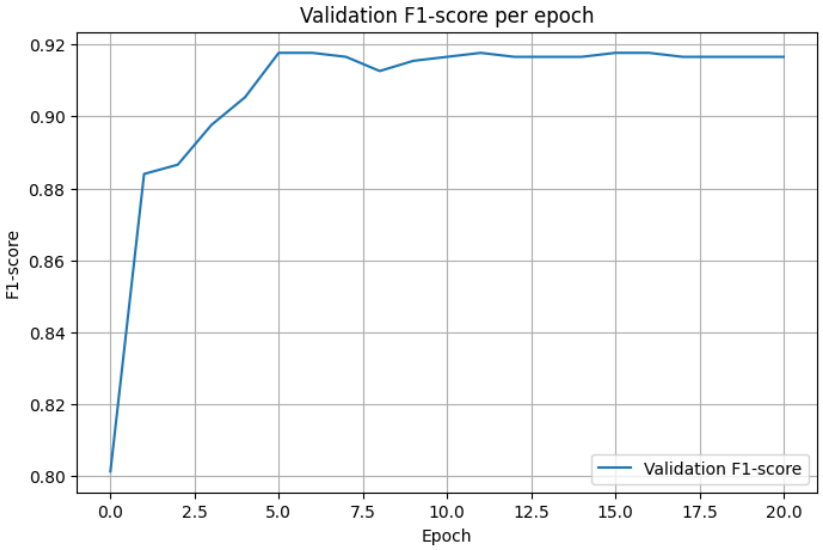


Figure 25: Validation F1-score growth over training epochs.

5.2.6 *Conclusion*

The results reveal several important insights. First, the prediction of immediate next-cycle anomalies consistently yields stronger performance than the prediction of defects within a broader 7-cycle future window. This likely reflects the overlapping of good and poor data within the cycles leading to the 7-cycle range, meaning that while a non-conform part could have 3 cycles of poor data leading up to it aiding in predicting the next cycle, if that non conform part is in the near end of the 7-cycle range, the cycle in short memory used for prediction are then 4 to N cycles away from poor data, hence contradicting the odds for predictive of failure. The drop in recall and F1-score for the anomaly class under extended horizons further confirms the added complexity of the forecasting task, that does not contribute towards a better result.

Table 19 consolidates the macro-averaged F1-scores and corresponding AUC values for all combinations of feature configurations and prediction horizons.

Table 19: Summary of performance across feature configurations and prediction horizons

Feature Set	**Features**	**Horizon**	**F1-Score**	**AUC (approx.)**
Double Torque	2	Next Step	0.94	0.97
Double Torque	2	Future 7 Steps	0.82	0.91
Pearson Correlation	14	Next Step	0.93	0.96
Pearson Correlation	14	Future 7 Steps	0.82	0.89
Explained Variance	33	Next Step	0.94	0.98
Explained Variance	33	Future 7 Steps	0.82	0.90

While high-dimensional representations derived from explained variance (33 features) offer marginal improvements in classification metrics, they do not decisively outperform the minimalistic double torque feature set. In fact, both configurations achieve equivalent macro-averaged F1-scores (0.94) in the next-step prediction task, with the torque-only approach exhibiting greater interpretability and simplicity. This suggests that torque signals encapsulate a significant portion of the predictive information necessary for defect detection, likely due to their direct physical linkage to mechanical properties going into this type of non-conformities.

The feature set derived from Pearson correlation (14 features) performs slightly worse than both extremes in both prediction horizons, potentially due to residual redundancy or noise introduced through selection based on pairwise linear dependencies rather than supervised relevance.

Interestingly, AUC values—indicative of ranking ability independent of threshold—remain high across all configurations, especially for the next-step predictions. This underscores that although extended horizon models face reduced classification accuracy at optimal thresholds, they still preserve useful signal for downstream use in prioritization or risk

ranking frameworks. Being only then discarded due to the significantly better performing alternatives experimented with in this project.

Ultimately, these findings argue in favour of using compact, interpretable features for near-term predictive maintenance applications in injection moulding processes. More complex feature sets may be reserved for contexts where interpretability is secondary to marginal gains, or where extended horizons are unavoidable and require richer temporal representation. This consensus could however change to any minor alteration in the production process, changes to material entry conditions would undoubtedly impact the importance of this 2 features independently of it being positively or negatively. Furthermore degrees of feature importance can quickly swift from one type of part non-conformity to another, being important to bear in mind that this project aimed at some specific types of non-conformities highlighted by the operators.

Ultimately LSTM, as previously speculated significantly exceeds standard models results with substantially minimal investment of parametrisation and training other than the need for labelling which is needed as a base for supervised solutions.

5.3 SUMMARY

This chapter detailed the progressive development of modelling approaches, beginning with unsupervised outlier detection and clustering techniques and culminating in the adoption of sequential deep learning. Initial efforts with classical models such as K-Means, LOF, Isolation Forest, OPTICS, Mean Shift and One-Class SVM showed gradual improvements through systematic hyperparameter tuning and feature selection. However, their static assumptions and lack of temporal awareness limited their ability to capture the evolving nature of anomalies in the injection moulding process.

These shortcomings motivated a shift towards Long Short-Term Memory (LSTM) networks, which naturally incorporate sequential dependencies. Using a sliding-window formulation and carefully balanced class weighting, LSTM models delivered a step-change in predictive capability.

These insights set the stage for the next chapter, which explores the dataset's explainability and demonstrates how feature importance supports the models' results.

6

FEATURE EXPLAINABILITY

One critical aspect of the applicability of the project pertained from its ability to bring clarity to the injection process on how process parameters might affect the production pieces outcome. As previously addressed in Section 4.2.2, we stirred from using methodologies that would compromised the traceability of the features origin, to that end, supervised solutions of feature importance were employed to evaluate how each feature contributed to non-conform pieces.

6.1 MUTUAL INFORMATION REGRESSION

Mutual Information (MI) can be used for evaluating the dependencies between random variables. The MI of two variables, let say X and Y, is the amount of information obtained from X in the presence of Y. In time series prediction problem, if Y is the output and X is a subset of input variables, the MI between X and Y is particularly the criterion for measuring the dependence of Y on X. Thus, the inputs subset X giving MI should be selected to predict the output Y [36].

Formally, it measures the reduction in uncertainty about one variable given knowledge of another. Specifically, for two random variables X and Y, the mutual information $H(X;Y)$ is defined as:

$$H(X;Y) = H(X) + H(Y|X)$$

where $H(X)$ is the entropy of X, representing the total uncertainty or variability in X, and $H(X|Y)$ is the conditional entropy of X given Y, which reflects the uncertainty in X that remains after observing Y. The difference between these quantities gives the mutual information, which intuitively represents how much knowledge of Y reduces the uncertainty of X, or equivalently, how much information X and Y share. This is called the Chain Rule for Entropy [37].

One of the critical properties of mutual information is that it is non-negative and symmetric, meaning $I(X;Y) = I(Y;X)$, since the mutual dependence between the two variables is bidirectional. Moreover, the mutual information is zero if and only if X and Y are completely independent, as independence implies no shared information between the variables.

By leveraging mutual information in regression, we can effectively quantify how much information the independent variables contribute toward predicting the dependent variable.

Allowing us to, without the aid of complex models, and in a very deterministic and simplistic approach, evaluate correlations that directly relate the impact of each feature. This method is applicable to both linear and non-linear problems, but it shines most in high-dimensional problems, where it usually becomes more difficult to identify complex relations via specific regression solutions.

6.1.1 *Results and Visualisation*

To effectively communicate the contribution scale among the features, the MI scores were ranked and visualised using a bar plot. In Figure 26 we can see the overview of the top five features when ranked by their MI values. The plot illustrates the relative importance of each feature, highlighting those that significantly influence the outcome. This approach allows for a clear comparison and provides insights into the most impactful aspects of the injection process.

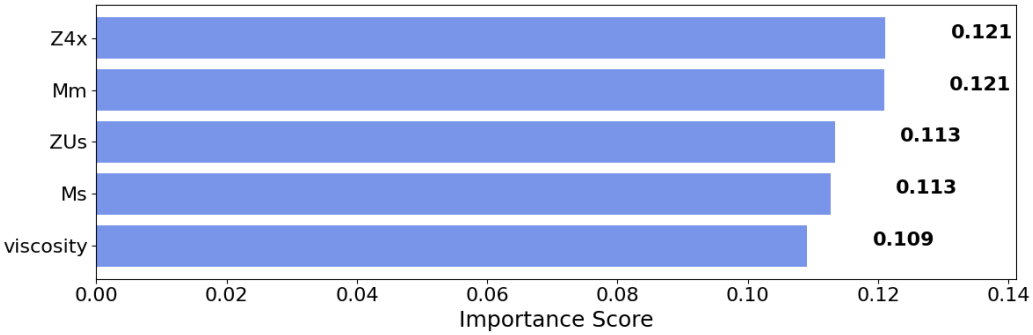


Figure 26: Feature Importance based on Mutual Information Scores. The features are ranked in descending order of their MI values, highlighting the top 5 Features.

We can see how the ranked bottom looks on the other end of the scope in Figure 27. As we touch on ahead, the scale at which features contribute to the outcome of non-conform pieces highly varies from feature to feature, as some features contribute a hundred times more to that outcome.

6.2 RANDOM FOREST IMPORTANCE

The Random Forest (RF) algorithm introduced in 2001 [38], is an ensemble of learning method widely used for classification and regression tasks, that has since established itself as one of the most popular algorithms in machine learning [39]. RF popularity comes from its wide range of applications, good performances on high dimensionality problems and fast computation speeds along with easy tuning.

In a Random Forest model, the Mean Decrease Inpurity (MDI) for each feature measures its contribution to reducing impurity at the nodes where it is used to split the data. Impurity

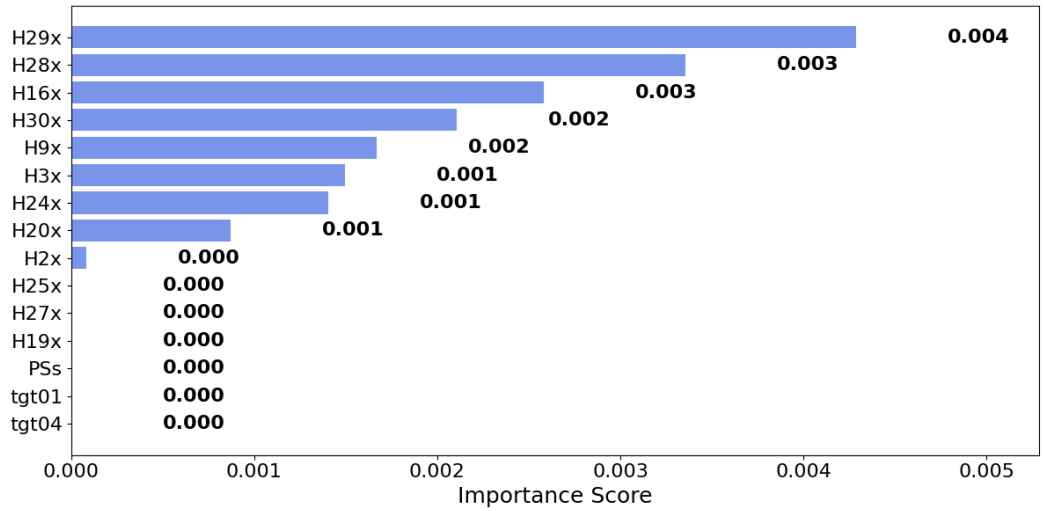


Figure 27: Feature Importance based on Mutual Information Scores. The features are ranked in descending order of their MI values, highlighting the bottom 10 Features.

at a node can be measured by *Gini impurity* (for classification) or *Mean Squared Error (MSE)* (for regression) [31].

For each feature, we calculate the MDI by summing the reduction in impurity across all nodes and trees where the feature is used as a split. This process is explained in high detail in **Classification and Regression Trees** [40] and is methodology used by the Python library Scitkit-learn, when calculating feature importance via Random Forest model.

6.2.1 *Results and Visualisation*

Results obtained by this methodology are visible in Figure 28, in a similar format to the previous solution where Mutual Information regression was employed. At a first glance we can immediately recognise familiar features that top the charts in both instances, further emphasizing their impact and moreover the congruence of information and results between both approaches.

6.3 DISCUSSION

The results from both the Mutual Information Regression and Random Forest Importance approaches present a striking level of agreement, with the same features consistently ranking at the top in terms of their importance. This alignment not only strengthens the validity of the findings but also underscores the robustness of the identified features across different methodologies.

The MI method, with its focus on quantifying the shared information between input features and the outcome, offers a more direct measure of dependence. In contrast, the RF algorithm, using impurity reduction as a criterion, provides a more nuanced view by

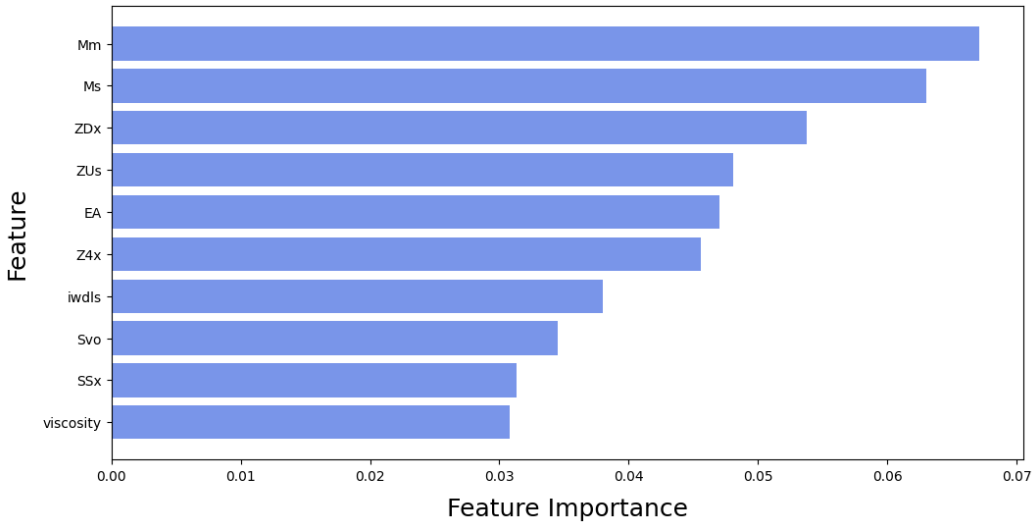


Figure 28: Feature Importance based on Random Forest Importance. The features are ranked in descending order of their importance values, highlighting the top 10 Features.

considering how features affect the overall decision-making process across multiple decision trees. Despite these methodological differences, both techniques point to a similar set of critical features, suggesting that their impact is not only statistically significant but also practically meaningful in the context of production quality control.

This agreement between the two approaches supports the viability and reliability of the results, indicating that the identified features are genuinely influential in predicting the occurrence of non-conformities. It also suggests that both methods can be used complementarily to reinforce the insights derived from one another, making the findings more robust and applicable in industrial settings.

Key takeaways from this analysis include:

- High feature importance in torque related features, this goes for Mm (Average Torque value during screw rotation) and Ms (Peak Torque value during screw rotation). This could relate to material quality, since torque in this context is a direct reflection of difficulty to achieve desired levels of compaction pre-injection. Notably, this finding aligns with results discussed in Section 4.3, where these two features also emerged as key indicators for detecting outliers.

- High feature importance in cycle time related features such as ZDx (Efective Dosage time), ZUs (Cycle final mould release time) and Z4x (Cooling cycle time). Though this features display strong correlation to the defected injection pieces, their direct impact may be hard evaluate. ZDx and ZUs are both consequences of other inputs, so their variation in values is a direct result from its dependencies. On the other hand Z4x is a value inputted by the operator, having its small variations due to demands from the process where kinematic movements always vary slightly and from changes on the inputted value from time to time where adjustments to the injection process are made.

- Lower feature importance in temperature-related features (e.g., tgt04 and tgt01) may indicate that temperature variation within the observed range has limited impact. With that said, we immediately can not extrapolate from this results that temperature is a less relevant factor to the quality of the process. What we can instead conclude is that changes to the values of temperature features in the scales present within the dataset, were decisively not enough to impact injection piece quality outcome.

It is important to emphasize that the conclusions drawn in these approaches are based solely on the analysis of the dataset, its features and the general understanding of the injection molding process from an Engineering perspective. These insights are limited to the scope of the available data and the context of this research. Engineers at Aspöck, with their in-depth expertise in the specific production environment, could leverage these findings to uncover additional, process-specific insights to guide further refinement of the injection process.

Deeper knowledge could potentially allow for a more precise interpretation of the data or even the draw of different conclusions, highlighting that our limited knowledge of the process could prevent us from recognising key relations, be them less significant numbers wise, that would however be more relevant to the specific injection process.

6.4 SUMMARY

This chapter explored the contribution and importance of each feature in the dataset, providing additional explainability to support the findings from the exploratory and modeling phases. By combining Mutual Information Regression and Random Forest Importance, we quantified how individual features influence the likelihood of non-conforming pieces, highlighting the most critical parameters in the injection process.

The analysis emphasizes explainability without compromising feature transparency, allowing process engineers to interpret the results in the context of production. High-impact features, such as torque-related and cycle-time variables, were consistently identified across methods, while temperature-related features showed limited influence within the observed ranges. These insights reinforce the robustness of the feature selection, offering actionable guidance for understanding and potentially improving production quality.

Ultimately, this chapter demonstrates that feature explainability can provide both statistical and practical understanding, supporting data-driven decision-making while preserving the traceability of each feature's role in the process.

7

CONCLUSION

This thesis culminated in the successful development of a high-performing, scenario-specific model capable of identifying faulty production time zones with a weighted F1-score of 0.94, achieved on 2 of the 3 refined datasets used. This highlights the model's ability to minimize both false positives and false negatives in an imbalanced dataset. Despite the extensive effort invested across the full scope of this project, the final model is both simple and efficient, training in just a few minutes on the available data while delivering excellent results where alternative approaches failed.

Beyond model performance, the project achieved two equally critical outcomes: first, it delivered feature level explainability that can inform machine parameter tuning by process engineers in near real time; and second, it assembled a modular, data-driven development pipeline spanning exploratory analysis, feature selection, model experimentation, and validation that addresses the challenges of raw, volatile industrial data. Key to this achievement was the adaptation to the lack of precise defect labeling by introducing the concept of "faulty time zones" (defined as a seven-minute range), thereby enabling a semi-supervised learning structure grounded in domain understanding. This work serves as a practical reference for navigating the complex interface between industrial needs and data science capabilities, highlighting the importance of flexibility, domain adaptation, and methodological rigour.

From a computational standpoint, this thesis presents a robust and modular data science workflow tailored for raw, high-frequency industrial sensor data. The project began with a fully unsupervised approach, motivated by the imprecise alignment between defect documentation and production timestamps. This initial constraint prompted the use of anomaly detection techniques and clustering-based methods, which subsequent convergence toward better-performing models through a custom evaluation framework. As results matured, the pipeline evolved into a semi-supervised architecture that strategically tuned unsupervised models by optimizing them against manually derived labels within loosely defined faulty production time windows. This setup necessitated the development of an internal validation strategy that adapted unsupervised models and traditional classification metrics, particularly the F1-score, to reflect performance on weakly labelled data.

Hyperparameter tuning was computationally intensive, leveraged on brute-force grid search techniques, made feasible only through CUDA accelerated execution. Without GPU support, model experimentation at this scale would have been infeasible within the given time constraints, likely forcing a more limited exploration of the model space.

The project is not only an achievement in terms of model success but also a testament to the foundational work that made such success possible. Extensive effort was devoted to exploring, understanding, processing, and selecting the data fed to the various models.

Dimensionality reduction and feature selection combined statistical rigour with interpretability, employing domain-constrained PCA (per feature group), Pearson correlation filtering, and Mann–Whitney U tests all to validate and isolate feature contribution. Moreover, experimental datasets were curated to isolate and compare torque-related features, whose relevance emerged during data exploration stages, proving to significantly correlated with injection part quality.

At a later stage, LSTM models became the primary focus in response to the limitations and suboptimal results observed with earlier approaches. The adoption of LSTM networks emerged as a key differentiator, providing extreme improvements in results from the get-go. While other models achieved reasonable values of accuracy, the low F1-scores obtained were less than desirable, a very direct result of an extremely unbalanced dataset as it is expected from the context of having a minimal amount of faulty parts compared to acceptable ones. LSTMs far out-scaled previous models not only in terms of accuracy and any other metric, but mostly F1-score, severely reducing the amount of false negatives and positives. Unlike static models, LSTMs are inherently capable of capturing the temporal dependencies and dynamic characteristics intrinsic to manufacturing processes. Their superior performance not only validated the hypothesis that time-aware models are better suited for this task, but also established a robust foundation for further advancements. In particular, these findings open promising avenues for exploration of transformer-based architectures, which extend the temporal modelling capabilities of LSTMs by incorporating mechanisms for global attention. Much like LSTMs represent a significant improvement over traditional RNNs, transformer-based solutions may be viewed as the natural progression in model sophistication, especially given the long-term dependencies and delayed effects observed in mould injection processes.

One key takeaway from addressing this scenario is the recognition that mathematically defined outliers, in the form of purely numerical deviations from statistical norms, do not inherently correspond to defective parts. Large deviations in certain parameters may have negligible impact on product quality, while small, contextually significant variations particularly when they occur in critical features or in synergistic patterns, may be far more predictive of non-conformities. This insight underscores the necessity of robust feature selection methods and reinforces the value of interpretability in model outputs. Ultimately, it highlights the importance of deploying models capable of navigating subtle, multivariate deviations rather than relying solely on amplitude-based anomaly detection. The success of the solution, therefore, hinged not only on raw performance metrics, but also on the model's capacity to discern meaningful variation in a complex, high-dimensional, and interdependent process landscape.

All code was structured for modular reuse and experimentation, implemented as version-controlled Jupyter notebooks with callable external modules and packaged within a Docker container to ensure full reproducibility. This pipeline stands as a technical contribution in itself, providing a transferable template for data-driven fault detection under weak supervision in non-stationary industrial systems.

Although the model has yet to be deployed in a live production setting, preliminary evaluation of its results by process engineers indicated strong potential. Particularly noteworthy was the unexpected yet logical emphasis placed by the model on torque-related features. While domain experts initially anticipated parameters such as viscosity to dominate the feature importance scale, the highlighted torque dynamics, likely reflecting material quality, humidity, and density fluctuations, which in turn increase the torque required for effective compaction, offered new perspectives for process monitoring. These findings suggest that torque may serve as a key early indicator of part conformity deviations, rendering it a valuable parameter for real-time feedback systems. Even in the absence of immediate deployment, the thesis has provided Aspöck Portugal with actionable insights into feature relevance. Additionally, the observation that certain theoretically important features from a process standpoint, such as temperature, held low predictive power within this finely regulated process has helped refocus attention on more sensitive indicators of process stability for this given production setting. Variability in machine tuning across the six-month dataset reinforced the need for models capable of adapting to non-stationary operating conditions, the LSTM-based architecture proved particularly adept at modelling subtle, gradual shifts inherent to an industrial environment shaped by continuous, small-scale adjustments by machine operators. These shifts stem from gradual changes in environmental variables, such as air pressure, temperature, and humidity, which naturally fluctuate over the course of a year and subtly influence optimal machine parameters.

From a methodological perspective, the project represents a significant step toward bridging the academic rigour of unsupervised anomaly detection with the practical demands of an industrial setting. Aligning model capabilities with engineering needs, particularly regarding explainability and actionability, was a recurring priority. This alignment was achieved through custom strategies for evaluating individual feature contributions and designing interpretability-preserving dimensionality reductions that remained meaningful to domain experts.

The reflective process of model development underscored several valuable lessons. The most impactful aspects extended beyond raw model performance to encompass the entire pipeline—from data understanding and cleaning to labelling strategies, metric engineering, and validation routines. The ability to continuously adapt based on insights gained at each stage was essential, iterative feedback between exploratory analysis and modelling proved critical to refining the pipeline and achieving alignment between analytical rigour and domain relevance, further reinforcing the significance of a data-driven approach.

The final result of this work is multifaceted: a functional model pipeline with strong performance on historical data, a reproducible methodology grounded in real data and realistic substantiated assumptions, and a document that contributes significant value to both academia and industry. The thesis not only demonstrates technical competency but also exemplifies a comprehensive approach to problem-solving under realistic constraints, serving as a reference for data scientists, engineers, and researchers seeking to apply advanced learning techniques to imperfect industrial datasets with complex, time-sensitive behaviours.

7.1 ACHIEVEMENT OF OBJECTIVES

Three objectives were defined at the outset of this work in Section 1.2, all of which were fully addressed.

Detection of events/anomalies was the first defined objective. This was successfully met. A high-performing model was developed that reliably identified faulty production time zones, achieving a weighted F1-score of 0.94 on two of the three refined datasets. Although specific identification of individual faulty parts remains impossible due to the nature of the process and data, the results are highly satisfactory and can be used to drive direct improvements in the production setting.

Due to the nature of the project one of the defined goals was to benchmark models, as such, this thesis conducted a thorough comparative evaluation of a broad range of machine learning models, including unsupervised anomaly detection methods, clustering techniques, and supervised/semi-supervised architectures. Earlier approaches such as clustering and static classifiers provided valuable baselines but ultimately demonstrated limitations in handling weakly labelled, imbalanced data. This benchmarking justified the transition toward LSTM-based models, which significantly outperformed all alternatives in both detection accuracy and F1-score. The analysis also highlighted the specific strengths and weaknesses of each method, providing practical insight into their behaviour under the same industrial conditions.

Although explainability was not achieved through the unsupervised models as initially expected, the project still provided substantial insight into the process through its various exploratory and graphical analyses. A dedicated chapter later focused on explainability, identifying torque-related features among others as key predictors of faulty production, a finding validated by domain experts and considered actionable for real-time process monitoring. These results aligned closely with insights uncovered throughout the project, further cementing their validity and practical importance.

In summary, all three objectives were achieved. Moreover, the work extended beyond these goals by delivering a modular, reproducible pipeline and a semi-supervised methodology applicable to other industrial fault detection scenarios.

7.2 FUTURE WORK

Future research may focus on the live deployment and monitoring of the developed models, including the incorporation of continuous learning strategies to handle data drift and process changes. The success of LSTM models suggests that further exploration into temporal architectures, particularly transformers, could unlock even higher performance. As transformer-based models continue to evolve, their capacity to handle long-range dependencies and multi-sensor fusion may prove especially relevant.

Additionally, expanding the labelling strategy to include more granular feedback from quality control and integrating those labels into a semi-supervised or weakly supervised learning framework could further enhance model generalizability. There is also an opportunity to design operator-facing interfaces that leverage the explainability outputs, thereby closing the loop between model insights and production-level decisions. For instance, a small warning indicator could be implemented to flag the onset of suboptimal parameter deviations, prompting operators and engineers to review current settings and operations to mitigate the risk of producing non-conforming parts. This can be achieved by model deployment on industrial microcontrollers or PLCs using lightweight inference frameworks, directly interfacing with the injection moulding machine's control system.

Finally, the modular and interpretable nature of the framework makes it a valuable foundation for future work in industrial AI, providing a toolbox of techniques adaptable to similar manufacturing contexts with minimal assumptions and strong empirical grounding.

BIBLIOGRAPHY

[1] Ramachandran K K. 'DATA SCIENCE IN THE 21ST CENTURY: EVOLUTION, CHALLENGES, AND FUTURE DIRECTIONS'. In: 1 (Jan. 2024).

[2] Saurabh Vaidya, Prashant Ambad and Santosh Bhosle. 'Industry 4.0 – A Glimpse'. In: *Procedia Manufacturing* 20 (2018). 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA, pp. 233–238. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2018.02.034. URL: https://www.sciencedirect.com/science/article/pii/S2351978918300672.

[3] 3ERP. 'Plastic Injection Molding: The Complete Guide to Injection Molding'. In: (2025). Accessed on August 14, 2025.

[4] Gurpreet Matharu et al. 'Empirical Study of Agile Software Development Methodologies'. In: *ACM SIGSOFT Software Engineering Notes* 40 (Feb. 2015), pp. 1–6. DOI: 10.1145/2693208.2693233.

[5] Carl Boettiger. 'An introduction to Docker for reproducible research'. In: *ACM SIGOPS Operating Systems Review* 49.1 (2015), pp. 71–79.

[6] Adrian Benton et al. 'Deep learning for NLP at the edge: A preliminary study'. In: *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*. 2017, pp. 31–36.

[7] V. Chandola, A. Banerjee and V. Kumar. 'Anomaly detection: A survey'. In: *ACM Computing Surveys (CSUR)* 41.3 (2009), pp. 1–58.

[8] Adam Bardos, Panagiotis Doupidis, Theodoros Kotsiopoulos et al. 'Anomaly Detection in Industrial Processes: Supervised vs. Unsupervised Learning and the Role of Explainability'. In: *Open Research Europe* 5 (2025), p. 8. DOI: 10.12688/openreseurope.18593.1.

[9] Guansong Pang et al. 'Deep Learning for Anomaly Detection: A Review'. In: *ACM Computing Surveys* 54.2 (2020), pp. 1–38. DOI: 10.1145/3439950.

[10] Manish Gupta et al. 'Outlier Detection for Temporal Data: A Survey'. In: *IEEE Transactions on Knowledge and Data Engineering* 26.9 (2014), pp. 2250–2267. DOI: 10.1109/TKDE.2013.184.

[11] Faouzi Tayalati et al. 'Hybrid Approach Integrating Deep Learning-Autoencoder With Statistical Process Control Chart for Anomaly Detection: Case Study in Injection Molding Process'. In: *IEEE Access* 12 (July 2024), p. 23. DOI: 10.1109/ACCESS.2024.3425582.

[12] Hanling Wang et al. 'Unsupervised anomaly detection via generative adversarial networks: poster abstract'. In: New York, NY, USA: Association for Computing Machinery, 2019. ISBN: 9781450362849. DOI: 10.1145/3302506.3312605.

[13] Reinhard Schiffers and Philipp-Jan Honysz. 'Anomaly detection in injection molding process data based on unsupervised learning'. In: *Zeitschrift Kunststofftechnik* 1 (Jan. 2018), pp. 301–347. DOI: 10.3139/0999.02052018.

[14] David Arthur and Sergei Vassilvitskii. 'How slow is the k-means method?' In: New York, NY, USA: Association for Computing Machinery, 2006. ISBN: 1595933409. URL: 10.1145/1137856.1137880.

[15] Markus M. Breunig et al. 'LOF: identifying density-based local outliers'. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 93–104. ISBN: 1581132174. URL: https://doi.org/10.1145/342009.335388.

[16] Ricardo J.G.B. Campello et al. 'Density-based clustering'. English. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (Apr. 2020). ISSN: 1942-4787. URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1343.

[17] Erich Schubert and Michael Gertz. 'Improving the Cluster Structure Extracted from OPTICS Plots'. In: *Lernen, Wissen, Daten, Analysen*. 2018. URL: https://api.semanticscholar.org/CorpusID:52088333.

[18] Yizong Cheng. 'Mean shift, mode seeking, and clustering'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.8 (1995), pp. 790–799. DOI: 10.1109/34.400568.

[19] Yizong Cheng. ' Mean Shift, Mode Seeking, and Clustering '. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 17.08 (Aug. 1995), pp. 790–799. ISSN: 1939-3539. DOI: 10.1109/34.400568. URL: https://doi.ieeecomputersociety.org/10.1109/34.400568.

[20] Fei Tony Liu, Kai Ming Ting and Zhi-Hua Zhou. 'Isolation-Based Anomaly Detection'. In: *ACM Trans. Knowl. Discov. Data* 6.1 (2012). ISSN: 1556-4681. URL: https://doi.org/10.1145/2133360.2133363.

[21] Corinna Cortes and Vladimir Vapnik. 'Support-Vector Networks'. In: *Mach. Learn.* 20.3 (Sept. 1995). ISSN: 0885-6125. URL: https://doi.org/10.1023/A:1022627411411.

[22] Mariette Awad and Rahul Khanna. 'Support Vector Machines for Classification'. In: *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Berkeley, CA: Apress, 2015, pp. 39–66. DOI: 10.1007/978-1-4302-5990-9_3.

[23] David M. J. Tax and Robert P. W. Duin. 'Support Vector Data Description'. In: *Machine Learning* 54.1 (2004), pp. 45–66. URL: 10.1023/B:MACH.0000008084.60811.49.

[24] Sepp Hochreiter and Jürgen Schmidhuber. 'Long Short-Term Memory'. In: *Neural Computation* 9 (Nov. 1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[25] Sepp Hochreiter. 'Untersuchungen zu dynamischen neuronalen Netzen'. In: (Apr. 1991).

[26] Aston Zhang et al. *Long Short-Term Memory (LSTM)*. https://d2l.ai/chapter_recurrent-modern/lstm.html. Accessed: 2025-04-26. 2023.

[27] P. Malhotra et al. 'Long short-term memory networks for anomaly detection in time series'. In: *Proceedings of the 23rd European Symposium on Artificial Neural Networks (ESANN)*. 2015.

[28] Kyle Hundman et al. 'Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding'. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)* (2018), pp. 387–395. DOI: 10.1145/3219819.3219845.

[29] Raghavendra Chalapathy and Sanjay Chawla. 'Deep Learning for Anomaly Detection: A Survey'. In: *CoRR* abs/1901.03407 (2019). arXiv: 1901.03407. URL: http://arxiv.org/abs/1901.03407.

[30] Yilixiati Abudurexiti et al. 'An explainable unsupervised anomaly detection framework for Industrial Internet of Things'. In: *Computers & Security* 148 (Sept. 2024), p. 104130. DOI: 10.1016/j.cose.2024.104130.

[31] Xabier Olano et al. 'Outcomes and features of the inspection of receiver tubes (ITR) system for improved O&M in parabolic trough plants'. In: vol. 2033. Nov. 2018, p. 030011. DOI: 10.1063/1.5067027.

[32] Nadim Nachar. 'The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution'. In: *Tutorials in Quantitative Methods for Psychology* 4 (Mar. 2008). DOI: 10.20982/tqmp.04.1.p013.

[33] Wikipedia. *Mann–Whitney U test — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Mann%E2%80%93Whitney%20U%20test&oldid=1254931243. [Online; accessed 10-November-2024]. 2024.

[34] SciPy Community. *scipy.stats.mannwhitneyu*. Accessed: 2024-11-10. 2024. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mannwhitneyu.html#scipy.stats.mannwhitneyu.

[35] Fabian Pedregosa et al. *Scikit-learn: Machine Learning in Python*. Vol. 12. 2011, pp. 2825–2830.

[36] Jin Hao. 'Input Selection Using Mutual Information - Applications to Time Series Prediction'. Master's Thesis. Department of Computer Science and Engineering: Helsinki University of Technology, Sept. 2005. URL: https://research.cs.aalto.fi/aml/Publications/Publication58.pdf.

[37] Madhur Tulsiani and Shubhendu Trivedi. *Information and Coding Theory Lecture 2: Joint Entropy and Source Coding Theorem*. Lecture notes, Autumn 2014, University of Chicago. 2014. URL: https://home.ttic.edu/~madhurt/courses/infotheory2014/l2.pdf.

[38] Leo Breiman. 'Random Forests'. In: *Machine Learning* 45.1 (2001). DOI: 10.1023/A:1010933404324.

[39] Manuel Fernández-Delgado et al. 'Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?' In: *Journal of Machine Learning Research* 15.90 (2014), pp. 3133–3181. URL: http://jmlr.org/papers/v15/delgado14a.html.

[40] L. Breiman et al. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN: 9780412048418. URL: https://books.google.pt/books?id=JwQx-WOmSyQC.

DECLARATION

I declare on my honor that the work presented in this dissertation, entitled *"INJECTION MOLDING PROCESS MONITORING BASED ON MACHINE LEARNING ALGORITHMS"*, is original and was carried out by Pedro Alexandre da Ponte Costa (2220129) under the supervision of Professor Sílvio Priem Mendes, PhD (smendes@ipleiria.pt) and Professor Paulo Jorge Gonçalves Loureiro, PhD (paulo.loureiro@ipleiria.pt).

Leiria, September 2025

Pedro Alexandre da Ponte Costa