

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automated Feature Engineering for Predictive Maintenance

José Albano de Almeida Gaspar



Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Vitor Cerqueira

Second Supervisor: Prof. Carlos Soares

July 11, 2025

Automated Feature Engineering for Predictive Maintenance

José Albano de Almeida Gaspar

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. João Moreira

Referee: Prof. Nuno Lourenço

Supervisor: Prof. Vitor Cerqueira

July 11, 2025

Resumo

A Manutenção Preditiva (PdM) está a tornar-se cada vez mais uma estratégia fundamental na Indústria 4.0, aproveitando os avanços em Inteligência Artificial (AI), Internet das Coisas (IoT) e nos Sistemas Ciberfísicos (CPS) para otimizar o desempenho das máquinas e, ao mesmo tempo, reduzir os custos de manutenção. No entanto, o processo de engenharia de atributos para aplicações em PdM continua a ser desafiante devido à complexidade dos dados gerados pelos sensores, que geralmente incluem séries temporais multivariadas irregulares com problemas relacionados com os dados, como ruído e valores em falta.

Esta dissertação apresenta o *optimusPipe*, uma ferramenta para automatizar o processo de engenharia de atributos no contexto de PdM. A ferramenta aborda os principais desafios no processamento de dados de sensores incorporando várias operações, como a reamostragem para irregularidade, o preenchimento automático para valores em falta e a análise de correlação para a seleção de atributos. Utilizando a otimização de procura em grelha (*grid search*) guiada por métricas de desempenho do modelo, a ferramenta desenvolvida cria automaticamente configurações de engenharia de atributos eficazes.

Validámos a nossa ferramenta através de um caso de estudo na indústria farmacêutica, com foco nas máquinas de compressão de comprimidos, onde as mudanças frequentes de configuração e os fatores operacionais podem levar à degradação do desempenho das mesmas. Os resultados experimentais demonstram a eficácia da ferramenta no tratamento de vários desafios de dados, ao mesmo tempo que automatiza o processo de engenharia de atributos. Além disso, fornecemos análises empíricas de diferentes operações de engenharia de atributos, métodos de extração de features em séries temporais e algoritmos de aprendizagem no contexto de PdM na indústria farmacêutica. Este trabalho contribui para o avanço da área ao oferecer uma abordagem generalizável para engenharia de atributos automatizada no contexto de PdM, reduzindo a dependência de conhecimento técnico e de domínio.

Abstract

Predictive Maintenance (PdM) is increasingly becoming a key strategy in Industry 4.0, leveraging advances in Artificial Intelligence (AI), Internet of Things (IoT), and Cyber-Physical Systems (CPS) to optimize the performance of machinery while reducing maintenance costs. However, the feature engineering process for PdM applications remains challenging due to the complexity of sensor-generated data, which often includes irregular multivariate time series with data-related problems such as noise and missing values.

This dissertation introduces *optimusPipe*, a framework for automating the feature engineering process in PdM applications. The framework addresses key challenges in processing sensor data by incorporating various operations such as resampling for irregularity, forward fill for missing values, and correlation analysis for feature selection. Using grid search optimization guided by model performance metrics, the developed pipeline builds effective feature engineering configurations automatically.

We validate our framework through a case study in the pharmaceutical industry, focusing on pill compression machines where frequent configuration changes and operational factors can lead to performance degradation. The experimental results demonstrate the framework's effectiveness in handling various data challenges while automating the feature engineering process. Additionally, we provide empirical analyses of different feature engineering operations, time series feature extraction methods, and learning algorithms in the context of pharmaceutical manufacturing PdM. This work contributes to the field by offering a generalizable approach to automated feature engineering for PdM applications, reducing the dependency on domain and technical expertise.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

The specific Sustainable Development Goals mentioned have the following names:

SDG 8 Promote sustained, inclusive and sustainable economic growth, productive employment, and decent work for all.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization, and foster innovation.

SDG 12 Ensure sustainable consumption and production patterns.

SDG 13 Take urgent action to combat climate change and its impacts.

SDG	Target	Contribution	Performance Indicators and Metrics
8	8.1	Minimizing maintenance costs and reducing production downtime through predictive maintenance	Predictive performance of ML models (AUC, precision, recall), reduction in false negatives, improved machine uptime
	8.2	Enhancing labor productivity through automation of feature engineering tasks	Time savings for data scientist-s/engineers, reduction in manual data preprocessing steps
9	9.1	Promoting the adoption of smart manufacturing technologies through modular and reusable software tools	Faster deployment of predictive maintenance systems, scalability to different industrial environments
	9.5	Supporting research and innovation in AI-based industrial maintenance	Integration of advanced ML tools, benchmarking of techniques across real-world datasets
12	12.2	Contributing to efficient resource usage by predicting failures and avoiding unnecessary part replacements	Reduced waste from unneeded maintenance, prolonged equipment life cycle
13	13.2	Supporting climate-related actions by optimizing industrial operations and reducing energy/resource waste	Lower environmental impact due to less frequent emergency maintenance and machine failures

Funding

This work was partially funded by projects AISym4Med (101095387) supported by Horizon Europe Cluster 1: Health, ConnectedHealth (n.º 46858), supported by Competitiveness and Internationalisation Operational Programme (POCI) and Lisbon Regional Operational Programme (LISBOA 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) and Agenda “Center for Responsible AI”, nr. C645008882-00000055, investment project nr. 62, financed by the Recovery and Resilience Plan (PRR) and by European Union - NextGeneration EU, and also by FCT plurianual funding for 2020-2023 of LIACC (UIDB/00027/2020 UIDP/00027/2020);

José Albano de Almeida Gaspar

Acknowledgements

First of all, I want to thank the only person who never let me down, my mother, for the support she gave me during my journey through life. Special thanks to the other family members: my father, my grandparents, and my cousins.

I want to thank my friends for the amazing memories we created together, the good times we spent together, and the unforgettable five years we spent laughing, crying, and struggling during the exam periods.

I want to thank the women I met along the way for letting me understand what is worth fighting for - love.

I want to thank my supervisor, Vitor Cerqueira, for guiding me during the entire time with his wisdom and experience, and Prof. Carlos Soares, for the useful insights given throughout the development of this dissertation.

Lastly, I want to thank Michael Bay for his amazing work on directing the Transformers movies.

José Albano de Almeida Gaspar

*“Fate rarely calls upon us
at a moment of our choosing.”*

Optimus Prime - Transformers Revenge of the Fallen (2009)

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.2	Problem Definition	2
1.3	Research Objectives	3
1.4	Research Questions	3
1.5	Dissertation structure	4
2	State-of-the-Art	5
2.1	Types of Maintenance	5
2.2	Example Applications	6
2.3	Irregular Multivariate Time Series	6
2.4	Feature Engineering	7
2.4.1	Dealing with time series irregularity	8
2.4.2	Dealing with missing values	9
2.4.3	Feature Extraction	9
2.4.4	Anomaly Definition	11
2.4.5	Feature selection	12
2.4.6	Normalization	13
2.5	Automatic feature engineering	13
2.6	Modeling	15
2.6.1	Learning approaches	15
2.6.2	Imbalance problem	16
2.7	Evaluation	16
2.7.1	Evaluation Metrics	17
2.7.2	Cross-Validation	17
2.8	Conclusions	18
3	Methodology	19
3.1	Problem Definition	19
3.2	Automatic Feature Engineering	20
3.2.1	Pipeline	21
3.2.2	Transformations	22
3.2.3	Modeling	25
3.2.4	Pipeline Optimization	25
4	Case Study	27
4.1	Context	27
4.2	Dataset	28

4.3	Data pre-processing	29
4.4	Problem definition	29
4.5	Pipeline execution	30
5	Experiments	32
5.1	Evaluation Protocol	32
5.2	Operation performance	34
5.3	Best performance parameters	35
5.4	Comparative results	35
5.5	Feature extractors	37
5.5.1	Feature importance	37
5.5.2	Execution Time	38
5.6	Disussion	39
5.6.1	Strengths of the proposed framework	39
5.6.2	Limitations	40
6	Conclusions and Future Work	42
6.1	Research question answers	42
6.2	Future work	43
	References	45
A	optimusPipe package	52
A.1	optimusPipe example	52

List of Figures

2.1	Typical feature engineering process for PdM tasks	8
2.2	Lag-based feature example (Pressure_mean) using 3 lags	10
3.1	Proposed framework to automate feature engineering	21
3.2	Processing steps supported by pipeline in sequential order	21
3.3	Optimization conducted on the feature extraction step	24
4.1	Used processing steps of the pipeline in sequential order	30
5.1	Optimal model evaluation process	33
5.2	Top 10 feature importance distribution	38
5.3	Top 10 most important features for the best pipeline using distinct feature extractors	39

List of Tables

2.1	Irregular Multivariate Time Series Example	7
3.1	Implemented transformations in each pipeline step	22
3.2	Configuration space	26
4.1	Configuration space for case study	31
5.1	Imputation Analysis	34
5.2	Feature Extraction Analysis	34
5.3	Feature Selection Analysis	35
5.4	Comparison of best pipeline (with multiple tree-based classifiers) and baseline pipeline on test set.	36

List of Acronyms

AI	Artificial Intelligence
ARIMA	Autoregressive Integrated Moving Average
AUC	Area Under (ROC) Curve
CART	Classification And Regression Trees
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
IoT	Internet-of-Things
KNN	K nearest neighbours
LSTM	Long-Short Term Memory Neural Network
ML	Machine Learning
PCA	Principle Component Analysis
PdM	Predictive Maintenance
PvM	Preventive Maintenance
R2F	Run-To-Failure
ROC	Receiver Operating Characteristic
RUL	Remaining Useful Life
SVM	Support Vector Machine

Chapter 1

Introduction

The industrial revolutions brought significant changes in the industrial sector due to advances in electricity, mechanization, and technology. Experts now refer to the current phase as Industry 4.0 [32], which is characterized by advancements in the internet-of-things [9] and cyber-physical systems [61]. These technologies enable businesses to gather and examine data from machinery and operations, which reduces maintenance expenses and boosts output.

1.1 Context and Motivation

Industrial machines serve as crucial components in automating production systems. Optimizing their performance (i.e., minimizing downtime and maximizing productivity) with continuous monitoring and regular maintenance is vital for maximizing operational efficiency. However, effective and efficient maintenance processes are often challenging to implement due to the high costs and resource allocation.

Predictive Maintenance (PdM) has emerged as a key strategy for improving machine performance and reducing downtime. Its emergence is attributed to the recent advances in Artificial Intelligence (AI), Cyber-Physical Systems (CPS), and the Internet of Things (IoT) [9, 61]. PdM involves continuous monitoring of machine operations and the use of machine learning models to predict when maintenance should be performed. PdM has been shown to be effective in reducing maintenance costs and improving machine performance [84].

IoT transforms common objects into intelligent devices, enabling both monitoring and physical world interaction. For example, extending a traditional watch into a smartwatch capable of measuring vital signs and triggering emergency responses when necessary. This interconnected network of smart devices enables companies to deploy sensors on industrial machinery, and real-time data that monitors the machine operations [9].

This large-scale data collection also enables data-driven decision-making using machine learning approaches. By leveraging AI algorithms to analyze and learn from this data, companies can

predict future events such as potential machine failures before they occur [84]. Accurate predictions can lead to the implementation of more effective and efficient maintenance processes, minimizing downtime and maintenance costs.

The implementation of PdM offers significant economic benefits by reducing costs associated with the maintenance process. Specifically, PdM delivers multiple operational benefits including reduced machine faults, minimized repair downtime, optimized spare parts inventory management, extended component lifespan, increased production output, improved operator safety, enhanced repair verification processes, and overall profit improvement [15].

PdM can be applied in different application domains, including in wind turbines [25], manufacturing equipment, [73], railway systems [29], electric power systems [49], and on the pharmaceutical industry, as the conducted case study of this dissertation, among many others. Due to its applicability across varied domains, PdM is becoming a widely researched topic in the scientific literature [15].

1.2 Problem Definition

A key characteristic of PdM problems is that they are typically based on sensor-generated datasets. These datasets are collected with various sensors over time, potentially at irregular intervals [15, 25, 49]. This results in challenging, multivariate time series that often contain noise and other irregularities such as missing values or outliers. Consequently, the feature engineering process becomes particularly challenging and time-consuming, and often dependent on domain knowledge for effective dataset preparation.

In machine learning, a feature set representing a dataset is a fundamental component for building accurate predictive models [16, 36]. The importance of features directly impacts the performance and generalization capabilities of models. Therefore, feature engineering is a critical step in machine learning tasks. This process involves transforming raw data into an effective representation that captures the underlying patterns and relationships for building accurate predictive models.

In PdM tasks, the feature engineering process involves unique challenges due to the sensorial nature of data [29]. Sensor readings are often high-dimensional and amenable to data difficulty problems such as noise, outliers, and missing values that require robust preprocessing methods. The process of creating useful features requires an in-depth understanding of the domain and the data. Domain expertise becomes crucial in identifying relevant features that capture the patterns related to failures. Besides that, the feature engineering process also involves applying operations such as feature selection (e.g. [49]), and normalization for adequate modeling [36].

Several feature engineering approaches have been shown to be effective in different PdM applications. These include computing aggregated statistics from numeric sensor values, implementing correlation-based methods [13, 24, 39, 67] and applying dimensionality reduction techniques such as Principal Component Analysis (PCA) [25, 49, 73] for feature selection, and the normalization

of the dataset [4, 39, 67]. Additionally, incorporating lag-based features has shown promising results in capturing temporal patterns in PdM datasets [13, 23, 34].

The feature engineering process in PdM is typically ad-hoc, as it must be tailored to the specific characteristics of each given application. Different domains comprise different types of machines that operate under distinct conditions. Consequently, the feature engineering process of identifying relevant failure predictors and, more generally, transforming raw sensor-based data into a prepared dataset involves domain-specific knowledge. This inherent variability has hindered the development of feature engineering approaches that generalize methodologies across different PdM scenarios.

1.3 Research Objectives

This dissertation aims to address this gap by developing a framework, called *optimusPipe*, for automating the feature engineering process in PdM problems. We assume that datasets involving PdM can be framed as multivariate, potentially irregular, time series. Our framework builds a configuration space that includes operations that tackle different problems involved in the feature engineering process of PdM tasks. For example, resampling to address irregularity, forward fill to tackle missing values, correlation analysis for the feature selection problem, among others.

Then, we leverage grid search to explore the configuration space and find the best configuration for the feature engineering process. The optimization is guided by the performance of the model in the validation set using a metric such as the area under the ROC curve (AUC).

We address a case study in the pharmaceutical industry, focusing on pill compression machines. These machines operate with different configurations based on the specific type of pill being manufactured. Frequent configuration changes, along with other operational factors, can gradually degrade machine performance. The goal is to monitor the quality of manufactured pills and accurately predict potential machine failures before they occur.

1.4 Research Questions

This dissertation aims to address PdM problems by developing a framework - *optimusPipe* - for automating the feature engineering process in these tasks. Based on the challenges identified above concerning PdM and feature engineering, we aim to address the following research questions:

RQ1: *How can we automate the feature engineering process in PdM problems while handling various data difficulty challenges of multivariate time series data collected on sensors?*

RQ2: *How do different feature engineering operations affect the performance and effectiveness of the feature engineering framework in a pharmaceutical manufacturing application?*

***RQ3:** Which time series feature extraction method is the most effective lag-based feature extraction in a pharmaceutical manufacturing application?*

***RQ4:** How do different learning algorithms compare in terms of predictive performance for a pharmaceutical manufacturing PdM application?*

We address these research questions by developing an automated feature engineering geared towards PdM problems. The framework is tested on real-world data from the pharmaceutical manufacturing domain.

1.5 Dissertation structure

This dissertation is structured as follows: In chapter 2, a state-of-the-art is presented. This chapter begins with a brief presentation on the evolution of maintenance strategies over time, culminating in PdM. To demonstrate how important PdM can be, a section with case study examples is provided. Then it proceeds by introducing, formally, the concept of irregular multivariate time series, the resultant data structure of many PdM problems, including the problem tackled in the case study. The state-of-the-art chapter also overviews progress in the context of automatic feature engineering, a central theme of this dissertation, highlighting both examples of frameworks for the entire feature engineering process, such as VEST [16], PREPIPE [33], ExploreKit [43], and Cognito [46].

Chapter 3 presents the methodology behind the proposed automated feature engineering process. The chapter begins with a formalization of the problem, outlining the specific PdM challenges addressed in this work. The methodology section also details the proposed framework, including the key transformations developed to handle various feature engineering challenges. These transformations involve techniques for addressing irregular time series, missing values, feature extraction, feature selection, and data normalization.

Chapter 4 presents the case study that validates the proposed framework. We start by describing the context and the dataset, which is related to the pharmaceutical manufacturing industry. We also present the specific problem definition used in the case study. We also explain some nuances applied to the framework to address the unique requirements of the pharmaceutical manufacturing case study.

Chapter 5 presents the experiments conducted to validate the developed framework. The chapter begins by detailing the experimental design methodology, including the evaluation metrics and procedures used to assess the framework's performance. Then, we present the experimental results. Moreover, a discussion over the experimental results from the case study is addressed in detail, containing both empirical and company points of view. This chapter ends with a discussion of the pros and cons of the developed work.

The dissertation is concluded in Chapter 6, which summarizes the key contributions and findings of this work. We also evaluate the extent to which the research objectives were achieved by answering the research questions and outline promising directions for future research.

Chapter 2

State-of-the-Art

This chapter provides a background to this dissertation, describing the state-of-the-art in PdM problems. A particular focus is given to the feature engineering process, which is the central theme of this work.

Section 2.1 provides a brief introduction to maintenance tasks and the main approaches used to conduct them. Then, we defined irregular multivariate time series (Section 2.3), the typical data type used to represent datasets concerning PdM problems. In this state-of-the-art, it is crucial to understand how previous works have implemented their solutions for PdM. For that purpose, it is presented the most relevant techniques for feature engineering (Section 2.4), including frameworks for automatic feature engineering (Section 2.5), modeling (Section 2.6), and evaluation methods (Section 2.7).

2.1 Types of Maintenance

Designing and manufacturing dependable systems and high-quality goods is essential to the profitability and competitiveness of the industrial sectors [5]. To achieve this, machines must operate under strict conditions. Therefore, it is crucial to maintain such equipment running in these settings, which encourages ongoing observation and periodic maintenance.

According to the literature, there are many terminologies for maintenance. In this dissertation, the terminologies supported by Susto et al. [70] will be followed. Based on this work, there are three main approaches for conducting maintenance tasks:

- Run-To-Failure (R2F): This method waits for a machine to break, which would stop production, resulting in significant downtime, losses from repairs, and lower productivity.
- Preventive maintenance (PvM): Compared to R2F, regular maintenance expenditures are reduced since breakdowns are prevented rather than waiting for failure to occur. Right now, this is the most widely employed tactic [26].
- Predictive maintenance (PdM): PdM consists of using predictive tools to determine when maintenance actions are required [15]. Such a data-driven approach is different from PvM,

which follows fixed schedules irrespective of equipment condition. PdM leverages data-driven insights to optimize maintenance timing by analyzing sensor data and operational parameters. PdM aims to identify potential issues before they cause failures and avoid unnecessary maintenance when equipment is functioning normally.

2.2 Example Applications

Several case studies across different domains have shown the effectiveness of PdM approaches. For example, in a study by Neeraj Sharma [67], tyre-related data was used to predict the remaining useful life (RUL) of vehicle tyres. The dataset included various tyre information, including brand, pressure, standard specifications, current tread depth, and position on the vehicle. The feature engineering process involved several steps: handling missing values through removal, normalizing the dataset, and performing feature selection using both Spearman and Pearson correlation [37] analyses.

In an industrial case study, Hrnjica and Softic [40] demonstrated the practical application of PdM by analyzing a dataset containing telemetry data, error logs, failure records, and machine specifications. Their system predicted impending component failures, enabling component replacement before actual failure occurred. This approach successfully prevented production interruptions and minimized downtime. The feature engineering process in this study primarily utilized lag-based features, which will be discussed in detail in Section 2.4.3.1.

PdM can be regarded as a particular instance of early anomaly detection tasks, where the goal is to identify potential issues before they lead to system failures. This relationship is exemplified in the work of Cerqueira et al. [17], who applied early anomaly detection techniques to healthcare monitoring. Their study used patient vital signs data, including heart rate, systolic blood pressure, diastolic blood pressure, mean arterial blood pressure, respiration rate, and oxygen saturation, to predict critical health episodes. Their feature engineering approach was based on statistical and wavelet functions [17].

As described above, PdM studies have been conducted in many different areas, from healthcare to the automobile industry, revealing the current importance of PdM across different fields. Moreover, the feature engineering process used does not follow a standard or systematic methodology, being done, most of the time, ad-hoc, as in these examples.

2.3 Irregular Multivariate Time Series

PdM requires continuous monitoring of machine conditions over time. This monitoring is typically done with sensor equipment that collects operational data from various dimensions or machine components. In this context, the data concerning PdM problems are often best represented as a multivariate time series. Each instance of the collected data corresponds to a row in the dataset and has an associated timestamp. This means each row represents a point in time. These multivariate time series are characterized by the presence of multiple variables, each representing a different

aspect of the machine's condition. PdM relies, most of the time, on multiple sensors monitoring the same machine. Accelerometers, displacement sensors, velocity sensors, and torque sensors [25] are just examples of the diverse sensors that can be used. As such, a machine may be equipped with several different sensors that measure, for example, temperature, pressure, and vibration, among other variables.

In the context of PdM tasks, these multivariate time series are usually characterized by two aspects: irregularity [76] and missing values. As seen, irregular means using different sampling intervals and also missing data [83]. In other words, the time units from each sensor or even the trigger events of a single sensor can vary. Sensors may operate independently and collect data at different frequencies or time intervals. This asynchronous data collection creates irregularity in the resulting dataset. Moreover, sensors might fail due to hardware problems, and the communication system between the sensor and the device used to store the values might also fail. In both of these situations, data is lost, introducing missing values. These and other data preparation steps need to be addressed during the feature engineering stage.

In the Table 2.1, an example of this data structure can be observed. This example illustrates multiple features (e.g., temperature, pressure, and vibration), missing values, and irregularity (the distance between every two consecutive timestamps is not constant).

Table 2.1: Irregular Multivariate Time Series Example

Timestamp	Temperature (°C)	Pressure (kPa)	Vibration (mm/s)
2025-05-28 10:00:00	68.4	—	3.2
2025-05-28 10:05:10	68.9	100.8	—
2025-05-28 10:10:22	—	101.0	3.3
2025-05-28 10:12:47	—	101.2	2.9
2025-05-28 10:16:55	70.1	—	3.0
2025-05-28 10:19:40	69.5	100.5	—

2.4 Feature Engineering

It is commonly acknowledged in the literature on machine learning that a key element in creating accurate predictive models is the feature set that is used to represent a data collection [36]. Therefore, feature engineering is an essential phase in machine learning initiatives. This process consists of transforming raw data into a format suitable for machine learning models. The quality of the final representation (the set of features) has a significant impact on the performance of predictive models, being decisive in how well the machine learning algorithms perform.

Feature engineering is a process that involves generating features based on the domain knowledge and expertise [16], and is also time-consuming. However, when dealing with irregular time series, this process can become more challenging due to the temporal aspect of the data and the irregularity in the sampling frequency.

This section shows different approaches to feature engineering applied to PdM. Particularly, we will describe several approaches to deal with: time series irregularity (Section 2.4.1), missing values (Section 2.4.2), feature extraction (Section 2.4.3), feature selection (Section 2.4.5), and variables in different scales (2.4.6).

In Figure 2.1, an overview of the typical feature engineering process in a PdM problem, from the data collection stage until the model-building stage, is presented. During data preparation, several steps can be relevant, depending on the specific dataset.

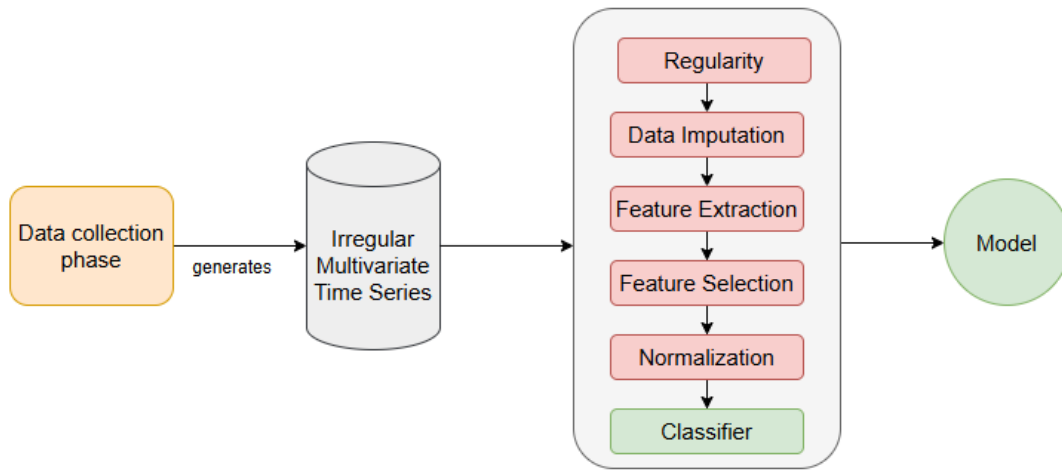


Figure 2.1: Typical feature engineering process for PdM tasks

2.4.1 Dealing with time series irregularity

Regarding the usage of irregular multivariate time series, modeling approaches typically either keep the irregularity in the data or aggregate it into an adequate sampling frequency [38]. Regarding transformation into a regular time series, there are a few techniques: interpolation [38], re-sampling [38], and structural invariance [11].

Interpolation [11, 38, 39, 56] consists of generating new time stamps at regular intervals and then filling the gaps generated by the missing data. This missing data is often computed based on statistical features such as mean, standard deviation, and similar statistics. With that strategy, irregular time series are then transformed into regular time series, reducing the problem to a forecast of regular time series, allowing the use of algorithms specially designed for this problem, such as ARIMA [69], for example.

Re-sampling is done by, most of the time, down-sampling methods, finding a frequency at which the time series is sampled into regular intervals due to a data aggregation process [38]. For a better understanding of this technique, consider a sensor that captures irregular data from a machine each minute. A down-sampling method can then be used (e.g., analysing data in terms of hours instead of minutes) where all the available data from the hour h (timestamps between minute

0 and 59) is aggregated into one single value, using, for example, weighted averages, quantiles, and other statistics. Like so, in the hourly rate, the time series is now regular.

Structural invariance is another technique that consists of removing the temporal context of the time series [11]. Without that context, the irregularity has not been eliminated; it is just ignored, and the dataset represents only points in time, but without relation to each other.

2.4.2 Dealing with missing values

Sensorial data is often associated with missing values, mainly due to sensor failures. Moreover, the implementation of some techniques, such as interpolation or resampling, previously addressed in Section 2.4.1, also needs to deal with missing values for their effective implementation. Missing data is commonplace in PdM tasks. In effect, PdM systems require systematic approaches to deal with this problem.

From the conducted research on feature engineering in PdM, the main strategy adopted consists of replacing the missing values with statistical features such as mean, standard deviation, among others [42]. This technique is also known as interpolation. As was mentioned in 2.4.1, interpolation, besides being used to regularize the time series, also deals with missing values.

In PdM, temporal context is key to understanding when a given machine is going to fail, so other approaches, such as forward propagation and backward propagation, are used to deal with missing data. Forward propagation fills missing values by propagating the last observed value forward in time until a new value is encountered. It assumes that the most recent observation remains valid until a new measurement is recorded [54]. Backward propagation does the opposite: it fills missing values by propagating the next observed value backward in time [54].

Another strategy consists of simply erasing the missing values' instances [67] (typically when missing values are uncommon), or erasing features that are mainly composed of missing values [2, 19].

2.4.3 Feature Extraction

Performing feature extraction is crucial for machine learning since it will allow the introduction of meaningful features into the dataset [81]. Moreover, in this phase, the time series is transformed from a sequence of observations into a tabular format ready for supervised learning. Each instance in this tabular dataset is composed of a set of explanatory variables (Section 2.4.3.1) that summarise the recent dynamics of the data, and the anomaly indicator variable, a binary variable that represents whether an anomaly occurs.

2.4.3.1 Explanatory Variables

In the context of PdM tasks, explanatory variables attempt to summarise recent past dynamics at a given point in time. They provide information to the dataset and try to explain the relationship between the dependent and independent variables [78].

Lag-based Features The introduction of lag-based features generates a sliding window (composed of lags) and contains information that represents past events. Each lag n is a previous timestamp ($t - n$) relative to a specific timestamp t . By selecting the last n lags, some metrics, usually based on statistics such as mean or skewness, can be computed over the explanatory variables, for timestamp t . These introduced values now explain for a given t what happened in the past temporal interval $[t - n, t - 1]$ [13, 24, 40, 82]. This gives the model information about past events and can be crucial, especially when supervised classifiers are used [23]. In Figure 2.2, a visual representation of a lag-based average feature with 3 lags is presented to enhance clarity.

Date	Pressure (kPa)	Pressure_mean
2025-05-01 (t - 3)	101.2	null
2025-05-02 (t - 2)	100.9	null
2025-05-03 (t - 1)	101.4	null
2025-05-04 (t)	101.0	101.17
2025-05-05	100.8	101.10

Figure 2.2: Lag-based feature example (Pressure_mean) using 3 lags

A lag-based feature, by definition, introduces information about past records into the present, so, in general, they are useful to capture trends or seasonality. The lag-based features that do so are also called trend-based features [60]. Consider a machine that must have a given value (e.g., temperature) in a given range of values (e.g, between 70°C and 80°C). If, in a given moment, that value goes off that range, it can indicate a failure (e.g. machine broke due to overheating). Trend-based features intend to register these value fluctuations over time [60].

Moreover, there are other types of features that can be extracted, such as historical count features, that represent the number of some relevant event that happened in a predefined period (e.g. past 24 hours). Suppose a machine that produces samples, each sample produces data, captured by sensors, and the higher the sample number, the more likely the machine is to fail [60]. Historical count features can also be implemented in a cumulative way, the so-called cumulative counters [60].

Another used technique is the wavelet transform. The integration of continuous and discrete wavelet transform with machine learning algorithms has demonstrated promising results in feature extraction over non-stationary data [1]. This technique consists of calculating the wavelet coefficients. This process is computed directly over the continuous signal captured from sensor data. These coefficients provide information about the amplitude of the signal at different scales and translations. In literature, wavelet transforms for feature extraction can be found, as in Cerqueira et al. [17] and Rami Ahmad et al.[1] works.

Time Series Feature Extraction Feature extraction is a crucial and well-known step in the feature engineering process that consists of extracting new features from the existing dataset. This process brings benefits such as dimensionality reduction through a common representation of varied time series, improved interpretability by highlighting fundamental patterns such as trends or seasonality, and enhanced algorithm performance in tasks such as clustering, classification, and forecasting.

Regarding feature extraction, there are several frameworks tailored for time series, such as TSFEL [7], tsfresh [21], and pycatch22 [52]. While other approaches exist, we focus on these three as they are widely used in the literature.

TSFEL, standing for Time Series Feature Extraction Library, receives as input the dataset, and then outputs generated features that give additional information to the temporal, statistical, or spectral domain [7]. Also, TSFEL has been shown to be effective on anomaly detection tasks.

Another feature extraction framework for time series is pycatch22 [52]. This framework computes 22 time series-related features, implemented in C. This can be an advantage in terms of run time, since the C language is faster than others, such as Python [47]. These 22 metrics are inserted in one of the following domains: Distribution shape, extreme event timing, linear autocorrelation structure, non-linear autocorrelation, symbolic, incremental differences, simple forecasting, and self-affine scale. There is still the option to compute two basic statistics: mean and standard deviation [51].

Lastly, tsfresh [21] is another option for feature extraction. This complete framework computes an extensive and complete set of features and is also able to choose the relevant ones. Besides, it is compatible with the scikit-learn library, allowing the user to easily create custom features.

2.4.4 Anomaly Definition

PdM is a problem that consists of predicting if a given machine, or specific components of that machine, are about to fail. The failure can then be represented as a binary variable with values 0 for non-anomalous data and 1 for anomalous observations. Failure data are rare events, which can make the accurate prediction of failures difficult. In other words, PdM can be seen as an instance of an anomaly detection problem. The definition of anomaly in PdM is highly dependent on the domain of application. Such a definition can bear an impact in terms of data preparation, for example, in the regularity of the time series.

Anomalies are often defined based on events occurring in a predefined future period. Recapping one of the examples provided in Section 2.2, Cerqueira et al. [17] considered that an anomaly would be defined as a “30-minute window having at least 90% of its mean arterial blood pressure (MAP) values below 60 mmHg [millimeters of mercury]”. Accordingly, the anomaly was defined as a 30-minute window, and so, the data representation for assessing the anomaly status needs to reflect this structure. In this example, resample was used to do so, achieving 30-minute periods of time.

Another example of this type of anomaly definition is made by Ariel Gorenstein et al. [34], where a key objective of his work is to utilize predicted failure probabilities of system components (of a critical infrastructure) within a specified time interval $[0, T]$ to propose a cost-effective replacement strategy that also considers the spatial distribution of the components. In other words, the anomaly definition is based on time intervals as well, with a predefined horizon T .

One last example of anomaly definitions based on time intervals is presented by Rita Ribeiro et al. [64]. In this article, the main research objective was to predict, with a defined horizon of one week, sequential door operations on trains that contribute to system deterioration. By reducing the downtime of trains' doors, both public and freight transportation become more reliable.

As seen in the examples, the definition of anomaly depends on the context of what is being monitored and requires domain knowledge, in this case, medical (first example) and mechanical engineering (second and third examples) knowledge, and impacts the feature engineering process.

2.4.5 Feature selection

Feature selection is another key step that is frequently included in the feature engineering process. This task involves selecting a subset of features that improve the model's predictive performance. Preventing the issue known as the curse of dimensionality is one benefit of feature selection [3]. Furthermore, by simplifying the model and facilitating quicker training and inference times, feature selection increases computing efficiency. By emphasizing the most important features, it also improves interpretability by making it simpler to comprehend how inputs and outputs relate to one another. In general, there are three types of approaches regarding feature selection: filter, embedded, and wrapper methods.

2.4.5.1 Filter methods

Filter methods, as the name suggests, work by selecting features based on a given rule. To assess the link between features and the target variable, they use statistical measures such as correlation, mutual information, or statistical tests (such as the chi-square test or ANOVA). Despite their computational efficiency, these approaches might miss relevant features by disregarding feature interactions [41].

The literature commonly uses correlation matrices as a filter method [13, 23, 24, 39, 67, 79]. Those matrices represent how similar features are to each other. If two features have a high correlation value, it means that they are similar, meaning that selecting only one of those would reduce the noise of the dataset. These correlation coefficients can be computed by using statistical methods such as the Pearson and Spearman correlations [67].

It is also possible to use a Principal Component Analysis (PCA) to select features. This technique is a statistical method that takes as input the dataset and performs linear combinations of the provided features that explain the variance of all the features in a maximal way, the so-called components. Then the components are used instead of the original features [35]. In PdM, PCA is also used [1, 49], even though it has some limitations, such as only being able to detect linear

relations between features. If data has non-linear correlated features, autoencoders can be used instead [31]. Autoencoders are explained in more detail in Section 2.6.1.2.

2.4.5.2 Embedded methods

In embedded methods, features are automatically ranked or chosen by algorithms such as Random Forest, Lasso regression, and decision trees according to how well they contribute to the model's performance. These techniques incorporate feature selection straight into the learning algorithm, striking a balance between accuracy and computing efficiency [41].

A typically used method of performing feature selection, under the context of PdM, is using the Random Forest algorithm [50], which automatically assigns given importance to each feature while training the model. Then, those values can be used to rank the features by importance, and by selecting only the best ones, the input becomes more focused and potentially more predictive [14, 23].

2.4.5.3 Wrapper methods

By iteratively evaluating feature subsets produced by a predetermined search strategy, wrapper methods use a modeling algorithm (such as K-means for clustering tasks and algorithms such as Naïve Bayes or SVM for classification tasks) as a “black box” to evaluate performance [41]. Because evaluation is based on actual model performance, wrappers frequently yield better-performing feature subsets than filter approaches. However, because the model must be repeatedly trained and tested for each subset, they are computationally costly. Furthermore, the subsets that are chosen are frequently skewed in favor of the modeling approach that was employed for the assessment. It is crucial to validate the final subset using separate data and perhaps a different model in order to lessen this [41].

2.4.6 Normalization

Normalization is also a popular step in the feature engineering process. Normalization consists of transforming the features into the same scale (e.g., between 0 and 1). This can significantly improve the overall performance of machine learning models, and it can also be employed for PdM tasks. Examples include mix-max normalization [39] or standardization [67]. While working with machine learning algorithms that internally use a distance metric, such as k-nearest neighbors (KNN), normalization has a significant impact on the performance of the algorithm [63].

2.5 Automatic feature engineering

Automatic feature engineering consists of automating the feature engineering process using a set of techniques such as the ones presented during Section 2.4. With such automation, it is possible to transform raw data into data ready for supervised learning with minimal user input.

Professionals frequently lack the necessary time series analysis skills even when they have topic expertise [72]. Implementing the most advanced time series forecasting techniques is a difficult and time-consuming process that requires a great deal of skill. In this regard, creating frameworks for automatically extracting an appropriate representation from an input time series can benefit practitioners with little technical knowledge [16]. Regarding automatic feature engineering for PDM, some frameworks have already been developed.

As an example, in their work, Cerqueira et al. [16] developed an automatic feature engineering approach called VEST (Vector of Statistics from Time series). The process involved transforming time series into several distinct representations. Each representation was then summarized using statistical measures such as mean and standard deviation, among others, and finally, a feature selection procedure was performed to avoid the curse of dimensionality. By establishing a well-described pipeline, it is possible to have an automatic feature engineering process that can be applied to many forecasting problems.

In the work of Danilo Giordano et al. [33], a case study was performed to predict when the oxygen sensor of an internal combustion engine was going to be clogged, which leads to imprecise measures of that sensor. A comprehensive pipeline called PREPIPE was used to automate the full feature engineering process, addressing topics such as signal selection, feature extraction, and selection [33]. This framework is, among all the others mentioned in this state-of-the-art, the most similar to the one that was developed in this dissertation, since both automate the feature engineering process under the PDM context.

AutoLearn [44] is a domain-independent framework for supervised learning that automates the processes of feature generation and selection. At its core, AutoLearn uses a regression-based feature learning algorithm to transform the original features into a new, more informative set. This is achieved by applying regression models to identify and model relationships among the original features, effectively creating new features that capture underlying patterns in the data. These learned features are then evaluated and selected based on their predictive utility for the target task, allowing AutoLearn to improve model performance without manual feature engineering.

ExploreKit [43] is also a framework for automated feature generation and selection. Similarly to AutoLearn, this framework also computes, based on the original features, other features called candidate features. The innovation of ExploreKit resides in the ML-based feature selection step they present, capable of predicting, from the previously computed candidate features, which ones would perform better, diminishing the total number of features and avoiding the curse of dimensionality.

Cognito [46] is another framework for automated feature engineering that consists of applying different transformations over the dataset and storing the resultant dataset in a tree data structure. These datasets are then tested, being the consequent results recorded. Then a search over the tree is done to see the best results.

However, these last three frameworks are not tailored for time series specifically, but for tabular data. As such, transformations do not take into account the temporal nature of the time series.

2.6 Modeling

PdM can be seen as an anomaly detection problem, more specifically, a binary classification problem where the label, that is, the target variable that is being predicted, can have a binary value, representing whether the machine failed or not. Moreover, this target variable has an imbalanced distribution since anomalies are rare events.

2.6.1 Learning approaches

This section provides a concise overview of several widely used machine learning algorithms, including their applications and key characteristics. There are two main types of learning approaches used for PdM: supervised and unsupervised learning. Supervised learning consists of finding patterns based on labeled data. In this kind of approach, a discriminative function is used between inputs (explanatory variables) and outputs (the target variable or label). On the other hand, unsupervised approaches try to find patterns and the structure without reference to the target variable [62].

2.6.1.1 Supervised Learning Approaches

Supervised learning involves the process of training a model to learn a mapping between a set of input variables X (features) and an output variable Y (target) [22]. This is accomplished by utilizing a dataset that has the appropriate input-output pairings (labeled data). By reducing the error between its predictions and the actual outputs during the training phase, the model improves the mapping function. After training, the model may effectively generalize to new instances by using this mapping to predict the output for unknown data. In this work, we will focus on classification-based approaches, specifically those where the target variable is binary.

A decision tree is a supervised learning algorithm that splits data based on feature values to create increasingly pure subsets regarding the target variable. This process continues until the stopping criteria are met, for example, maximum tree depth [53].

Random forest is an ensemble learning method that combines multiple decision trees to improve the accuracy of predictions [66]. In the context of PdM, both decision trees, with CART [14, 42] or with other approaches [33, 40, 67, 79], and random forest [2, 12–14, 17, 24, 33, 58, 82] are widely used. Boosting algorithms are ensemble methods that iteratively train weak learners, typically decision trees, to produce a strong predictive model. Under the PdM context, boost algorithms such as Light Gradient Boosting Machine [12, 17, 23, 40, 67], xgboost [12], boosted trees [19, 79] were used with the intent of enhancing the results produced, mainly by decision trees.

Neural networks are also widely used in PdM tasks, including distinct architectures such as convolutional neural networks [1, 6, 27, 48, 57], dense neural networks [42], and long-short-term memory neural networks (LSTM) [1, 6, 8, 11, 27, 39, 48, 56, 57, 65]. Other learning algorithms have also been explored for PdM, such as support vector machines [71], or K-nearest neighbors [23, 25, 66, 67].

2.6.1.2 Unsupervised Learning Approaches

Unsupervised learning approaches model the input data without reference to output labels [74]. These methods can also be used to tackle PdM problems. This learning approach consists of modeling the input data, typically the non-anomalous ones, and trying to reconstruct them. The reconstruction error indicates the level of anomaly [31].

One popular unsupervised and reconstruction-based method in the literature is the autoencoder [4]. Autoencoders are a type of neural network architecture that aims to compress (encode) input data into a low-dimensional representation, and then reconstruct it back to its original structure. The overall goal is to make the neural network learn the core patterns of the input data. The intuition is that anomalous observations will deviate significantly from those core patterns, resulting in large reconstruction errors that indicate potential anomalies. While unsupervised learning techniques are well-established for anomaly detection, in this work we will focus on supervised approaches.

2.6.2 Imbalance problem

In classification problems, classes represent the possible output labels or categories that the model can predict. The imbalance problem occurs when data contains a highly uneven distribution of these classes, meaning that there are majority classes, which have the vast majority of data samples, and minority classes, which have only a very small fraction of samples.

As discussed in Section 2.4.4, PdM can be seen as a binary classification problem, meaning there is only one majority and one minority class. In PdM, most of the data represents non-failures, while only a few percentage of the datasets are data of machine failures [13] since they are rare events.

This class distribution imbalance poses additional challenges for the adequate modeling and generalization of predictive models [10]. In classification approaches, the model tries to find patterns and features that characterize, as accurately as possible, the classes it aims to predict. When there is a discrepancy over the percentage of records of the non-anomalous class and the anomalous class, as it happens in PdM, the model lacks information about the minority class, thus complicating the learning process.

To solve this problem, the most common approach is to transform the training data by changing the target variable distribution. This can be done either by down-sampling on the non-failure class (i.e., reducing the number of examples from the majority class to balance the dataset) [2], or up-sampling of the failure class (i.e., increasing the number of examples in the minority class by duplicating or synthetically generating new samples), can be done with methods such as SMOTE [14].

2.7 Evaluation

In this section, we describe the typical evaluation process applied in the context of PdM tasks.

2.7.1 Evaluation Metrics

To evaluate the performance of PdM models, several classification metrics are commonly used: precision, recall, F1-score, and AUC (Area Under the ROC Curve). The AUC is derived from the predicted probabilities of the positive class output by the model, while the other metrics are derived from the confusion matrix, which is a table that summarizes the prediction results by comparing predicted labels to actual labels [30]. The confusion matrix includes:

- **True Positives (TP):** Correctly predicted anomalies (failures).
- **True Negatives (TN):** Correctly predicted normal states.
- **False Positives (FP):** Incorrectly predicted anomalies (false alarms).
- **False Negatives (FN):** Missed anomalies (failures that were not predicted).

Precision is the ratio of true positives to the total predicted positives ($TP / (TP + FP)$). It shows the frequency with which the model accurately forecasts a failure [4]. Recall is defined as the ratio of true positives to the total actual positives ($TP / (TP + FN)$). It measures the model's ability to detect actual failures [4]. F1-score is the harmonic mean of precision and recall ($2 * (Precision * Recall) / (Precision + Recall)$), being a balanced statistic that takes into consideration both false positives and false negatives [4].

The metric known as AUC, or more precisely AUC-ROC (Area Under the Receiver Operating Characteristic Curve), assesses a model's capacity to discriminate between classes at various threshold values. The ROC curve is generated by plotting the true positive rate (recall) against the false positive rate ($FP / (FP + TN)$). Excellent discrimination ability is demonstrated by a model with an AUC around 1, whereas random guessing is indicated by an AUC of 0.5. Because it evaluates performance across the whole range of classification thresholds, AUC is especially useful in PdM settings where the cost of false positives and false negatives may vary [20, 28].

In addition, an important metric that can be taken into account is the remaining useful life (RUL) [3, 24, 33, 75]. The RUL of a machine is the time left before it reaches a safe operational limit. By predicting this statistic, it is possible to see how close the equipment is to failure, as exemplified by Dhiman et al. [24].

2.7.2 Cross-Validation

Cross-validation is a frequent technique used for evaluating predictive models' performance by testing the model on different subsets of the data. To ensure the reliability of the results, the entire modeling process, including the feature engineering process and model training, must occur within each cross-validation fold. This ensures that no information from the test data leaks into the training process [77]. Cross-validation is widely used in literature to estimate predictive performance [2, 12, 17, 33, 42].

When dealing with cross-validation and temporal data, it is necessary to take into account the temporal order of the records while doing the partitions in training and validation sets, ensuring no future data is used to predict past data [18].

2.8 Conclusions

In this section, several techniques were presented regarding feature engineering, modeling, and evaluation of machine learning algorithms under the context of PdM. Remembering the scope of this dissertation, the goal is to develop an automatic feature engineering framework, taking into account problems such as time series irregularity and missing data, feature extraction, feature selection, and normalization of the data. The basis of the framework development consists of techniques described in this chapter.

As seen, PdM is far from a trivial problem. Anomaly events are scarce records, making it difficult to accurately predict the desired failures. Furthermore, datasets are prone to having multiple data problems, such as missing data and irregularity. All these issues make PdM a difficult task. The feature engineering process, under the PdM context, is time-consuming, and domain knowledge is required to extract relevant features and patterns since the anomaly definition is highly dependent on the underlying domain. In effect, there is a need to automate the feature engineering process.

So far, in literature, there are already some frameworks that try to do that, by automating feature extraction [16], and feature selection [3, 16] in time series or even AutoLearn [44] or ExploreKit [43], more suitable for tabular, non-temporal data, but there is a lack of frameworks that automates the entire feature engineering process specifically for PdM problems. The work of this dissertation tackles that gap by proposing a novel approach to achieve this.

Chapter 3

Methodology

This chapter presents the proposed approach for automating the feature engineering process in PdM tasks. First, the feature engineering process problems are defined, with an emphasis on supervised learning techniques. Afterwards, an introduction to the developed pipeline, which aims to automate the feature engineering process, is given. The created transformations for that endeavor are later detailed, culminating the pipeline description with the modeling part. Finally, an entire section will address the optimizations performed over the pipeline, in terms of fine-tuning.

3.1 Problem Definition

PdM aims to predict when a given machine, or its components, are about to fail. The collected sensorial data, as explained in the previous chapter, can be represented as an irregular multivariate time series. These datasets encompass some challenges that need to be addressed during the feature engineering stage, such as irregularity, missing values, feature extraction, feature selection, and normalization.

Let $\mathbf{P} = \{\mathbf{p}_t\}_{t=1}^T$ be a multivariate time series, where $\mathbf{p}_t \in \mathbf{R}^w$ represents the w -dimensional feature vector collected at time t , and T is the total number of observed time points. In the context of PdM, $\mathbf{P}$ often exhibits irregular sampling intervals $\Delta t_i = t_{i+1} - t_i$, where Δt_i varies across time points i . Additionally, missing observations in certain dimensions of $\mathbf{p}_t$ result in incomplete data, requiring imputation strategies or methods robust to missing values.

The goal is to transform $\mathbf{P}$ into a structured representation $\mathbf{Z} \in \mathbf{R}^{T \times u}$, where u denotes the number of selected or extracted features. It is also important to mention that $y \in \mathbf{Z}$, being y the target variable, which defines the anomaly occurrence. This allows the model to be trained using supervised learning algorithms for binary classification. Also, $\mathbf{X} \in \mathbf{Z}$, being $\mathbf{X}$ the set of explanatory variables. All variables in $\mathbf{X}$ are numerical. The transformation described above must preserve the temporal dependencies and relevant patterns necessary for failure prediction while addressing challenges such as:

- **Irregularity:** Irregular sampling Δt_i arises from asynchronous sensor readings or system delays, complicating temporal analysis. Regularizing the time series can have a positive impact on models' predictive performance.
- **Missing Values:** Sensor data often suffers from missing entries due to malfunctions, communication errors, or power failures. Handling these gaps involves selecting appropriate imputation strategies $\mathcal{I}(\mathbf{P})$.
- **Feature Extraction:** Raw time series data may not be informative enough for predictive models. Feature extraction aims to derive higher-level representations $\phi : \mathbf{P} \rightarrow \mathbf{R}^u$ that encode meaningful characteristics.
- **Feature Selection:** With potentially hundreds of extracted features, it is essential to identify the most informative subset $\mathcal{S} \subseteq \{1, \dots, u\}$. This step reduces overfitting, enhances interpretability, and lowers computational costs.
- **Normalization:** Differences in units and scales across features can negatively impact model convergence and accuracy. Normalization techniques ensure that features contribute proportionally to the learning process. This step is especially important when using distance-based algorithms.

We refer the reader to the previous chapter for a more detailed read on these challenges. This project is focused on supervised approaches for PdM and aims to study the feature engineering process and how to automate it to maximize supervised learning approaches' performance.

3.2 Automatic Feature Engineering

This work aims to create a framework that can automate different parts of the feature engineering process, namely transforming time series to regular frequency, dealing with missing values, feature extraction, feature selection, and normalization, and to develop an optimized model for anomaly detection¹.

Our approach is based on defining a pipeline, responsible for transforming the raw data into a trained model. This pipeline sequentially addresses the previously mentioned feature engineering tasks, in that order. The last step of the developed pipeline is the training of a model with the transformed data. As an example, one possible pipeline would consist of (1) using resampling to regularize the time series, (2) using linear interpolation as the data imputation strategy, (3) performing feature extraction with TSFEL and window size of three lags, (4) performing feature selection based on feature correlation, (5) using standard scaling to normalize the dataset and (6) train a binary classification model.

However, for each task (e.g., missing data imputation), different techniques can be applied, each of which has several possible configurations. This results in a large configuration space. As

¹optimusPipe package: <https://github.com/JoseAlbanoGaspar/PdM-TS-framework>

such, there is a need to add another stage to the framework, the optimization stage. This is done through search strategies such as grid search, which systematically explores the space of pipeline configurations to maximize predictive performance, as explained in detail by Jay I. Myung et al. [55]. The optimization of the model also occurs here, together with the feature engineering optimization, as the training of the model is also a pipeline step. The training of the model at the end of the pipeline is crucial for identifying the best parameters, since it allows us to measure the pipeline’s performance by using, in this case, the AUC statistic. Figure 3.1 presents an overview of the developed framework.

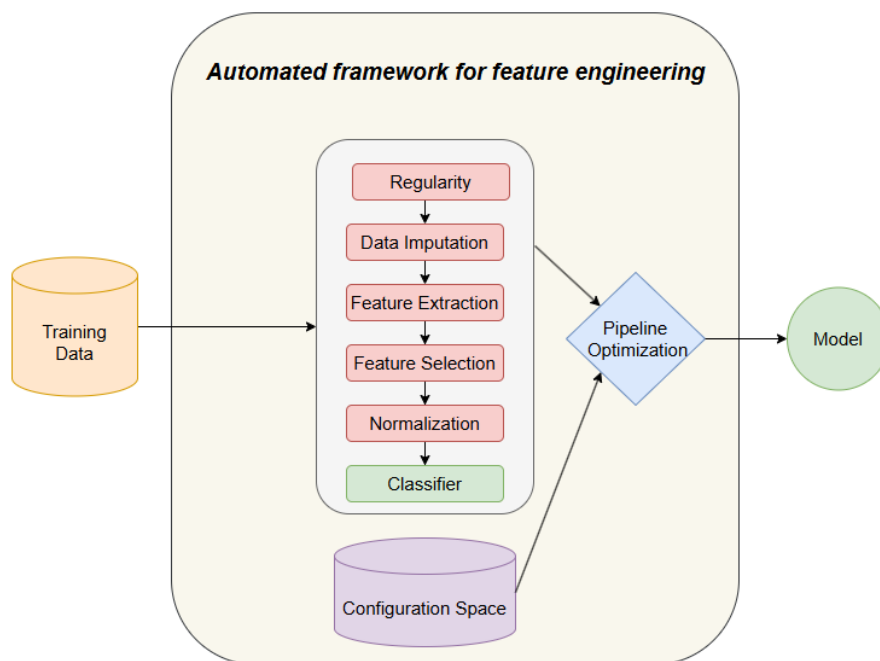


Figure 3.1: Proposed framework to automate feature engineering

3.2.1 Pipeline

The proposed pipeline addresses, sequentially and in an automated way, a set of typical problems on the feature engineering process, culminating in a classification step as shown in Figure 3.2

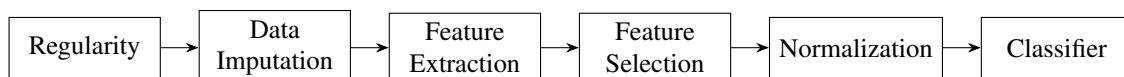


Figure 3.2: Processing steps supported by pipeline in sequential order

The strategy behind this pipeline is the expand-reduce technique. In this approach, the “expand” phase is responsible for generating a large number of candidate features from the raw time

series data across different domains, such as statistical, fractal, and temporal domains, among others. To perform such tasks, state-of-the-art feature extractors such as TSFEL [7], tsfresh [21], and pycatch22 [52] can be used. However, not all of these generated features are equally informative or relevant for the prediction task. As such, it is necessary to select, from the candidate features, the most relevant ones. This is done on the “reduce” phase, where, by applying feature selection strategies such as correlation-based feature selection techniques, only the most relevant features are kept. By combining these two phases—first expanding to capture as much information as possible, and then reducing to retain only the most predictive components—the pipeline balances completeness with efficiency. This strategy is particularly well-suited to automated feature engineering in PdM, where the underlying failure patterns may be subtle and distributed across different temporal resolutions. This approach has been effectively applied in previous works, e.g. [45, 59].

With the defined pipeline structure, it becomes possible to transform the input data automatically into a trained model. Note that the training of the classifier inside the pipeline (last step of the pipeline defined on Figure 3.2) is crucial since it will evaluate the transformations that occurred during the pipeline execution, using the AUC statistic. This statistic will later be used in the optimization stage to distinguish the optimal pipeline.

3.2.2 Transformations

Each step of the pipeline tackles one problem using one or more approaches, chosen based on common techniques found in the literature as described in the state-of-the-art. In the next sections, the implemented approaches, per step, are detailed. A summary table is presented in Table 3.1.

Table 3.1: Implemented transformations in each pipeline step

Pipeline Step	Transformations
Regularity	Resampling, Irregular
Data Imputation	Interpolation (linear, polynomial), ffill
Feature Extraction	lag-based features with TSFEL, tsfresh and pycatch22
Feature Selection	Correlation threshold, PCA
Normalization	Min-Max normalization, Standardization

3.2.2.1 Regularity

To deal with irregular time series, a resampling strategy was implemented. This strategy regularizes time series at a given frequency, given to the transformer as a parameter. This operation introduces new rows in the dataset, associated with the generated regular timestamps. As such, it is worth mentioning that this transformer operates over both $\mathbf{X}$ and y , $\mathbf{X}$ being the explanatory variables and y the target variable, since it adds new rows to the dataset.

Another strategy is to model the data directly using its irregular periodicity. In other words, no regularization method is applied.

3.2.2.2 Data imputation

Regarding imputation, two strategies were implemented: interpolation and forward fill. These imputation strategies occur over $\mathbf{X}$, the set of explanatory variables. However, as described in Section 3.2.2.1, the implemented regularity strategies can introduce new timestamps in $\mathbf{Z} = \{\mathbf{X}, y\}$. It is necessary to take into account the imputation of the target column, y , propagating either 0 (no failure) or 1 (failure) based on the closest original value.

3.2.2.3 Feature extraction

Lag-based features The base idea behind the transformers for feature extraction is the creation of sliding windows, which consist of the last n records on the dataset that occur immediately before the timestamp t . With those n values from the windows, from timestamps $t - 1$ until $t - n$, many statistics can be computed, such as mean, standard deviation, etc. So, for each timestamp t presented on the dataset, we pick the corresponding sliding window for a given feature f , and we compute several statistics for time t . These statistics generate other features in the dataset, derived from f , for example, f_mean , f_std , etc, the so-called lag-based features.

The question now is, which metrics should be computed? As mentioned in Section 2.4.3.1, there are already feature extractors that compute many features, so three of them were chosen: TSFEL [7], tsfresh [21], and pycatch22 [52]. As such, three transformers were created, one for each feature extractor, using the previously described idea of using sliding windows and then using each feature extractor to generate the lag-based features.

Preliminary feature selection Since some feature extractors computed a lot of features, $\mathbf{M}$, drastically worsening the performance of the code, an optimization was done to select the top j most important features of $\mathbf{M}$, called $\mathbf{m}$, where $\mathbf{m} \in \mathbf{M}$. Also, since the feature extractors were computing lag-based features over all numeric features of the dataset, $\mathbf{X}$, we also selected the top k most important features from the dataset, $\mathbf{F}$, where $\mathbf{F} \in \mathbf{X}$. That way, instead of having $\mathbf{M} \times \mathbf{X}$ combinations, there are only $\mathbf{m} \times \mathbf{F}$ combinations, making the code faster, yet not compromising the results due to the way this enhancement was implemented. To perform such optimization, the following steps were carried out:

1. Take a small percentage (10%) of the dataset for faster results.
2. The LightGBM model was trained with this data, and then the feature importance was checked to select the top k features - $\mathbf{F}$.
3. Then, for TSFEL and tsfresh, a pipeline was created and executed (containing linear interpolation for imputation, TSFEL and tsfresh, respectively, for feature extraction, and correlation as the feature selection strategy) over the set of features $\mathbf{F}$. This results in a dataset ready to be trained by a supervised learning algorithm.
4. Train another LightGBM with the data obtained from the previous step, and check the feature importance, taking into account only the created ones.

5. Generate the extractor configuration containing the subset of the most important features - $\mathbf{m}$. This subset is derived from all the features a given feature extractor can compute - $\mathbf{M}$. This information, $\mathbf{m}$, is passed to the feature extractor transformer, making it compute the $\mathbf{m} \times \mathbf{F}$ combinations instead of the $\mathbf{M} \times \mathbf{X}$.

In Figure 3.3, a diagram containing this optimization step is presented.

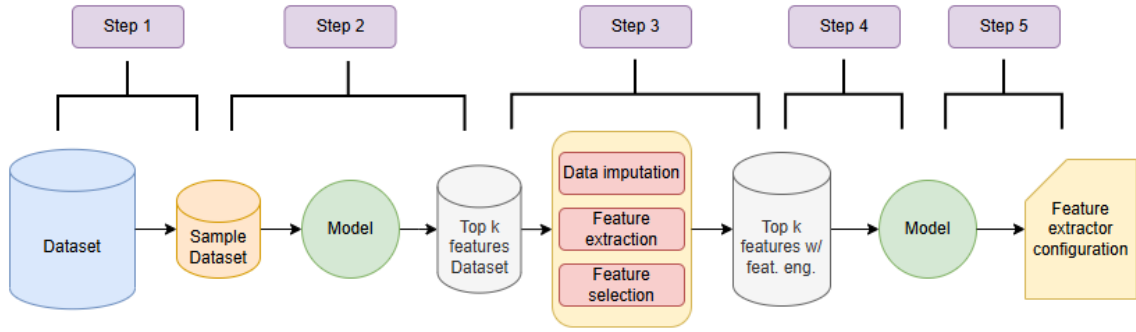


Figure 3.3: Optimization conducted on the feature extraction step

3.2.2.4 Feature selection

For feature selection, two complementary approaches were implemented: correlation-based filtering [37] and PCA [35]. The correlation approach uses a threshold to identify highly correlated features. A correlation matrix is computed using the Pearson correlation coefficient, and when features exceed the threshold, only one representative feature is retained to eliminate redundancy.

For PCA, another threshold is given to control the trade-off between dimensionality reduction and information loss. A lower threshold reduces dimensions but may lose information. Since PCA does not handle missing values, and some values computed by feature extractors were null values or infinity, it was ensured, during the feature extraction step, that those missing values were filled properly. The imputation strategy used in this part was to compute the mean of the column.

3.2.2.5 Normalization

Normalization is also an important step to take into account during the feature engineering process and was proven to improve performance when ML models that use some distance metric internally, such as KNN, are used [63]. For normalization, techniques such as min-max normalization (rescales data to a fixed range - $[0, 1]$) or standardization (centers data with mean 0 and standard deviation 1) can be used. These techniques have been applied in several works (e.g [39, 67]) under the PdM context.

3.2.3 Modeling

As already mentioned, the developed pipeline ended with a learning algorithm for probabilistic binary classification. The model, similar to what happens on the other pipeline steps, which correspond to the feature engineering process, can also be optimized. Taking LightGBM [68] as an example, the depth of its trees and the number of estimators are just two examples of parameters that can be used to optimize the model. As such, the model optimization occurs at the same time as the feature engineering optimization, during the optimization stage of the framework. More details of this stage will be provided in the next Section 3.2.4.

In addition, in PdM, most of the time, there is an imbalance problem between the anomaly class and the non-anomalous class. Like so, the score used by this model was AUC, which indicates the discriminative capacity between both classes, allowing a comprehensive study of the model's performance.

Furthermore, it is worth mentioning that the framework shows flexibility in the choice of model and its configuration parameters. In other words, the model used to evaluate pipeline operations is freely selected and customizable, since it outputs class probabilities. This is linked with the AUC usage that requires those probabilities to be computed.

3.2.4 Pipeline Optimization

Having established the pipeline implementation and evaluation framework, the next step involves optimizing the parameter configuration. As part of this optimization stage, search strategies are employed to explore and identify the best combinations of parameters.

As demonstrated throughout the Section 3.2.2, transformers can have several parameters: regularity transformer needs a regularity frequency, imputation needs the strategy as argument and strategies such as interpolation also need arguments (linear, polynomial), feature extraction needs the feature extractor to use and its configuration and feature selection needs the strategy and the corresponding parameters. Moreover, the model itself is also parameterizable. This generates a parameterizable space, and it is important to understand which combination of parameters leads to better results.

To achieve this, a search strategy needed to be applied to the entire pipeline and its parameters. The framework supports multiple search strategies, like random search [80] and grid search [55], together with time series cross-validation.

Time series cross-validation is a technique tailored for temporal data, where the order of the observations is preserved. This technique creates, as such, training and validation folds based on that temporal order, ensuring that future data is never used to predict the past. In the Table 3.2, the space configuration is presented.

Table 3.2: Configuration space

Pipeline Step	Parameter	Values
Regularization	resampling	frequency = [30min, 1h, 2h]
	irregular	-
Imputation	interpolate (linear)	-
	interpolate (polynomial)	degree = 2
	interpolate (spline)	degree = 2
	ffill	-
Feature Extraction	n_lags	3, 5, 8
	feature extractor	TSFEL, tsfresh, pycatch22
Feature Selection	strategy	correlation, pca
	threshold	0.85, 0.9, 0.95
Normalization	strategy	standardization, MinMax
Classifier (LightGBM)	max_depth	3, 5, 7
	n_estimators	25, 50, 100

Chapter 4

Case Study

This chapter presents a case study analyzing real operational data from a pharmaceutical company’s pill compression machine. The dataset contains production data collected during the manufacturing process of pharmaceutical pills, providing an opportunity to evaluate our methodology on a real-world PdM industrial application.

This chapter is organized as follows: Section 4.1 provides background context for the case study. Section 4.2 describes the dataset in detail. Section 4.3 explains the pre-processing steps used to transform the raw data into a suitable input for our framework. Finally, Section 4.4 formally defines the specific problem addressed in this case study.

4.1 Context

The data consists of sequential production batches, where each batch represents the manufacturing process of a specific drug. During production, raw materials are compressed into pharmaceutical pills using specialized machinery. Quality control is maintained by sampling events that occur approximately every 30 minutes. During these events, sample pills are collected and analyzed across multiple quality parameters, including thickness, weight, and hardness, to ensure they meet manufacturing specifications.

If the sample analysis reveals that quality parameters deviate from acceptable ranges (which vary across different drugs), the system triggers an anomaly. When this occurs, the production process must be halted and the machine stopped, triggering a failure event that is logged with a specific alarm code. These production interruptions can significantly impact manufacturing efficiency and output.

The primary goal of this case study is to develop a predictive model that can anticipate these anomalous events before they occur, allowing preventive action to be taken before the machine enters a failure state. To achieve this, we apply the automated feature engineering methodology detailed in the previous chapter. Our framework systematically handles the entire feature engineering process and the training of an optimized model for anomaly detection.

4.2 Dataset

This dataset is divided into several tables, as described below:

- **Diameter** - Has information about the diameter of the pill, in millimeters.
- **Hardness** - Has information about the force applied in the compression process in newtons.
- **Thickness** - Has information about the thickness of the pill, also in millimeters.
- **Weight** - Has information about the weight of the pill, in grams.
- **Report** - Has summarized information about the previous tables, such as the maximum and minimum weight of pills, the mean and standard deviation for weight, thickness, hardness, diameter, as well as other domain-related metrics.
- **Parameter** - This table has information about the parameters of the machine. These parameters are monitored automatically during the sampling events. These values represent the real values that the machine has during that time. Some of those parameters are penetration depth of main compression, permissible load, in Newtons, on the compression process, among other domain-related metrics.
- **Setup** - This table defines that the parameter a must have the value b for the production of the pill c , and it is set up by the machine operators before the beginning of the production phase of a given pill. These values are the expected parameter values, and the difference from these values to the ones captured on the table **Parameter** can be responsible for triggering some of the events presented on the **Event** table. Those events are triggered alarms related to a machine malfunction.
- **Event** - Has information about the events that led to failures and indicates the time when those failures happened. There are 189 different failure types. However, only 5 of them require the machine to stop.

Those tables represent several points in time (time series), and for each timestamp, a set of pill samples is provided. For tables **Diameter**, **Hardness**, and **Thickness**, their batch, per timestamp, consists of 10 samples of pills. For **Weight**, there are 20 samples. Also, there is information equipment (an ID), and the current quantity produced.

As it is detailed in Section 3.1, the input data for the framework is a multivariate time series composed of explanatory variables $\mathbf{X}$, and target variable y .

However, as seen, there were 8 tables. Like so, a pre-processing step was required, where those tables were merged. First, those tables needed to be treated following some feature engineering conventions such as removing columns with only a single value, removing duplicated entries, converting timestamps into only one datatype to achieve consistency on merges, and formatting some values (e.g, transforming 2,1 into 2.1).

As previously mentioned, the dataset is divided into batches. Each batch is, formally, a time series representing what happened to the machine during the production of a given drug. As such, this dataset can be defined as a multiple irregular multivariate time series, and the `optimusPipe` package supports such input by addressing each batch (each time series) individually.

4.3 Data pre-processing

So far, we have a single table that has all the explanatory variables $\mathbf{X}$, represented by the columns of the previously mentioned table. To make this dataset suitable to be used by the framework, only the target variable y needs to be added.

After that, a selection of the important alarms was conducted after a discussion with domain experts. There were a lot of different alarm codes, each one representing a different problem with the machine. However, only five were considered critical enough to force the machine to stop. Since one of the main objectives of PdM is to minimize downtime, these were the chosen alarms that the model should be able to recognize. The next step was to merge the tables altogether.

To incorporate the target variable y , a time window of 30 minutes was defined. For each timestamp t in the merged dataset, the framework checked whether any failure occurred within the interval $[t, t + 30]$ in the **Event** table. If at least one failure was found during that interval, the value of y was set to 1; otherwise, it was set to 0. In this way, a binary target variable was constructed.

This process effectively defines what constitutes an anomaly in the context of this case study: any timestamp t that precedes a failure event within the following 30 minutes is labeled as anomalous. This explicit definition aligns with the framework's goal of identifying potential failures before they cause unplanned downtime, as introduced in Section 2.4.4. As a result, the dataset now meets all the necessary criteria to serve as input for the `optimusPipe` framework.

The final dataset had 213 columns, of which 210 were explanatory variables and the other 3 were the target variable, the id of the batch, and the timestamp, and 33411 rows. The timespan goes from 2022-01-10 to 2024-02-09.

4.4 Problem definition

The predictive maintenance (PdM) problem in this case study represents a specific instance of the general problem defined in Section 3.1. While our framework was designed to handle multiple requirements, including data irregularity, missing values, feature extraction, feature selection, and normalization, along with model training, this particular case study only requires a subset of these capabilities. Specifically, we found that neither data irregularity handling nor normalization was necessary for this application.

Regarding irregularity, we model the dataset without conducting resampling. As mentioned before, each observation denotes a sampling event. In effect, we keep this granularity for creating explanatory variables.

Additionally, data normalization was not required since we employed tree-based models rather than distance-based models in our framework (described in chapter 3). Tree-based models are robust to varying scales in individual features, making normalization unnecessary for optimal model performance.

Based on the analysis of the case study requirements, the framework workflow was adapted to include the following steps:

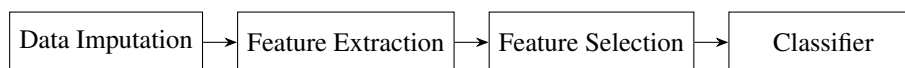


Figure 4.1: Used processing steps of the pipeline in sequential order

4.5 Pipeline execution

In this case study, LightGBM [68] was chosen as the model to be used. This ML algorithm is a tree-based algorithm, and the main reason for choosing it over other models was its fast performance.

After defining the steps of the pipeline required for this case study, we define the configuration space of the framework (also represented in the Figure 3.1). The configuration space for the pipeline is defined across four main stages: imputation, feature extraction, feature selection, and classification. The parameters explored during the optimization process are described below:

1. Imputation Step

Despite all of the performance optimization implemented on the framework, in the imputation step, only forward fill was used, in the interest of computational efficiency.

2. Feature Extraction Step

In this step, 3 feature extraction libraries were used, each with specific configurations:

- **TSFEL** using a predefined subset of features that the feature extractor needs to compute.
- **tsfresh**, using a set of predefined feature calculators, similar to what happens with TSFEL.
- **pycatch22**, using its default fast execution without additional parameters.

Additionally, different lag window sizes were tested with values of 3, 5, and 8.

3. Feature Selection Step

Two strategies were evaluated for selecting relevant features:

- Correlation-based selection
- Principal Component Analysis (PCA)

For both strategies, three threshold values were tested: 0.85, 0.90, and 0.95. In correlation-based selection, this threshold represents the absolute correlation coefficient above which one of a pair of highly correlated features is removed to reduce redundancy. For PCA, the threshold corresponds to the cumulative explained variance ratio, indicating the proportion of total variance retained by selecting the top principal components.

4. Classification Step

For the classification stage, a LightGBM [68] was used, and the following hyperparameters were optimized:

- Maximum depth of the base estimator: 3, 5, and 7
- Number of trees (estimators): 25, 50, and 100

Regarding the posterior optimization stage, the grid search strategy was used. By brute forcing all the possible combinations, it is guaranteed that the best results possible were obtained. Moreover, with grid search, it is possible to measure the importance of each step, on average, of the feature engineering process, which is one of the objectives of this dissertation, justifying the search strategy choice. Grid search works with cross-validation. A time series cross-validation approach with five folds was used, as previously mentioned in chapter 3.

In Table 4.1, the configuration space used in this case study is detailed.

Table 4.1: Configuration space for case study

Pipeline Step	Parameter	Values
Imputation	ffill	-
Feature Extraction	n_lags	3, 5, 8
	feature extractor	TSFEL, tsfresh, pycatch22
Feature Selection	strategy	correlation, pca
	threshold	0.85, 0.9, 0.95
Classifier (LightGBM)	max_depth	3, 5, 7
	n_estimators	25, 50, 100

Chapter 5

Experiments

This chapter is split into three main parts: the experimental design Section 5.1, where the evaluation format of the framework is detailed, the following sections that address the results, and a last section that presents a discussion about the developed work. The results include an operation performance Section 5.2, where the average performance of each step of the pipeline is measured, a best performance Section 5.3, where the best pipeline is presented, a comparative results Section 5.4, where the best pipeline is compared with different baselines and pipelines containing different models, and a dedicated Section 5.5 for results related with feature extractors, namely related with feature importance and execution time. This chapter ends with a discussion about the strengths and limitations of the developed work, in the Section 5.6.

5.1 Evaluation Protocol

To evaluate the developed framework, we first analyze how each transformation impacts the results in the validation folds during the cross-validation process (Section 5.2). For each pipeline containing a specific transformation, we calculate the mean and standard deviation of AUC scores from both validation and training sets used during framework optimization. We analyze each pipeline step separately - the imputation step based on its strategy, the feature extraction step based on both the feature extractor type and sliding window lag size, and the feature selection step based on its strategy and corresponding threshold value. This systematic analysis helps us understand the contribution of each transformation to the overall pipeline performance.

The automatic framework's optimization phase produces an optimized pipeline by maximizing the AUC metric using a LightGBM classifier. The details of this optimized pipeline are presented in Section 5.3. To evaluate the pipeline's real-world performance, we measure its AUC score on a held-out test dataset. Figure 5.1 illustrates this evaluation process.

To evaluate the framework's performance, we conducted comparisons against two baseline approaches. The first baseline (Baseline 1) implements a simplified pipeline with just two steps of the feature engineering process: data imputation using forward fill and correlation-based feature selection. We configured these components using the same parameters found in our optimized

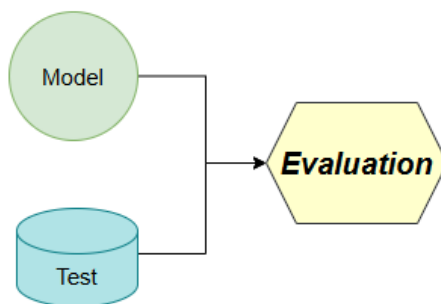


Figure 5.1: Optimal model evaluation process

pipeline. Our choice of forward fill and correlation-based selection is well-supported by the literature, as both techniques are widely adopted in their respective domains, as discussed in Section 2.4. We intentionally omitted the feature extraction step from this baseline to better understand how lag-based features and feature extractors can improve performance when analyzing irregular multivariate time series, particularly in our case study of predicting failures in pharmaceutical compression machines.

For our second baseline (Baseline 2), we constructed a three-step pipeline comprising data imputation, feature extraction, and feature selection. Rather than using the optimal parameter combination found by our framework, this baseline uses the strategies and parameters that showed the highest average performance during the optimization phase. By comparing this baseline against our optimized pipeline, we can determine whether selecting parameters based on average performance during optimization is competitive with the framework’s final optimized configuration.

Both Baseline 1 and Baseline 2 have as a classifier a LightGBM, similarly to what happens with the best (optimized) pipeline that we obtained. By using the same classifier on both baselines and the best pipeline, the comparisons rely solely on the feature engineering process.

In addition to comparing the baselines with the best pipeline, we created two variations of the best pipeline by changing only the final classifier, while keeping all other components identical. This allows us to evaluate which classifier performs best for this case study among the chosen models. The evaluation compares multiple performance metrics, including precision, recall, F1-score, AUC, and the number of false positives and false negatives. The detailed results of these comparisons are presented in Section 5.4.

Moreover, a benchmark including the three used feature extractors was performed in order to discover the given feature importance table of the best pipelines for each feature extractor. Those results were obtained during the optimization stage of the automatic framework, and the idea is to see how important the features generated by the feature extractors were. The runtime between the three feature extractors was also taken into account during the code execution. Both the feature importance and the runtime comparison can be seen in Section 5.5.

5.2 Operation performance

We analyzed how each of the three pipeline steps - imputation, feature extraction, and feature selection - impacts the overall performance of the framework. Tables 5.1, 5.2, and 5.3 present the results using the AUC metric as the performance measure.

Starting with data imputation 5.1, it can be noticed that for forward fill, the mean test score was 59.23%. Since only forward fill was applied, this value represents the average of all configurations.

In feature extraction 5.2, the best feature extractor, on average, is TSFEL using 3 lags, that is, the 3 last observations of a given feature, with an AUC of 63.09%, followed by pycatch22 with 3 or 5 lags, with an AUC of 61.07%. Lastly, the worst results were from tsfresh with a highest AUC of 57.99% when using 3 lags. Just looking at these 3 results, it can be seen that both TSFEL and pycatch22 have above 60% AUC results, but when looking at the standard deviation, TSFEL has around 2% while pycatch22 has around 5%. This means that, on average, over the given data for this case study, the TSFEL extractor performs better, but it may not mean that the best combination possible is obtained by using TSFEL, since the pycatch22 extractor has more dispersed results.

For the feature selection operation, two strategies were implemented: correlation and PCA. According to the results in Table 5.3, the correlation strategy outperformed PCA. The best threshold for correlation was 85%, with an AUC of 62.86%, while the best AUC PCA obtained was at 55.76%.

Table 5.1: Imputation Analysis

Strategy	Validation Score Mean	Validation Score Std. Dev.	Train Score Mean	Train Score Std. Dev.
ffill	0.5923	0.0436	0.8896	0.0889

Table 5.2: Feature Extraction Analysis

n_lags & Extractor	Validation Score Mean	Validation Score Std. Dev.	Train Score Mean	Train Score Std. Dev.
(3, pycatch22)	0.6107	0.0543	0.9050	0.0750
(3, TSFEL)	0.6309	0.0220	0.8817	0.0907
(3, tsfresh)	0.5799	0.0308	0.8906	0.0637
(5, pycatch22)	0.6107	0.0485	0.9156	0.0740
(5, TSFEL)	0.6042	0.0279	0.8790	0.0987
(5, tsfresh)	0.5674	0.0410	0.8621	0.0987
(8, pycatch22)	0.5910	0.0372	0.9234	0.0699
(8, TSFEL)	0.5758	0.0201	0.8827	0.0983
(8, tsfresh)	0.5599	0.0419	0.8664	0.1015

Table 5.3: Feature Selection Analysis

Strategy & Threshold	Validation Score Mean	Validation Score Std. Dev.	Train Score Mean	Train Score Std. Dev.
(correlation, 0.85)	0.6286	0.0268	0.9645	0.0380
(correlation, 0.9)	0.6252	0.0273	0.9650	0.0371
(correlation, 0.95)	0.6270	0.0264	0.9656	0.0368
(pca, 0.85)	0.5576	0.0259	0.8142	0.0550
(pca, 0.9)	0.5576	0.0259	0.8142	0.0550
(pca, 0.95)	0.5576	0.0259	0.8142	0.0550

5.3 Best performance parameters

By searching over the results dataset for the entry with the highest AUC score, the best pipeline could be found, being the best parameters described below:

- The classifier uses a base estimator with a maximum depth of 5 and 100 estimators.
- The feature extraction process includes 5 time lags.
- The extraction method is pycatch22, with default parameters (meaning that it computes all the metrics it can and not a subset of those).
- Features are selected using a correlation-based strategy with a threshold of 0.85.
- Missing values are handled through the forward fill strategy.

During the grid search optimization on the validation data, this pipeline achieved an AUC score of 67.94%. The performance of this optimal pipeline on the held-out test dataset is evaluated in Section 5.4.

Moreover, as shown in Section (5.2), although the pycatch22 extractor exhibited greater variability in AUC scores compared to the other extractors, it still demonstrated the best overall performance. This indicates that, despite some inconsistency across configurations, pycatch22 has the potential to generate highly effective pipelines, as was ultimately confirmed in the case study.

This finding aligns with our earlier discussion in Section 5.2, where we noted that TSFEL's higher mean AUC score did not necessarily guarantee optimal performance. Indeed, pycatch22's greater exploration of the solution space allowed it to find superior configurations, despite having a lower average performance than TSFEL.

5.4 Comparative results

As mentioned above in Section 5.1, this section consists of comparing the best pipeline (Pipeline 1 on Table 5.4) against two different baselines (Baseline 1 and Baseline 2 on Table 5.4), both

described in the same Section 5.1 as well, and the comparison of several tree-based models by replacing them in the classifier step of the best pipeline (Pipelines 1, 2, and 3 on Table 5.4).

Both the comparison of the best pipeline using multiple models and the baselines can be found in the Table 5.4.

Table 5.4: Comparison of best pipeline (with multiple tree-based classifiers) and baseline pipeline on test set.

Pipeline	Precision	Recall	F1-score	AUC	False Positives	False Negatives
1 - Best (LightGBM)	0.4545	0.0446	0.0813	0.6275	6	107
2 - Best (RandomForest)	0.0000	0.0000	0.0000	0.6944	0	112
3 - Best (CART)	0.3256	0.1250	0.1806	0.6518	29	98
4 - Baseline 1 (LightGBM)	0.3333	0.0092	0.0179	0.6021	4	215
5 - Baseline 2 (LightGBM)	0.2000	0.0067	0.0129	0.6386	4	149

Regarding the baseline comparison, it is noticeable that the best obtained pipeline (pipeline 1) outperformed Baseline 1 (pipeline 4). Precision increased from 33.33% to 45.45%, recall from 0.92% to 4.46%, and AUC improved by 2.54 percentage points (from 60.21% to 62.75%). Notably, the number of false negatives dropped by more than half (from 215 to 107), demonstrating that the model is catching more actual failures while maintaining a relatively low number of false positives (only 2 more than the baseline). Although recall remains low, the boost in precision is particularly important, indicating that nearly half of the predicted failures were correct.

Regarding Baseline 2 (pipeline 5), it has higher discriminative capacity than pipeline 1. In other metrics, such as precision, recall, and F1-score, the pipeline 1 outperformed the baseline as expected.

However, outperforming does not mean that the results were great. The biggest achievement is, regarding pipeline 1, the precision, fixed at 45.45%. This result means that, from the failure predictions of the model, fewer than half were correct. However, under the PdM context, recall is a crucial metric since it indicates the percentage of corrected predicted anomalies, and, in this metric, the LightGBM only got 4.46%, which is low.

From a company perspective, recall and precision are both important, since they represent the percentage of identified anomalies, and precision represents the percentage of predicted failures that are actual failures. However, depending on the problem definition, one statistic may be more important than the other. Since one of the main goals of PdM is to reduce the downtime of the monitored machine, identifying the maximum number of failures is desirable, while maintaining a good precision rate to diminish false positives (false alarms). If the precision rate decreases too much, the number of false alarms increases, meaning the machine will be stopped without a valid reason - the model detected a failure that in fact was not a failure.

Since a combination of both precision and recall needs to be used, we need to take a look at the F1-score metric, which represents the harmonic mean of both precision and recall. In other words, a higher F1-score means a more useful pipeline to the company. Looking at the F1-measure of pipeline 1, we notice it is also a low number: 8.13%, as a direct consequence of the low recall.

To improve this metric, two other models were used, CART [53] and random forest [50], and, with the first one (pipeline 3), we achieved an F1-score that is more than two times better, 18.06%. According to the F1-score metric, the best pipeline is pipeline 3, which uses CART as the classifier, followed by pipeline 1 that uses LightGBM.

Regarding random forest (pipeline 2), even though it achieved the highest discriminative capacity with an AUC of 69.44%, it completely failed to identify any positive cases (precision and recall both 0%), making it ineffective in practice.

The metrics presented in Table 5.4, used during this section, were obtained based on a threshold of 0.5, which is a typical approach in machine learning. Overall, the CART decision tree model offers the most practical solution for this case study.

One factor that may be leading to poor results is the inability of LightGBM and other tree-based algorithms to fully understand the nuances presented on large datasets, like the one presented in the case study. In the experimental design, only tree-based models were used to compare models' predictive performance. However, LSTM neural networks are widely used in the PdM context, opening the possibility to enhance the results of the case study if they were used. The major problem, once again, would reside in the code performance. As stated in the chapter 3, the framework, due to its integration with external feature extractors, has poor computational performance, despite the developed optimizations, and using an LSTM neural network on top of that would worsen that performance even more.

5.5 Feature extractors

Feature extractors significantly expand the feature space by computing additional attributes from the original data. Therefore, it is essential to evaluate whether these newly computed features contribute meaningfully to the model's performance, specifically in our LightGBM classifier. Additionally, we must consider the computational overhead introduced by feature extraction, as this can be a critical factor in selecting the most appropriate extraction method for practical applications.

5.5.1 Feature importance

To analyze feature importance, we focused on the top 10 most important features for each pipeline. Figure 5.2 shows the distribution between original and extracted features among these top 10 features, comparing results across the different feature extraction methods used in the best-performing pipelines.

A more complete plot is also presented in the Figure 5.3. Three tables are presented, corresponding to the results of the best pipelines for each of the three feature extractors used. In each table, the names of the most important features and their corresponding importance are presented, ordered from the most to the least important ones. Moreover, a tag mentioning if the feature was computed by the feature extractor or not is included, validating the numbers presented in the Figure 5.2. The original feature names are discretized for privacy reasons.

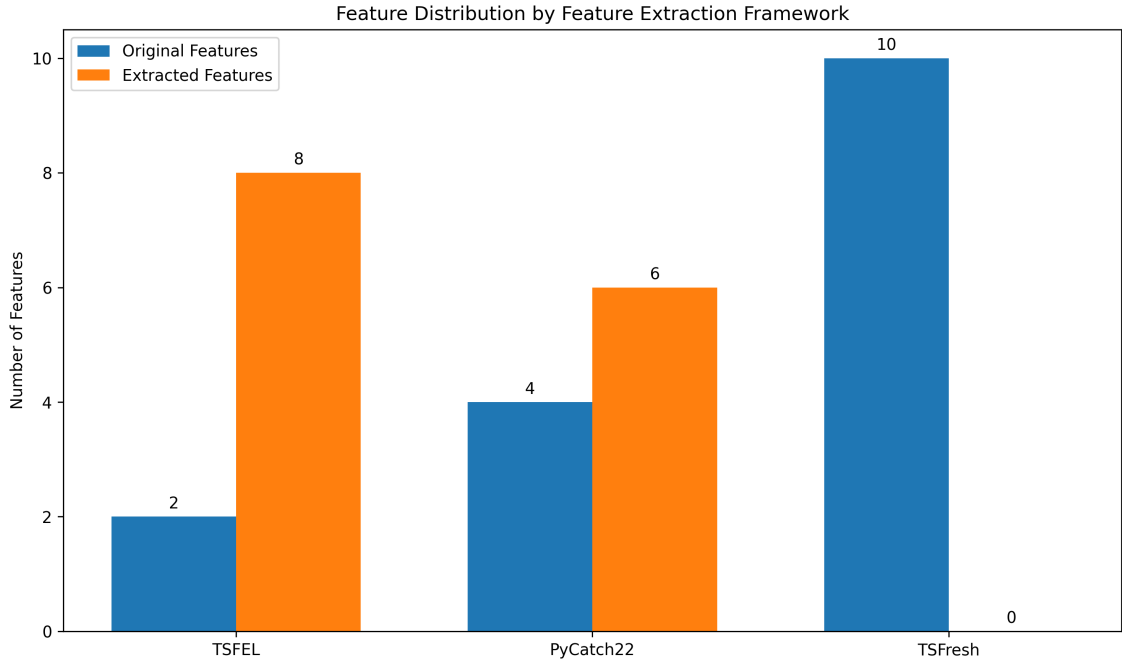


Figure 5.2: Top 10 feature importance distribution

As can be seen in Figure 5.2, TSFEL and pycatch22’s best pipelines have, at the top 10 most relevant features, a majority of those that were computed by the respective feature extractors, while the same does not happen on tsfresh. This suggests that the computed features by tsfresh are introducing either noise or irrelevant data to the model. This indicator also shows one potential reason for the slightly worse results obtained with tsfresh, while comparing with the other two extractors.

Following the same school of thought, we can also infer that both TSFEL and pycatch22 introduce valuable features to the model, helping it to better classify anomaly events. This, together with the comparison between the best pipeline and Baseline 1 (pipeline 1 and 4 in Table 5.4), helps to understand the importance of feature extraction over time series under the PdM context.

5.5.2 Execution Time

When working with large volumes of data, execution time and computational power are concepts that need to be addressed.

Regarding this subject, pycatch22 is clearly above the other options, taking only 6h to run the grid search, while TSFEL took 21h and tsfresh took 89h. These results were obtained using 5 cores of an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz. However, these times correspond to the entire run time of the pipeline and not specifically to the feature extractor time. However, since the execution time of other steps is residual compared to the feature extractor run time, we can use the total time to discuss the run time of the feature extraction step. The explanation of the fastest time being performed by pycatch22 resides in its implementation, in the C language, while the other frameworks were developed in Python, which is a much slower language [47].

Top 10 Features for each feature extractor

Top 10 Features: TSFEL

Feature	Importance	Type
feature1_0_Skewness	117	Extracted
feature2_0_Slope	93	Extracted
feature3	92	Original
feature1	87	Original
feature1_0_Root mean square	83	Extracted
feature4_0_Slope	75	Extracted
feature4_0_Root mean square	71	Extracted
feature5_0_Slope	71	Extracted
feature4_0_Skewness	69	Extracted
feature1_0_Slope	68	Extracted

Top 10 Features: TSFresh

Feature	Importance	Type
feature1	25	Original
feature3	18	Original
feature6	13	Original
feature7	8	Original
feature8	7	Original
feature9	7	Original
feature10	6	Original
feature11	6	Original
feature12	5	Original
feature13	4	Original

Top 10 Features: PyCatch22

Feature	Importance	Type
feature3	57	Original
feature14_DN_HistogramMode_5	44	Extracted
feature1_FC_LocalSimple_mean3_stderr	43	Extracted
feature15_DN_HistogramMode_10	43	Extracted
feature6	40	Original
feature15_DN_HistogramMode_5	36	Extracted
feature1	35	Original
feature15_CO_Embed2_Dist_tau_d_expfit_meandiff	35	Extracted
feature14_CO_Embed2_Dist_tau_d_expfit_meandiff	35	Extracted
feature16	34	Original

Figure 5.3: Top 10 most important features for the best pipeline using distinct feature extractors

5.6 Disussion

The developed framework brings numerous advantages in the context of PdM, particularly in scenarios where irregular and multivariate time series occur. The discussion in this section highlights the main key strengths and their practical relevance to real-world scenarios.

5.6.1 Strengths of the proposed framework

The most important contribution of this framework is the automation of the entire feature engineering process. Traditional PdM pipelines require vast domain knowledge and manual tuning to preprocess data, extract meaningful features, and select relevant ones, making it time-consuming.

In contrast, this framework integrates data regularization, imputation, feature extraction, feature selection, and normalization into a single pipeline, contrary to other frameworks presented on the state-of-the-art [16], which focus more on feature extraction and selection, for example.

This significantly reduces the effort needed to perform feature engineering and the time needed to prepare data for supervised learning tasks.

The framework also follows a modular design, which allows for easy expansion and modification. Because each transformation step is contained within a parameterized, reusable transformer, users can include additional parameters without needing to modify the framework. In other words, new techniques can easily be integrated with the framework, such as new feature extractors, new imputation methods, or even new steps in the pipeline can be easily added or removed, as demonstrated in the case study, where two steps were removed. Due to its versatility, the framework may readily be modified to accommodate different use cases, datasets, and modeling objectives outside of the conducted case study.

Moreover, the system incorporates clever efficiency optimizations to address computational complexity, mainly during feature extraction (Section 3.2.2.3). It accomplishes significant speed gains without compromising final model performance by pre-selecting the most pertinent features and reducing the number of combinations sent to extractors.

The design of the framework facilitates the simple incorporation of other binary classification algorithms, even though LightGBM was chosen as the default classifier due to its speed and resilience. To ensure seamless integration with the framework, only probabilistic classification algorithms are applicable. This is due to the decision to make AUC the default metric of evaluation, which is computed based on feature probabilities. This decision was made since it aligns with best practices for imbalanced classification, as happens in PdM, where failures are scarce events. Furthermore, probabilistic predictions are important for better decision-making. Another advantage of this framework consists of the test battery that was performed on the transformers, which ensures the reliability and reproducibility of the framework.

Lastly, the case study based on actual industrial machinery demonstrates the framework's application and usefulness. The framework exhibits its usefulness and resilience in a real-world scenario by tackling a specific issue in the pharmaceutical manufacturing sector: predicting compression machine failures.

5.6.2 Limitations

Even though the framework possesses a lot of advantages, it also has some downsides. The main limitation of this framework resides in its validation. Even though it was tested and validated in a real-world scenario of PdM, in the pharmaceutical industry, to really understand how well this framework adapts to diverse domains, additional tests with other case studies and other datasets needed to be conducted as well. Besides, in the conducted case study, some steps of the developed pipeline could not be tested due to the problem definition defined in Section 4.4, namely, regularization and normalization.

This framework is dependent on external frameworks for the feature extraction phase. As seen during the Sections 3.2.2.3 and 5.5.2, this step is the bottleneck of the pipeline execution, even after the performed optimizations. So, this step makes the pipeline computationally heavy as demonstrated in the result Section 5.5.2. Moreover, grid search is a computationally heavy search

strategy, and when added to some of the frameworks, such as tsfresh, it can worsen even further the code performance.

Chapter 6

Conclusions and Future Work

PdM depends on time series data, which is frequently high-dimensional, irregular, and incomplete. This presents a number of difficulties, including feature redundancy, missing values, and irregular data. The feature engineering process is done, most of the time, without a systematic methodology, and it's a time-consuming and domain-dependent task.

This dissertation addressed the challenge of automating the feature engineering process for PdM. A modular and adaptable pipeline was created to address these problems by automating crucial feature engineering steps, such as resampling, imputation, feature extraction and selection, and normalization, which ultimately resulted in a classification model. The proposed framework integrates state-of-the-art feature extraction libraries and supports parameter optimization, such as grid and random search strategies, to enhance predictive performance.

This dissertation has two main contributions. The first one is the proposed framework to automate and optimize the feature engineering process in PdM. The second one is the comparison between feature extraction libraries applied directly to a pharmaceutical industry case study.

6.1 Research question answers

***RQ1:** How can we automate the feature engineering process in PdM problems while handling various data difficulty challenges of multivariate time series data collected on sensors?*

By developing a framework that contains a structured pipeline that sequentially addresses irregularity, missing values, feature extraction, feature selection, and normalization, and an optimization stage, where, by leveraging search strategies, all different combinations for pipeline parameters are tested, we can indeed automate the entire feature engineering process addressing all the mentioned challenges, while getting an optimized model to further enhance results.

***RQ2:** How do different feature engineering operations affect the performance and effectiveness of the feature engineering framework in a pharmaceutical manufacturing application?*

Data imputation's AUC was 59.23% for the forward fill technique, the highest AUC for feature extraction was TSFEL with 3 lags, having an AUC of 63.03%, and lastly, for feature selection, the biggest AUC was 62.86% for the correlation strategy with a threshold of 0.85. Moreover, the results of feature importance analysis showed that TSFEL and pycatch22 provided more useful features than tsfresh. This can be justified by the better results for the feature extraction step when using TSFEL and pycatch22 compared to tsfresh.

***RQ3:** Which time series feature extraction method is the most effective lag-based feature extraction in a pharmaceutical manufacturing application?*

TSFEL presents, on average, a better AUC than pycatch22, and it has a lower standard deviation. This indicates that, on average, TSFEL produces better results. However, for this particular case study on a pharmaceutical manufacturing application, the best pipeline was produced with pycatch22 as the feature extractor. Furthermore, pycatch22's C-based implementation, as opposed to the slower Python-based extractors, as TSFEL and tsfresh, made it stand out in terms of execution performance.

***RQ4:** How do different learning algorithms compare in terms of predictive performance for a pharmaceutical manufacturing PdM application?*

From a commercial standpoint, where identifying as many real failures as possible is crucial while maintaining a good precision rate, the most reliable metric to evaluate the performance of a given model is the F1-score metric, since it represents a harmonic mean between precision and recall. While Random Forests, despite their excellent accuracy and AUC, were unable to identify any failures, making them ineffective, CART decision trees increased the F1-metric to 18.06%, providing a more workable alternative.

In this dissertation, only tree-based algorithms were used: LightGBM, Decision Tree (CART), and Random Forest. The most feasible one was the CART model, with an F1-score of 18.06%, since the Random Forest failed to identify any failures - F1-score was 0% - and the LightGBM has a low recall, meaning a lack of ability to detect existing failures. Even though it stands out in terms of precision, due to the low recall, it shows a poor F1-score, unlike the CART model.

6.2 Future work

As stated at the end of Section 5.4, results could be improved if more models were tested. Amongst these other models, the LSTM neural networks are a strong candidate due to their strong presence in PdM literature. However, implementing it would significantly worsen the computational performance of the overall framework. Regarding the framework itself, it would be more generic if it were suitable not only for supervised approaches but also had an option for unsupervised approaches.

Another way that could improve the results is to try to implement new strategies, new feature extractors, but especially new imputation methods due to the comparatively worse contribution to the overall result (AUC of 59.23%) of this step, and new feature selection methods.

Taking into account the evaluation of the framework, the results obtained in Table 5.4 were based on a decision threshold of 0.5. This value is static and could be optimized as future work. This optimization could bring great results improvements, especially regarding the random search model that was unable to detect any anomaly. Furthermore, testing and validating the framework with other case studies and other datasets would be beneficial to understand the behaviour of the framework over different domains and data.

Lastly, it would be interesting to compare the developed framework with other similar frameworks. However, from the conducted research presented in chapter 2, the only framework that automates the full feature engineering process for PdM was PREPIPE [33], whose implementation was focused on the case study conducted also in the same article, and required manual intervention to run each step of the framework individually, unlike the developed package. As such, comparisons with this framework and all the others mentioned during chapter 2 are not feasible and, for that reason, not implemented. However, when new frameworks emerge, it would be interesting to compare them with the one developed in this dissertation.

References

- [1] Rami Ahmad et al. “A Reliable Approach for Lightweight Anomaly Detection in Sensors Using Continuous Wavelet Transform and Vector Clustering”. In: *IEEE Sensors Journal* 24.15 (2024), pp. 24921–24930. DOI: [10.1109/JSEN.2024.3407158](https://doi.org/10.1109/JSEN.2024.3407158).
- [2] Daniele Apiletti et al. “iSTEP, an Integrated Self-Tuning Engine for Predictive Maintenance in Industry 4.0”. In: *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BD-Cloud/SocialCom/SustainCom)*. 2018, pp. 924–931. DOI: [10.1109/BDCLOUD.2018.00136](https://doi.org/10.1109/BDCLOUD.2018.00136).
- [3] Oluseun Omotola Aremu et al. “A relative entropy based feature selection framework for asset data in predictive maintenance”. In: *Computers & Industrial Engineering* 145 (2020), p. 106536. ISSN: 0360-8352. DOI: <https://doi.org/10.1016/j.cie.2020.106536>. URL: <https://www.sciencedirect.com/science/article/pii/S0360835220302709>.
- [4] Julien Audibert et al. “USAD: UnSupervised Anomaly Detection on Multivariate Time Series”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD ’20*. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 3395–3404. ISBN: 9781450379984. DOI: [10.1145/3394486.3403392](https://doi.org/10.1145/3394486.3403392). URL: <https://doi.org/10.1145/3394486.3403392>.
- [5] Mehdi Saman Azari et al. “A Systematic Literature Review on Transfer Learning for Predictive Maintenance in Industry 4.0”. In: *IEEE Access* 11 (2023), pp. 12887–12910. DOI: [10.1109/ACCESS.2023.3239784](https://doi.org/10.1109/ACCESS.2023.3239784).
- [6] Minghan Bao et al. “Integrating Machine Learning With Sensor Technology for Multiphase Flow Measurement: A Review”. In: *IEEE Sensors Journal* 24.19 (2024), pp. 29603–29618. DOI: [10.1109/JSEN.2024.3437292](https://doi.org/10.1109/JSEN.2024.3437292).
- [7] Marília Barandas et al. “TSFEL: Time Series Feature Extraction Library”. In: *SoftwareX* 11 (2020), p. 100456. ISSN: 2352-7110. DOI: <https://doi.org/10.1016/j.softx.2020.100456>. URL: <https://www.sciencedirect.com/science/article/pii/S2352711020300017>.
- [8] Mohammed Ayalew Belay et al. “Unsupervised Anomaly Detection for IoT-Based Multivariate Time Series: Existing Solutions, Performance Analysis and Future Directions”. In: *Sensors* 23.5 (2023). ISSN: 1424-8220. DOI: [10.3390/s23052844](https://doi.org/10.3390/s23052844). URL: <https://www.mdpi.com/1424-8220/23/5/2844>.

- [9] Eleonora Borgia. “The Internet of Things vision: Key features, applications and open issues”. In: *Computer Communications* 54 (2014), pp. 1–31. ISSN: 0140-3664. DOI: <https://doi.org/10.1016/j.comcom.2014.09.008>. URL: <https://www.sciencedirect.com/science/article/pii/S0140366414003168>.
- [10] Paula Branco, Luís Torgo, and Rita P Ribeiro. “A survey of predictive modeling on imbalanced domains”. In: *ACM computing surveys (CSUR)* 49.2 (2016), pp. 1–50.
- [11] Dominik Buhl and Carsten Lanquillon. “Challenges of Machine Failure Prediction with Product Data - A Case Study”. In: *Artificial Intelligence in HCI*. Ed. by Helmut Degen and Stavroula Ntoa. Cham: Springer Nature Switzerland, 2024, pp. 308–322.
- [12] Matteo Calabrese et al. “SOPHIA: An Event-Based IoT and Machine Learning Architecture for Predictive Maintenance in Industry 4.0”. In: *Information* 11.4 (2020). ISSN: 2078-2489. DOI: [10.3390/info11040202](https://doi.org/10.3390/info11040202). URL: <https://www.mdpi.com/2078-2489/11/4/202>.
- [13] Diogo Cardoso and Luís Ferreira. “Application of Predictive Maintenance Concepts Using Artificial Intelligence Tools”. In: *Applied Sciences* 11.1 (2021). ISSN: 2076-3417. DOI: [10.3390/app11010018](https://doi.org/10.3390/app11010018). URL: <https://www.mdpi.com/2076-3417/11/1/18>.
- [14] Tânia M. M. Carvalho. “Anticipation of Perturbances in Telco Services”. In: 2019, pp. 1–93.
- [15] Thyago P. Carvalho et al. “A systematic literature review of machine learning methods applied to predictive maintenance”. In: *Computers & Industrial Engineering* (2019), pp. 1–10.
- [16] Vitor Cerqueira, Nuno Moniz, and Carlos Soares. “Vest: Automatic feature engineering for forecasting”. In: *Machine Learning* (2021), pp. 1–23.
- [17] Vitor Cerqueira, Luis Torgo, and Carlos Soares. “Early anomaly detection in time series: a hierarchical approach for predicting critical health episodes”. In: *Machine Learning* 112.11 (2023), pp. 4409–4430.
- [18] Vitor Cerqueira, Luis Torgo, and Carlos Soares. “Model selection for time series forecasting an empirical analysis of multiple estimators”. In: *Neural processing letters* 55.7 (2023), pp. 10073–10091.
- [19] Vitor Cerqueira et al. “Combining Boosted Trees with Metafeature Engineering for Predictive Maintenance”. In: Oct. 2016, pp. 393–397. ISBN: 978-3-319-46348-3. DOI: [10.1007/978-3-319-46349-0_35](https://doi.org/10.1007/978-3-319-46349-0_35).
- [20] Anisha Chazhoor et al. “Predictive Maintenance using Machine Learning Based Classification Models”. In: *IOP Conference Series: Materials Science and Engineering* 954.1 (Oct. 2020), p. 012001. DOI: [10.1088/1757-899X/954/1/012001](https://doi.org/10.1088/1757-899X/954/1/012001). URL: <https://dx.doi.org/10.1088/1757-899X/954/1/012001>.
- [21] Maximilian Christ et al. “Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)”. In: *Neurocomputing* 307 (2018), pp. 72–77. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2018.03.067>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231218304843>.
- [22] Pádraig Cunningham, Matthieu Cord, and Sarah Jane Delany. “Supervised Learning”. In: *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*. Ed. by Matthieu Cord and Pádraig Cunningham. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. ISBN: 978-3-540-75171-7. DOI: [10.1007/978-3-540-75171-7_2](https://doi.org/10.1007/978-3-540-75171-7_2). URL: https://doi.org/10.1007/978-3-540-75171-7_2.

- [23] Nawal Daoudi, Smail Zaki, and Mohamed Aboussaleh. *Predictive Maintenance System for Screw Compressors Using Machine Learning: A Comparative Study*. <https://doi.org/10.21203/rs.3.rs-5326895/v1>. PREPRINT (Version 1). Oct. 2024.
- [24] Harsh S. Dhiman et al. “Enhancing Wind Turbine Reliability through Proactive High Speed Bearing Prognosis Based on Adaptive Threshold and Gated Recurrent Unit Networks”. In: *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*. 2023, pp. 1–6. DOI: [10.1109/IECON51785.2023.10312064](https://doi.org/10.1109/IECON51785.2023.10312064).
- [25] Gopi Krishna Durbhaka and Barani Selvaraj. “Predictive maintenance for wind turbine diagnostics using vibration signal analysis based on collaborative recommendation approach”. In: *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. 2016, pp. 1839–1842. DOI: [10.1109/ICACCI.2016.7732316](https://doi.org/10.1109/ICACCI.2016.7732316).
- [26] D.B. Durocher and G.R. Feldmeier. “Predictive versus preventive maintenance”. In: *IEEE Industry Applications Magazine* 10.5 (2004), pp. 12–21. DOI: [10.1109/MIA.2004.1330766](https://doi.org/10.1109/MIA.2004.1330766).
- [27] Abdussalam Elhanashi et al. “Integration of Deep Learning into the IoT: A Survey of Techniques and Challenges for Real-World Applications”. In: *Electronics* 12.24 (2023). ISSN: 2079-9292. DOI: [10.3390/electronics12244925](https://doi.org/10.3390/electronics12244925). URL: <https://www.mdpi.com/2079-9292/12/24/4925>.
- [28] Luís Ferreira et al. “Using supervised and one-class automated machine learning for predictive maintenance”. In: *Applied Soft Computing* 131 (2022), p. 109820. ISSN: 1568-4946. DOI: <https://doi.org/10.1016/j.asoc.2022.109820>. URL: <https://www.sciencedirect.com/science/article/pii/S1568494622008699>.
- [29] Joao Gama et al. “From fault detection to anomaly explanation: A case study on predictive maintenance”. In: *Journal of Web Semantics* 81 (2024), p. 100821.
- [30] Jean-Gabriel Gaudreault, Paula Branco, and João Gama. “An analysis of performance metrics for imbalanced classification”. In: *International Conference on Discovery Science*. Springer. 2021, pp. 67–77.
- [31] Alexander Geiger et al. *TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks*. 2020. arXiv: [2009.07769](https://arxiv.org/abs/2009.07769) [cs.LG]. URL: <https://arxiv.org/abs/2009.07769>.
- [32] Morteza Ghobakhloo. “Industry 4.0, digitization, and opportunities for sustainability”. In: *Journal of Cleaner Production* 252 (2020), p. 119869. ISSN: 0959-6526. DOI: <https://doi.org/10.1016/j.jclepro.2019.119869>. URL: <https://www.sciencedirect.com/science/article/pii/S0959652619347390>.
- [33] Danilo Giordano et al. “Data-driven strategies for predictive maintenance: Lesson learned from an automotive use case”. In: *Computers in Industry* 134 (2022), p. 103554. ISSN: 0166-3615. DOI: <https://doi.org/10.1016/j.compind.2021.103554>. URL: <https://www.sciencedirect.com/science/article/pii/S0166361521001615>.
- [34] Ariel Gorenstein and Meir Kalech. “Predictive maintenance for critical infrastructure”. In: *Expert Systems with Applications* 210 (2022), p. 118413. ISSN: 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2022.118413>. URL: <https://www.sciencedirect.com/science/article/pii/S0957417422015196>.
- [35] Michael Greenacre et al. “Principal component analysis”. In: *Nature Reviews Methods Primers* 2.1 (Dec. 2022), p. 100. ISSN: 2662-8449. DOI: [10.1038/s43586-022-00184-w](https://doi.org/10.1038/s43586-022-00184-w). URL: <https://doi.org/10.1038/s43586-022-00184-w>.

- [36] Isabelle Guyon and André Elisseeff. “An introduction to feature extraction”. In: *Feature extraction*. Springer, 2006, pp. 1–25.
- [37] Mark A Hall. “Correlation-based feature selection for machine learning”. PhD thesis. The University of Waikato, 1999.
- [38] Manuel Herrera et al. “Handling Irregularly Sampled IoT Time Series to Inform Infrastructure Asset Management”. In: *IFAC-PapersOnLine* 55.19 (2022). 5th IFAC Workshop on Advanced Maintenance Engineering, Services and Technologies AMEST 2022, pp. 241–245. ISSN: 2405-8963. DOI: <https://doi.org/10.1016/j.ifacol.2022.09.214>. URL: <https://www.sciencedirect.com/science/article/pii/S240589632201432X>.
- [39] Stephanie Holly et al. *Autoencoder based Anomaly Detection and Explained Fault Localization in Industrial Cooling Systems*. 2022. arXiv: 2210.08011 [cs.LG]. URL: <https://arxiv.org/abs/2210.08011>.
- [40] Bahrudin Hrnjica and Selver Softic. “Explainable AI in Manufacturing: A Predictive Maintenance Case Study”. In: *Advances in Production Management Systems. Towards Smart and Digital Manufacturing*. Ed. by Bojan Lalic et al. Cham: Springer International Publishing, 2020, pp. 66–73. ISBN: 978-3-030-57997-5.
- [41] Alan Jovic, Karla Brkić, and N. Bogunovic. “A review of feature selection methods with applications”. In: May 2015, pp. 1200–1205. DOI: [10.1109/MIPRO.2015.7160458](https://doi.org/10.1109/MIPRO.2015.7160458).
- [42] Ameeth Kanawaday and Aditya Sane. “Machine learning for predictive maintenance of industrial machines using IoT sensor data”. In: *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*. 2017, pp. 87–90. DOI: [10.1109/ICSESS.2017.8342870](https://doi.org/10.1109/ICSESS.2017.8342870).
- [43] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. “ExploreKit: Automatic Feature Generation and Selection”. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 2016, pp. 979–984. DOI: [10.1109/ICDM.2016.0123](https://doi.org/10.1109/ICDM.2016.0123).
- [44] Ambika Kaul, Saket Maheshwary, and Vikram Pudi. “AutoLearn — Automated Feature Generation and Selection”. In: *2017 IEEE International Conference on Data Mining (ICDM)*. 2017, pp. 217–226. DOI: [10.1109/ICDM.2017.31](https://doi.org/10.1109/ICDM.2017.31).
- [45] Udayan Khurana, Horst Samulowitz, and Deepak Turaga. “Feature engineering for predictive modeling using reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.
- [46] Udayan Khurana et al. “Cognito: Automated Feature Engineering for Supervised Learning”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. 2016, pp. 1304–1307. DOI: [10.1109/ICDMW.2016.0190](https://doi.org/10.1109/ICDMW.2016.0190).
- [47] Sabbi Vamshi Krishna et al. “Performance Comparison of Julia with C and Python for Solving Computational Problems”. In: *Advances in Computing and Data Sciences*. Ed. by Mayank Singh et al. Cham: Springer Nature Switzerland, 2025, pp. 35–45. ISBN: 978-3-031-70906-7.
- [48] kun han kun et al. “Residual Life Prediction Method of Industrial Equipment Based on Deep Learning and Attention Mechanism”. In: (2024). DOI: <http://dx.doi.org/10.2139/ssrn.4803714>.

- [49] Hongfei Li et al. “Improving rail network velocity: A machine learning approach to predictive maintenance”. In: *Transportation Research Part C: Emerging Technologies* 45 (2014). Advances in Computing and Communications and their Impact on Transportation Science and Technologies, pp. 17–26. ISSN: 0968-090X. DOI: <https://doi.org/10.1016/j.trc.2014.04.013>. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X14001107>.
- [50] Yanli Liu, Yourong Wang, and Jian Zhang. “New Machine Learning Algorithm: Random Forest”. In: *Information Computing and Applications*. Ed. by Baoxiang Liu, Maode Ma, and Jincai Chang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 246–252. ISBN: 978-3-642-34062-8.
- [51] Carl Lubba et al. *catch22 : CAnonical Time-series CHaracteristics - Feature Description*. <https://time-series-features.gitbook.io/catch22/information-about-catch22/feature-descriptions>. Accessed: 2025-05-28. 2024.
- [52] Carl H. Lubba et al. “catch22: CAnonical Time-series CHaracteristics”. In: *Data Mining and Knowledge Discovery* 33.6 (2019), pp. 1821–1852. ISSN: 1573-756X. DOI: [10.1007/s10618-019-00647-x](https://doi.org/10.1007/s10618-019-00647-x). URL: <https://doi.org/10.1007/s10618-019-00647-x>.
- [53] Ali Mahmood, Satuluri Naganjaneyulu, and Mrithyumjaya Kuppa. “An Overview of Recent and Traditional Decision Tree Classifiers in Machine Learning”. In: *International Journal of Research and Reviews in Ad Hoc Networks* (2011).
- [54] Steffen Moritz and Thomas Bartz-Beielstein. “imputeTS: Time Series Missing Value Imputation in R”. In: *The R Journal* 9.1 (2017), pp. 207–218.
- [55] Jay I. Myung, Daniel R. Cavagnaro, and Mark A. Pitt. “A tutorial on adaptive design optimization”. In: *Journal of Mathematical Psychology* 57.3 (2013), pp. 53–67. ISSN: 0022-2496. DOI: <https://doi.org/10.1016/j.jmp.2013.05.005>. URL: <https://www.sciencedirect.com/science/article/pii/S0022249613000503>.
- [56] Irene Niyonambaza, Marco Zennaro, and Alfred Uwitonze. “Predictive Maintenance (PdM) Structure Using Internet of Things (IoT) for Mechanical Equipment Used into Hospitals in Rwanda”. In: *Future Internet* 12.12 (2020). ISSN: 1999-5903. DOI: [10.3390/fi12120224](https://doi.org/10.3390/fi12120224). URL: <https://www.mdpi.com/1999-5903/12/12/224>.
- [57] Feng Pan et al. “A survival analysis based volatility and sparsity modeling network for student dropout prediction”. In: *PLOS ONE* 17 (May 2022), pp. 1–22. DOI: [10.1371/journal.pone.0267138](https://doi.org/10.1371/journal.pone.0267138). URL: <https://doi.org/10.1371/journal.pone.0267138>.
- [58] Marina Paolanti et al. “Machine Learning approach for Predictive Maintenance in Industry 4.0”. In: July 2018, pp. 1–6. DOI: [10.1109/MESA.2018.8449150](https://doi.org/10.1109/MESA.2018.8449150).
- [59] Fábio Pinto, Carlos Soares, and Joao Mendes-Moreira. “Towards automatic generation of metafeatures”. In: *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I* 20. Springer. 2016, pp. 215–226.
- [60] Parivash Pirasteh et al. “Interactive feature extraction for diagnostic trouble codes in predictive maintenance: A case study from automotive domain”. In: *Proceedings of the Workshop on Interactive Data Mining. WIDM’19*. Melbourne, VIC, Australia: Association for Computing Machinery, 2019. ISBN: 9781450362962. DOI: [10.1145/3304079.3310288](https://doi.org/10.1145/3304079.3310288). URL: <https://doi.org/10.1145/3304079.3310288>.

- [61] Ragunathan Rajkumar et al. “Cyber-physical systems: The next computing revolution”. In: *Design Automation Conference*. 2010, pp. 731–736. DOI: [10.1145/1837274.1837461](https://doi.org/10.1145/1837274.1837461).
- [62] Bashar Rajoub. “Chapter 3 - Supervised and unsupervised learning”. In: *Biomedical Signal Processing and Artificial Intelligence in Healthcare*. Ed. by Walid Zgallai. Developments in Biomedical Engineering and Bioelectronics. Academic Press, 2020, pp. 51–89. DOI: <https://doi.org/10.1016/B978-0-12-818946-7.00003-2>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128189467000032>.
- [63] V N Ganapathi Raju et al. “Study the Influence of Normalization/Transformation process on the Accuracy of Supervised Classification”. In: *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. 2020, pp. 729–735. DOI: [10.1109/ICSSIT48917.2020.9214160](https://doi.org/10.1109/ICSSIT48917.2020.9214160).
- [64] Rita P. Ribeiro, Pedro Pereira, and João Gama. “Sequential anomalies: a study in the Rail-way Industry”. In: *Machine Learning* 105.1 (Oct. 2016), pp. 127–153. ISSN: 1573-0565. DOI: [10.1007/s10994-016-5584-6](https://doi.org/10.1007/s10994-016-5584-6). URL: <https://doi.org/10.1007/s10994-016-5584-6>.
- [65] Safwan Mahmood Al-Selwi et al. “RNN-LSTM: From applications to modeling techniques and beyond—Systematic review”. In: *Journal of King Saud University - Computer and Information Sciences* 36.5 (2024), p. 102068. ISSN: 1319-1578. DOI: <https://doi.org/10.1016/j.jksuci.2024.102068>. URL: <https://www.sciencedirect.com/science/article/pii/S1319157824001575>.
- [66] Uday Shankar Shanthamallu et al. “A brief survey of machine learning methods and their sensor and IoT applications”. In: *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*. 2017, pp. 1–8. DOI: [10.1109/IISA.2017.8316459](https://doi.org/10.1109/IISA.2017.8316459).
- [67] Neeraj Sharma and Mala Kalra. “Predictive Maintenance for Commercial Vehicles Tyres Using Machine Learning”. In: *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 2022, pp. 1–6. DOI: [10.1109/ICCCNT54827.2022.9984497](https://doi.org/10.1109/ICCCNT54827.2022.9984497).
- [68] Yu Shi et al. *lightgbm: Light Gradient Boosting Machine*. R package version 4.6.0.99. 2025. URL: <https://github.com/Microsoft/LightGBM>.
- [69] Robert H. Shumway and David S. Stoffer. “ARIMA Models”. In: *Time Series Analysis and Its Applications: With R Examples*. Cham: Springer International Publishing, 2017, pp. 75–163. ISBN: 978-3-319-52452-8. DOI: [10.1007/978-3-319-52452-8_3](https://doi.org/10.1007/978-3-319-52452-8_3). URL: https://doi.org/10.1007/978-3-319-52452-8_3.
- [70] Gian Antonio Susto, Alessandro Beghi, and Cristina de Luca. “A predictive maintenance system for epitaxy processes based on filtering and prediction techniques”. In: *IEEE Transactions on Semiconductor Manufacturing* (2012), pp. 638–649.
- [71] Kyle Swanson et al. “From patterns to patients: Advances in clinical machine learning for cancer diagnosis, prognosis, and treatment”. In: *Cell* 186.8 (Apr. 2023), pp. 1772–1791. ISSN: 0092-8674. DOI: [10.1016/j.cell.2023.01.035](https://doi.org/10.1016/j.cell.2023.01.035). URL: <https://doi.org/10.1016/j.cell.2023.01.035>.
- [72] Sean J. Taylor and Benjamin Letham. “Forecasting at Scale”. In: *The American Statistician* 72.1 (2018), pp. 37–45. DOI: [10.1080/00031305.2017.1380080](https://doi.org/10.1080/00031305.2017.1380080). URL: <https://doi.org/10.1080/00031305.2017.1380080>.

- [73] Christopher Irgens Thorsten Wuest Daniel Weimer and Klaus-Dieter Thoben. “Machine learning in manufacturing: advantages, challenges, and applications”. In: *Production & Manufacturing Research* 4.1 (2016), pp. 23–45. DOI: [10.1080/21693277.2016.1192517](https://doi.org/10.1080/21693277.2016.1192517).
- [74] Kanishka Tyagi et al. “Chapter 3 - Unsupervised learning”. In: *Artificial Intelligence and Machine Learning for EDGE Computing*. Ed. by Rajiv Pandey et al. Academic Press, 2022, pp. 33–52. ISBN: 978-0-12-824054-0. DOI: <https://doi.org/10.1016/B978-0-12-824054-0.00012-5>. URL: <https://www.sciencedirect.com/science/article/pii/B9780128240540000125>.
- [75] Ting Wang et al. “A spatiotemporal feature learning-based RUL estimation method for predictive maintenance”. In: *Measurement* 214 (2023), p. 112824. ISSN: 0263-2241. DOI: <https://doi.org/10.1016/j.measurement.2023.112824>. URL: <https://www.sciencedirect.com/science/article/pii/S0263224123003883>.
- [76] Zhen Wang et al. “Irregularly Sampled Multivariate Time Series Classification: A Graph Learning Approach”. In: *IEEE Intelligent Systems* 38.3 (2023), pp. 3–11. DOI: [10.1109/MIS.2023.3239797](https://doi.org/10.1109/MIS.2023.3239797).
- [77] Johan A. Westerhuis et al. “Assessment of PLS-DA cross validation”. In: *Metabolomics* 4.1 (Mar. 2008), pp. 81–89. ISSN: 1573-3890. DOI: [10.1007/s11306-007-0099-6](https://doi.org/10.1007/s11306-007-0099-6). URL: <https://doi.org/10.1007/s11306-007-0099-6>.
- [78] Kirsty Williamson et al. *Research Methods for Students, Academics and Professionals*. Elsevier Science & Technology, 2002.
- [79] Ozlem Ece Yurek and Derya Birant. “Remaining Useful Life Estimation for Predictive Maintenance Using Feature Engineering”. In: *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2019, pp. 1–5. DOI: [10.1109/ASYU48272.2019.8946397](https://doi.org/10.1109/ASYU48272.2019.8946397).
- [80] Zeldia B Zabinsky et al. “Random search algorithms”. In: *Department of Industrial and Systems Engineering, University of Washington, USA* (2009), p. 34.
- [81] Rizgar R. Zebari et al. “A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction”. In: *Journal of Applied Science and Technology Trends* 1.1 (May 2020), pp. 56–70. DOI: [10.38094/jastt1224](https://doi.org/10.38094/jastt1224). URL: <https://www.jastt.org/index.php/jasttpath/article/view/24>.
- [82] Jan Zenisek, Florian Holzinger, and Michael Affenzeller. “Machine learning based concept drift detection for predictive maintenance”. In: *Computers & Industrial Engineering* 137 (2019), p. 106031. ISSN: 0360-8352. DOI: <https://doi.org/10.1016/j.cie.2019.106031>. URL: <https://www.sciencedirect.com/science/article/pii/S0360835219304905>.
- [83] Weijia Zhang et al. “Irregular Multivariate Time Series Forecasting: A Transformable Patching Graph Neural Networks Approach”. In: *Forty-first International Conference on Machine Learning*. 2024. URL: <https://openreview.net/forum?id=UZlMXUGI6e>.
- [84] Tiago Zonta et al. “Predictive maintenance in the Industry 4.0: A systematic literature review”. In: *Computers & Industrial Engineering* 150 (2020), p. 106889.

Appendix A

optimusPipe package

The development of the framework presented in this dissertation is presented as a Python package called optimusPipe. Its implementation can be seen in the following repository:

- <https://github.com/JoseAlbanoGaspar/PdM-TS-framework>.

In the following section, an example of how to use the package is presented.

A.1 optimusPipe example

Listing A.1: optimusPipe example

```
1 import pandas as pd
2 from lightgbm import LGBMClassifier
3
4 # Load dataset
5 raw_df = pd.read_pickle("Datasets/final_dataset.pkl")
6
7 # Dataset column configuration
8 COLUMN_CONFIG = {
9     'primary_key': ['ProcessId', 'DateTime'],
10    'time_col': 'DateTime',
11    'target_col': 'event',
12    'protected_cols': ['ProcessId', 'DateTime', 'event'],
13    'id_col': 'ProcessId',
14 }
15
16 # Feature extraction configuration
17 N_FEATURES = 10
18 N_TSFEL_FEATURES = 10
19 N_TSFRESH_FEATURES = 10
20
21 # Parameter grid from pipeline.py
```

```

22 param_distributions = {
23     # Imputation parameters
24     'imputation__params': [
25         ('interpolate', 'linear'),
26         ('interpolate', ('polynomial', 2)),
27         ('interpolate', ('spline', 2)),
28         ('ffill', None),
29     ],
30
31     # Feature extraction parameters
32     'feature_extraction__n_lags': [3, 5, 8],
33
34     # Feature selection parameters
35     'feature_selection__strategy': ['correlation', 'pca'],
36     'feature_selection__threshold': [0.85, 0.9, 0.95],
37
38     # Classifier parameters
39     'classifier__base_estimator__max_depth': [3, 5, 7],
40     'classifier__base_estimator__n_estimators': [25, 50, 100]
41 }
42
43 if __name__ == "__main__":
44     from optimusPipe import optimize_pipeline
45
46     # Configure the optimization
47     base_classifier = LGBMClassifier()
48
49     # Run optimization with random search
50     results_random = optimize_pipeline(
51         dataset=raw_df,
52         column_config=COLUMN_CONFIG,
53         n_features=N_FEATURES,
54         n_tsfel_features=N_TSFEL_FEATURES,
55         n_tsfresh_features=N_TSFRESH_FEATURES,
56         base_classifier=base_classifier,
57         search_strategy='random',
58         n_jobs=2,
59         random_iterations=2,
60         output_dir='random_search_results',
61         param_distributions=param_distributions,
62         steps=['imputation', 'feature_extraction', 'feature_selection'],
63         feature_extractors=['pycatch22']
64     )

```