



DATA-DRIVEN TECHNIQUES FOR REAL-TIME MONITORING IN  
INDUSTRIAL SYSTEMS: INSTANCE SELECTION AND DRIFT-AWARE  
DIAGNOSIS

Thiago Koichi Anzai

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Química, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Química.

Orientador: José Carlos Costa da Silva Pinto

Rio de Janeiro

Abril de 2026

DATA-DRIVEN TECHNIQUES FOR REAL-TIME MONITORING IN  
INDUSTRIAL SYSTEMS: INSTANCE SELECTION AND DRIFT-AWARE  
DIAGNOSIS

Thiago Koichi Anzai

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ  
COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA  
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS  
REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR  
EM CIÊNCIAS EM ENGENHARIA QUÍMICA.

Orientador: José Carlos Costa da Silva Pinto

Aprovada por: Prof. Mauricio Bezerra de Souza Junior

Prof. Cristiano Hora de Oliveira Fontes

Dr. André Domingues Quelhas

Dr. Maurício Melo Câmara

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2026

Anzai, Thiago Koichi

Data-Driven Techniques for Real-Time Monitoring in Industrial Systems: Instance Selection and Drift-Aware Diagnosis/Thiago Koichi Anzai. – Rio de Janeiro: UFRJ/COPPE, 2026.

XVIII, 122 p.: il.; 29,7cm

Orientador: José Carlos Costa da Silva Pinto

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Química, 2026

Referências Bibliográficas: p. 90 – 102.

1. Monitoramento de processos. 2. Detecção e diagnóstico de falhas. 3. Seleção de instâncias. 4. Concept drift. 5. Ciência de dados. I. Pinto, José Carlos Costa da Silva *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Química. III. Título.

*“Deve nutrir-se carinho por um sofrimento sobre o qual se soube construir  
a felicidade.”*

Valter Hugo Mãe

# Agradecimentos

Começando por quem me começou: aos meus pais, Sonia e Kokichi.

A minha irmã, a 11.225,83 km de distância (eu procurei).

À Mariana.

Aos amigos Diego, Fernanda e Clarissa.

Ao meu orientador Zé, que me acompanha desde os tempos remotos em que meus cabelos eram pretos.

À Petrobras, pelo suporte para a realização do trabalho, e aos amigos que lá tenho: Diehl, Alexandre, Gabriel, Pedro, Tati...

Um agradecimento especial — e que não será lido — ao Pepe e à Pepita.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MONITORAMENTO EM TEMPO REAL DE SISTEMAS INDUSTRIAIS  
UTILIZANDO TÉCNICAS BASEADAS EM DADOS: SELEÇÃO DE  
INSTÂNCIAS E DIAGNÓSTICO SENSÍVEL AO *DRIFT*

Thiago Koichi Anzai

Abril/2026

Orientador: José Carlos Pinto

Programa: Engenharia Química

A área de monitoramento de processos com base em dados tem recebido cada vez mais atenção ao longo dos últimos anos, não apenas pelo desenvolvimento de novas técnicas, mas também pela necessidade de melhorar o desempenho e a segurança das operações industriais. Contudo, apesar do interesse, muitas aplicações em ambientes reais fracassam por razões não diretamente relacionadas à técnica empregada ou à capacidade do modelo em detectar anomalias. Uma razão para isso é que etapas essenciais para implantar o monitoramento em tempo real raramente são tratadas na literatura, que tende a concentrar-se na detecção, negligenciando procedimentos críticos que a antecedem e a sucedem. Este trabalho busca suprir essas lacunas apresentando: (i) a *Instance Selection Library* (ISLib), desenvolvida para identificar conjuntos de treinamento representativos por meio de uma abordagem em duas etapas; e (ii) a técnica de contribuição *Nearest Normal Value* (NNV), proposta para distinguir falhas do processo de falhas do modelo decorrentes de mudanças no modo de operação. Os resultados, avaliados em múltiplos conjuntos de dados representativos, indicam que a seleção de instâncias e o diagnóstico apropriado são fundamentais para implementações de monitoramento industrial mais robustas, eficientes e sustentáveis.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DATA-DRIVEN TECHNIQUES FOR REAL-TIME MONITORING IN  
INDUSTRIAL SYSTEMS: INSTANCE SELECTION AND DRIFT-AWARE  
DIAGNOSIS

Thiago Koichi Anzai

April/2026

Advisor: José Carlos Pinto

Department: Chemical Engineering

Data-driven process monitoring has received increasing attention over the past few years, not only because of the development of new techniques, but also due to the necessity of industry sectors to improve the performance and safety of their processes. However, despite growing interest in the subject, many real-world applications fail for reasons unrelated to the technique or model's ability to detect anomalies. One of the reasons for this is that an important part of the process of deploying a real-time monitoring application is not typically addressed by publications in the field, which tend to focus mainly on the detection stage, overlooking critical steps that precede and succeed fault detection. The present work addresses these gaps by presenting: (i) the Instance Selection Library (ISLib), developed to identify representative training datasets through a two-stage approach; and (ii) the Nearest Normal Value (NNV) contribution analysis technique, proposed to distinguish between genuine process faults and model inadequacies caused by operating mode changes. The findings, which were tested across multiple representative datasets, show that accurate instance selection and proper fault diagnosis are fundamental for more robust, efficient, and sustainable industrial monitoring system implementations.

# Contents

<b>List of Figures</b> .....	<b>x</b>
<b>List of Tables</b> .....	<b>xiii</b>
<b>List of Abbreviations and Symbols</b> .....	<b>xiv</b>
<b>1 Introduction</b> .....	<b>1</b>
1.1 General aspects .....	1
1.2 Objective.....	3
1.3 Nomenclature and terminology .....	3
1.4 Thesis structure: .....	4
<b>2 Literature review</b> .....	<b>6</b>
2.1 A brief history of process monitoring.....	6
2.2 General process monitoring framework .....	10
2.3 Concluding remarks.....	26
<b>3 Before Detection: Instance Selection for Model Training</b> .....	<b>28</b>
3.1 Instance Selection Library (ISLib) .....	29
3.1.1 Batch instance selection using ranked clusters .....	29
3.1.2 Incremental instance selection using a crescent window strategy .	32
3.2 Results and discussion .....	35
3.2.1 Methodology.....	35
3.2.2 Tennessee Eastman Process (TEP).....	37
3.2.3 Flare gas flowmeter.....	43
3.2.4 Oil flowmeter.....	49
3.3 Concluding remarks.....	55
<b>4 After Detection: Drift-Aware Fault Diagnosis</b> .....	<b>58</b>
4.1 Nearest Normal Value (NNV) Contribution Method .....	59

4.2	Results and discussion .....	65
4.2.1	Linear Autoregressive System .....	66
4.2.2	Tennessee Eastman Process (TEP).....	70
4.2.3	Oil flowmeter.....	80
4.3	Concluding remarks.....	84
<b>5</b>	<b>Conclusions and Future Directions .....</b>	<b>86</b>
5.1	Contributions.....	86
5.1.1	Formalization of instance selection as a pre-detection problem in industrial process monitoring .....	86
5.1.2	Reinterpretation of concept drift as a competing diagnostic hypothesis after fault detection.....	87
5.1.3	An integrated, deployment-oriented monitoring framework linking pre-detection and post-detection decisions.....	87
5.2	Future Directions.....	88
5.3	Final Remarks .....	89
	<b>Bibliography .....</b>	<b>90</b>
	<b>Appendix A.....</b>	<b>103</b>
	<b>Appendix B.....</b>	<b>108</b>
	<b>Appendix C.....</b>	<b>113</b>

# List of Figures

Figure 1: Categories of publications for data-driven fault detection and diagnosis by selected sources from 2017 to 2018. M = algorithm and methods development; R = review article; AP = applications (ALAUDDIN, KHAN, <i>et al.</i> , 2018). .....	2
Figure 2: SPC chart of a variable $\mathbf{X}$ , showing its desirable value (or under control value), $\mathbf{X}$ , Upper Control Limit (UCL) and Lower Control Limit (LCL). Points in red represent a deviation from Nelson’s rules (WIKIPEDIA, 2015). .....	7
Figure 3: Classification of diagnostic algorithms as proposed by VENKATASUBRAMANIAN, RENGASWAMY, KAVURI, <i>et al.</i> , (2003). .....	8
Figure 4: Approaches to multivariate statistical process control (MSPC) (ALDRICH, AURET, 2013). .....	9
Figure 5: Classification of fault detection and diagnosis methods (ALAUDDIN, KHAN, <i>et al.</i> , 2018). .....	9
Figure 6: General framework for a data-driven process monitoring application (ANZAI, Thiago K., DE BRITO, <i>et al.</i> , 2025). .....	10
Figure 7: ADWIN algorithm (left) and output (right) for an abrupt change. Figure adapted from (BIFET, GAVALDÀ, 2007). .....	23
Figure 8: Types of drift in a classification setting. The axes represent the feature space $\mathbf{X}$ , and $\mathbf{y}$ denotes the class label (indicated by the circle colors). The dashed line represents the decision boundary of the model. ....	23
Figure 9: Traditional monitoring paradigm showing conventional interpretations (solid arrows) and overlooked diagnostic possibilities (dashed arrows). .....	24
Figure 10: Representation of a result obtained from ISLib during the batch instance selection stage. ....	31
Figure 11: Enlarging window approach for instance selection. ....	33
Figure 12: Workflow of dataset processing using the ISLib library, showcasing its two main phases: clustering-based batch instance selection and the enlarging window strategy. ....	35

Figure 13: Process flowsheet of the Tennessee Eastman Process (TEP), depicting five main operational units: reactor, condenser, compressor, separator, and stripper (XAVIER, DE SEIXAS, 2018). .....	38
Figure 14: MSE values for the enlarging window models generated from the TEP dataset using a resolution of 100, i.e., with increasing windows of N100. ....	40
Figure 15: SPE profiles for the reduced and full model against different types of faults.....	41
Figure 16: Cluster analysis applied to the gas flowmeter dataset encompassing visual representation of the operational regions and ranked clusters.....	44
Figure 17: MSE values for the enlarging window models generated from the gas flowmeter dataset using a resolution of 100. ....	45
Figure 18: Density distributions for the full and reduced datasets, demonstrating that ISLib reduced data volume while preserving the original distribution characteristics. ....	46
Figure 19: SPE profiles obtained from the AE model trained using the whole dataset (in black) and the AE generated from the data provided by ISLib (in red). The grey dashed line represents the $\lambda$ value. ....	48
Figure 20: SPE profiles of both models and the measured values from the features during anomalies observed in the volume flow signal.....	49
Figure 21: Cluster analysis applied to the oil flowmeter dataset encompassing visual representation of the operational regions and ranked clusters.....	52
Figure 22: MSE values for the enlarging window models generated from the oil flowmeter dataset using a resolution of 100. ....	53
Figure 23: Predicted versus measured normalized values of both models during a test period. ....	54
Figure 24: Residues distribution of both ESN models.....	54
Figure 25: Proposed integrated framework using fault diagnosis as an intermediate layer.....	59
Figure 26: Workflow of NNV diagnosis procedure at a given time (t).....	63

Figure 27: Time series of process variables under spurious fault. A bias introduced in $y_2$ at sample 1000 affects only the measured variable while other variables remain unaltered. ....	68
Figure 28: Time series of process variables under operating mode change. Matrix modifications at sample 1000 affect all variables through their dynamic interconnections. ....	68
Figure 29: SPE values for the linear autoregressive system with bias (left) and with operating mode change (right). The horizontal dashed line indicates the SPE control limit $\lambda$ , obtained from the training period. ....	69
Figure 30: Side-by-side comparison of NNV contributions for the linear autoregressive system with bias (left) and with operating mode change (right). ..	69
Figure 31: SPE values for the selected TEP fault scenarios considered in this study (IDV4, IDV7, and IDV14). The horizontal dashed line indicates the SPE control limit $\lambda$ , obtained from the training period. ....	72
Figure 32: Average NNV contributions for: (a) IDV4, (b) IDV7 and (c) IDV14 through all six TEP operating modes. ....	73
Figure 33: SPE values for TEP, during operating mode transitions from Mode 1. The horizontal dashed line indicates the SPE control limit $\lambda$ , obtained from the training period. ....	74
Figure 34: NNV contributions for operating mode transitions: (a) Mode 1 to Mode 2, (b) Mode 1 to Mode 3, (c) Mode 1 to Mode 4, (d) Mode 1 to Mode 5, and (e) Mode 1 to Mode 6. ....	76
Figure 35: Confusion matrices for binary classification using NNV contributions. ....	79
Figure 36: ROC curves for fault vs. operating mode changes classification using NNV contributions. ....	79
Figure 37: SPE values for Real Data 1, during a fault in one of the input variables. The horizontal dashed line indicates the SPE control limit $\lambda$ , obtained from the training period. ....	81
Figure 38: NNV contributions for Real Data 1. ....	82

Figure 39: SPE values for Real Data 2, during an operating mode change. The horizontal dashed line indicates the SPE control limit  $\lambda$ , obtained from the training period..... 82

Figure 40: NNV contributions for Real Data 2. .... 82

# List of Tables

Table 1: Summary of evaluated cases for Instance Selection.....	36
Table 2: Manipulated variables in the TEP process.....	38
Table 3: Process measurements in the TEP process. ....	38
Table 4: Description of considered faults in the TEP. ....	39
Table 5: Results for anomaly detection using full and reduced datasets for training. .....	42
Table 6: Summary of the results obtained for the evaluated cases. ....	56
Table 7: Steady-state different operation modes of the TEP dataset. ....	71
Table 8: Average values of the normalized dispersion index $S$ computed from NNV contribution patterns for the TEP dataset.....	78
Table 9: Normalized dispersion index $S$ computed from NNV contribution patterns for Real Data 1 and Real Data 2.....	83

# List of Abbreviations and Symbols

<b>a</b>	Number of principal components in PCA analysis
<b>ADWIN</b>	Adaptive Sliding Window
<b>AE</b>	Autoencoder
<b>AI</b>	Artificial Intelligence
<b>ANN</b>	Artificial Neural Network
<b>BN</b>	Bayesian Network
<b>C</b>	Contribution vector
<b>CPD</b>	Change Point Detection
<b>CPV</b>	Cumulative Percentage Variance
<b>CSTR</b>	Continuous Stirred Tank Reactor
<b>CUSUM</b>	Cumulative Sum
<b>CVA</b>	Canonical Variate Analysis
<b>DPCA</b>	Dynamic Principal Component Analysis
<b>e</b>	Residual or error vector
<b>E</b>	Residual or error matrix
<b>ES</b>	Expert Systems
<b>ESN</b>	Echo State Network
<b>EWMA</b>	Exponentially Weighted Moving Average
<b>FDD</b>	Fault Detection and Diagnosis

FDI	Fault Detection Index
FL	Fuzzy Logic
GMM	Gaussian Mixture Model
$h$	Data-driven model
HMM	Hidden Markov Model
ICA	Independent Component Analysis
IS	Instance Selection
$k$	Length of the current in-control segment from ADWIN
KIKA	Kernel Independent Component Analysis
KPCA	Kernel Principal Component Analysis
KPI	Key Performance Indicator
KPLS	Kernel Partial Least Squares
LCL	Lower Control Limit
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
MSPC	Multivariate Statistical Process Control
$N$	Number of observations
NKGMM	Nonlinear Kernel Gaussian Mixture Model
$M$	Number of variables
NNV	Nearest Normal Value

$p$	Number of predicted variables
$P(\cdot)$	Probability
$\mathbf{P}$	Loading matrix
$\rho$	Performance measure
PCA	Principal Component Analysis
PH	Page-Hinkley Test
PLS	Partial Least Squares
QTA	Qualitative Trend Analysis
RF	Random Forest
RT	Regression Tree
$s$	Instance Selection subset
$S$	Dispersion index
S-Chart	Shewhart Chart
SPC	Statistical Process Control
SPE	Squared Prediction Error
SVM	Support Vector Machine
$t$	Time index
$\mathbf{T}$	Score matrix
$\mathbf{X}$	Data matrix of measured variables
$\hat{\mathbf{X}}$	Reconstructed values of $\mathbf{X}$
$\mathbf{x}(t)$	Observation vector at time $t$

$\mathbf{y}$	Vector of measured output variables
$\hat{\mathbf{y}}$	Vector of model-predicted outputs
$\lambda$	SPE control limit
UCL	Upper Control Limit
Z-Chart	Z-Score Chart

# Chapter 1

## Introduction

### 1.1 General aspects

The increasing demand for process safety and sustainability has led to progressively more complex industrial plants (VENKATASUBRAMANIAN, RENGASWAMY, KAVURI, 2003). This complexity results in greater instrumentation degree as well as more advanced control and automation systems. In the oil and gas industry, for example, a typical platform may contain over 30,000 sensors that continuously produce information (MCKINSEY & COMPANY, 2017), creating 1 to 2 TB of data every day (CISCO, 2017).

In such complex, sensor-intensive systems, small deviations can propagate quickly, making early detection of anomalous behavior critical requirement and motivating the adoption of fault detection and diagnosis practices. Early detection enables the plant to operate in a reliable region, preventing faults from progressing and lowering the risk of losses and downtime (ALDRICH, AURET, 2013). According to VENKATASUBRAMANIAN, RENGASWAMY, KAVURI (2003), most of the problems regarding operational units, today, are still caused by the human factor. This results in losses of more than US\$ 2,000,000,000 per year to the industry because of unscheduled downtime (ABERDEEN, 2016).

These operational and economic consequences are consistent with the growth of research activity in process monitoring reported in the literature. Considering the context of Brazil's oil and gas industry, for example, investments in R&D — which in 2022 summed up to approximately R\$ 4.4 billion — have been experiencing a constant increase in the proportion of projects dedicated to digital solutions. According to the Brazilian National Agency of Petroleum, Natural Gas and Biofuels

(ANP), this share exceeded 30% in 2023, compared with less than 2% in 2017 (ANP, 2023).

As much as those numbers might bring expectations regarding new developments and applications, some challenges must be addressed. One shortcoming of rapid and decentralized progress is that few studies address the issues that precede or follow the monitoring step. As shown in Figure 1, very few works present practical applications in industrial environments and, even for those works, the focus usually relies on the performance of the techniques in diagnosing documented failures (ALAUDDIN, KHAN, *et al.*, 2018). This conclusion is also supported by FEISA, GEBREMEDHEN, *et al.*, (2025), GARCIA, RIOS-COLQUE, *et al.*, (2025), and SCHLEGL, TOMASELLI, *et al.*, (2022). Collectively, these works emphasize the importance of technique selection but relegates essential issues such as the formal definition of the problem to be solved, the relevant variables to the system, the training period, and the frequency of retraining to a second plan.

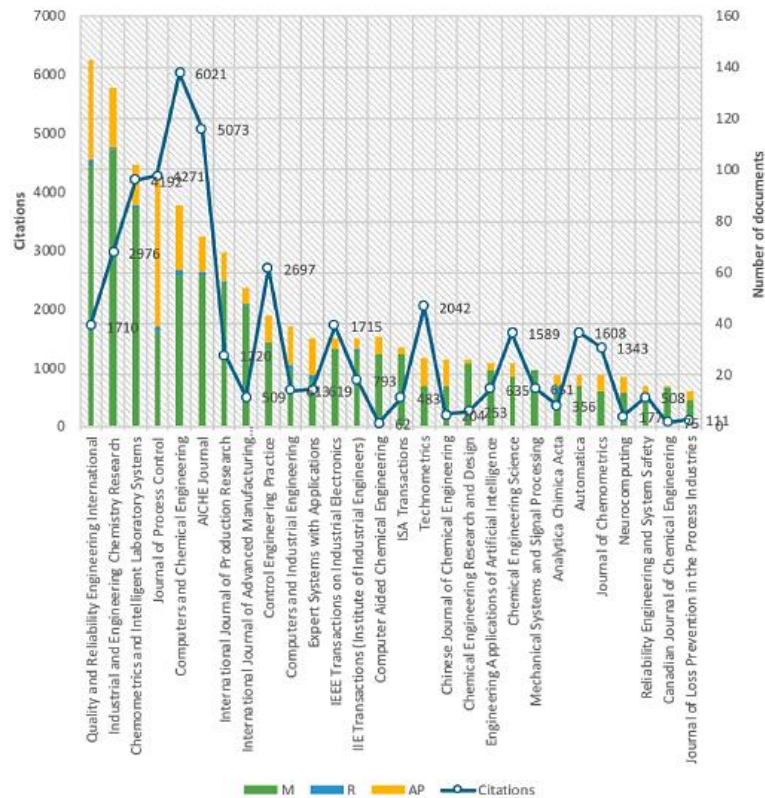


Figure 1: Categories of publications for data-driven fault detection and diagnosis by selected sources from 2017 to 2018. M = algorithm and methods development; R = review article; AP = applications (ALAUDDIN, KHAN, *et al.*, 2018).

## 1.2 Objective

The objective of this thesis is to develop and apply data-driven fault detection and diagnosis algorithms for monitoring real-world systems. In this thesis, the monitoring tasks are primarily formulated as regression problems, since industrial process data are typically composed of continuous measurements used for prediction, reconstruction, and residual-based anomaly detection. While process monitoring is well established in the scientific literature, few studies have focused on the challenges that arise before or after the monitoring step, as illustrated in Figure 1. These questions, however, are critical for practical applications and are often left to the scrutiny of the operation crew.

The current study aims to describe, discuss, and provide solutions for two fundamental concerns with process monitoring applied to real industrial settings:

- i. Selecting appropriate data for model training.
- ii. Distinguishing true process faults from model inadequacy caused by operating mode changes.

## 1.3 Nomenclature and terminology

This thesis follows a deployment-oriented view of data-driven process monitoring that separates (i) pre-monitoring decisions made before online detection and (ii) post-detection actions taken after an alarm is triggered. In this context, some terms appear interchangeably throughout the text. Unless stated otherwise, the expressions below should be interpreted as having the same meaning.

The term fault detection is used as the preferred expression for identifying abnormal behavior from monitoring statistics. In several contexts, the text may also use anomaly detection, abnormal event detection, or alarm generation to refer to the same detection step. When the discussion addresses both detection and diagnosis jointly, the phrase fault detection and diagnosis (FDD) is used.

The preferred term for the post-detection interpretation stage is fault diagnosis. The expressions diagnosis, diagnostic analysis, and contribution analysis are used to describe the same family of approaches in this thesis that attribute a detected deviation to one or more variables.

The term operating mode change refers to an operational shift that alters the process regime while remaining operationally plausible. The expressions mode transition, operating mode transition, operational change, and regime change are used interchangeably to denote the same phenomenon.

The term concept drift is used when referring to time-evolving changes in data distribution that might affect model performance after deployment. In this thesis, model drift, and model inadequacy are used as equivalent descriptions of the same condition.

The expressions training data, training dataset, and training period denote the historical data used to estimate the baseline model. When time localization is emphasized, training window is used. The term instance selection refers to the procedure of selecting representative training instances by identifying operational regions and removing redundant or non-representative samples.

The primary monitoring statistic in this thesis is the Squared Prediction Error (SPE). The expressions monitoring statistic, monitoring index, and fault detection index may be used to refer to the same scalar indicator.

## **1.4 Thesis structure:**

This thesis is organized into five chapters as follows: Chapter 1 contextualized the monitoring problem and its application to real world cases. Chapter 2 presents a literature review on data-driven process monitoring, emphasizing industrial applications and introducing a general monitoring framework. Chapter 3 presents the Instance Selection Library (ISLib) for systematically identifying representative training datasets and its applications.

Chapter 4 introduces the Nearest Normal Value (NNV) diagnostic framework for distinguishing process faults from operational changes, validated through benchmarks and real industrial data. Finally, Chapter 5 synthesizes contributions, discusses the complementary nature of both methodologies, and presents conclusions and future research directions.

# Chapter 2

## Literature review

This chapter will cover the main concepts in data-driven process monitoring. For this goal, a review of existing literature will be presented, with a focus on real-world applications. The chapter begins with a brief history of process monitoring, followed by the standard framework used in real monitoring applications. Each step is then described, and its challenges discussed. It must be clear that the present review does not intend to provide a thorough overview of the field, given the relatively large number of surveys that have been published in the open literature in this area (ALAUDDIN, KHAN, *et al.*, 2018, ALCALA, JOE QIN, 2011, ALDRICH, AURET, 2013, LEITE, ANDRADE, *et al.*, 2024, MD NOR, CHE HASSAN, *et al.*, 2020, PARK, FAN, *et al.*, 2020, TIDRIRI, CHATTI, *et al.*, 2016, VENKATASUBRAMANIAN, RENGASWAMY, KAVURI, 2003, YIN, Shen, DING, *et al.*, 2012). For this reason, particular attention is given to issues that have been overlooked in the previous publications, such as the selection of representative training data, and the role of drift in diagnosis and retraining strategies.

### 2.1 A brief history of process monitoring

Any industrial process, no matter how well designed or carefully maintained, will exhibit some inherent or natural variability (MONTGOMERY, RUNGER, 2012). The area of Statistical Process Control (SPC) was created precisely with the objective of separating the inherent noise of the process from the signals that may indicate some sort of abnormal situation (RUSSELL, LEO, *et al.*, 2000).

Quality control charts were first introduced by Shewhart in 1931 and have been widely used as a process monitoring tool since then, by the assumption that a process subjected to its natural variability will remain in a state of statistical control (VENKATASUBRAMANIAN, RENGASWAMY, YIN, *et al.*, 2003). Figure 2 shows an SPC chart applied to a generic Key Performance Indicator (KPI), denoted as  $\mathbf{X}$ , that could represent a quality measurement or any other critical process variable.

By tracking  $\mathbf{X}$  over time relative to its control limits (usually referred to as Upper Control Limit, or UCL, and Lower Control Limit, or LCL), it is possible to determine whether a measurement is under the desired condition. Several decision rules have been proposed to classify observed patterns and flag potential out-of-control behavior, including the Western Electric (MONTGOMERY, RUNGER, 2012) and Nelson rules (NELSON, 1984).

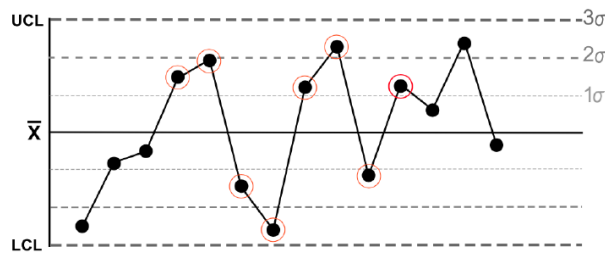


Figure 2: SPC chart of a variable  $\mathbf{X}$ , showing its desirable value (or under control value),  $\bar{\mathbf{X}}$ , Upper Control Limit (UCL) and Lower Control Limit (LCL). Points in red represent a deviation from Nelson's rules (WIKIPEDIA, 2015).

As can be seen, the SPC charts follow the trend of a single variable, and while multiple SPC charts can be created, they will only provide information on that measurement. This can be a concern for processes with a large number of measured and monitored variables since it increases the probability of making a wrong decision based on a faulty diagnosis. MILLER, SWANSON, *et al.* (1998) provide a basic example of an under-control process with 10 independent variables monitored using two standard deviations limit. Assuming a Gaussian distribution, the likelihood of a single variable falling outside of this limit is 5%. However, at any given time, the probability of all ten variables falling inside the normal range is  $0.95^{10} \approx 0.60$ . In other words, there is a 40% chance that at least one variable exceeds

the limit in a sample time, even when the system is controlled. Besides that, the fact that SPC charts do not exploit the correlations that might exist between the variables (ALDRICH, AURET, 2013) can also lead to false diagnosis; that is, the assumption that a certain condition is normal (or abnormal) when, in fact, an abnormal (or normal) event is in progress.

An important milestone in the process monitoring area was the definition of the  $T^2$  statistic, proposed by Hotelling in 1947 as a generalization of the Student's  $t$ -distribution for multivariate analysis of correlated variables. Since then, a range of methods has been developed or adapted to provide more informative representations of process behavior. These representations, according to RUSSELL, LEO, *et al.* (2000), are based on developments from statistical theory, pattern recognition, information theory, and systems theory. In this thesis, such a representation of process behavior is referred to as process model.

VENKATASUBRAMANIAN, RENGASWAMY, KAVURI, *et al.* (2003), in their review of fault detection and diagnosis, proposed a classification of process models based on how they are constructed, distinguishing among process history-based, qualitative model-based, and quantitative model-based approaches, as depicted in Figure 3.

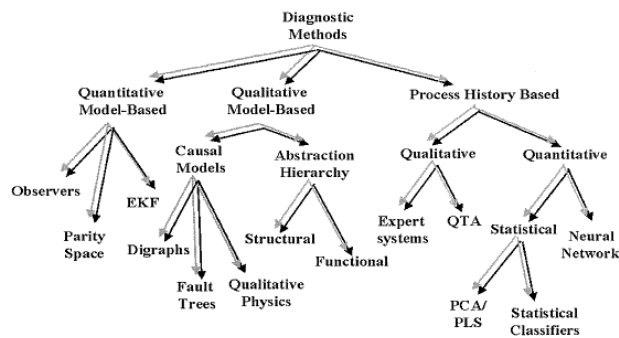


Figure 3: Classification of diagnostic algorithms as proposed by VENKATASUBRAMANIAN, RENGASWAMY, KAVURI, *et al.*, (2003).

Using a different approach, ALDRICH, AURET (2013) proposed a classification of data-driven models based on the attributes summarized in Figure 4, with machine learning methods highlighted in blue.

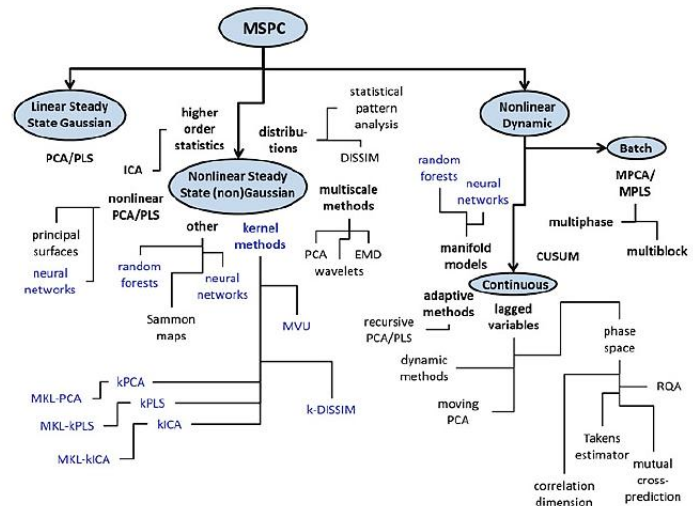


Figure 4: Approaches to multivariate statistical process control (MSPC) (ALDRICH, AURET, 2013).

For the purposes of this thesis, a more suitable classification method is presented by ALAUDDIN, KHAN, *et al.* (2018) and shown in Figure 5. In this scheme, methods are first divided into qualitative approaches — such as expert systems (ES) and qualitative trend analysis (QTA) — and quantitative approaches. Quantitative methods are then subdivided into statistical, hybrid, and AI-based categories. Finally, statistical methods are further classified as univariate, where the classical Shewhart chart approach can be found, and multivariate analyses.

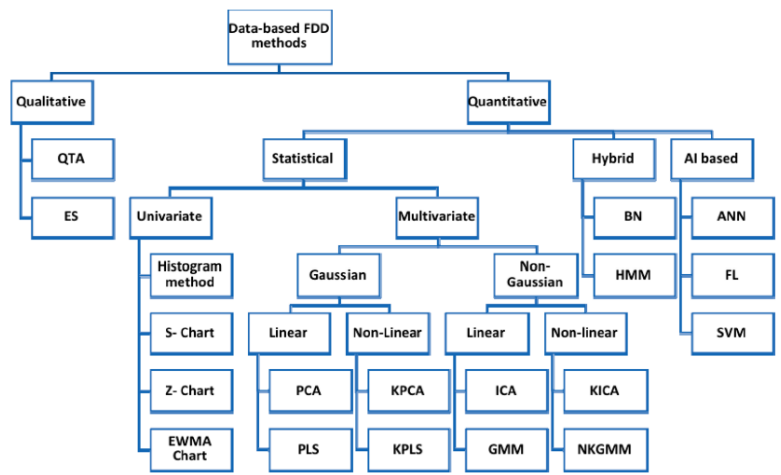


Figure 5: Classification of fault detection and diagnosis methods (ALAUDDIN, KHAN, *et al.*, 2018).

As can be seen, selecting which technique to use as a data-driven model can be a difficult task. Considering the full model-building workflow, including variable selection and the choice of performance metrics for identifying abnormal events, multiple decisions must be made, and key steps can be overlooked. Such steps will be addressed in the next section, in the context of a general framework for process monitoring.

## 2.2 General process monitoring framework

The deployment of a process monitoring task is, by definition, a multidisciplinary effort that involves data scientists, plant operators, process engineers and information technology personnel. This synergy can be crucial since, as aforementioned, the steps required to successfully implement a monitoring application are extensive. Figure 6 proposes a general and detailed framework for a data-driven process monitoring deployment, encompassing each step, its key points, and boundaries for offline and online analyses.

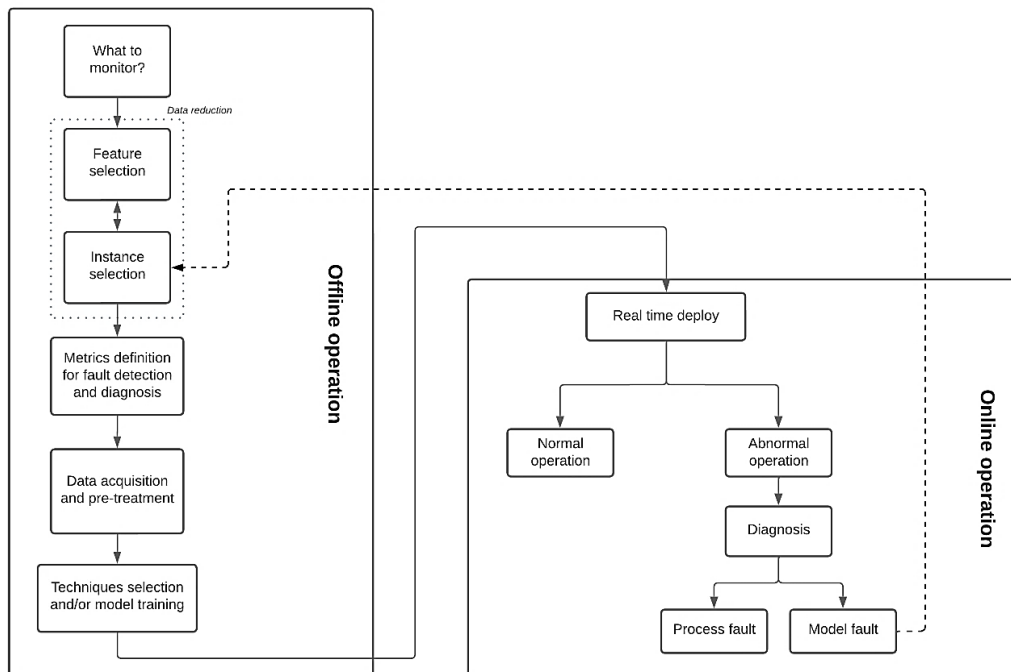


Figure 6: General framework for a data-driven process monitoring application (ANZAI, Thiago K., DE BRITO, *et al.*, 2025).

The first subsystem depicted in Figure 6 represents the problem formulation, that is, what is the monitoring purpose. As trivial as this question may sound, its answer is not straightforward and should be carefully addressed, since it shapes all subsequent steps. For instance, in many practical situations, the objective is stated as monitoring the performance of a critical system; however, this description may be too broad to translate into a well-defined set of features and targets. In other cases, the target variables are clearly specified from the outset. These different settings call for different problem formulations and, consequently, different learning approaches.

Important reviews in the field of process monitoring, such as the ones presented by the works of MD NOR, CHE HASSAN, *et al.* (2020), PARK, FAN, *et al.* (2020), VENKATASUBRAMANIAN, RENGASWAMY, KAVURI (2003), VENKATASUBRAMANIAN, RENGASWAMY, KAVURI, *et al.* (2003) and VENKATASUBRAMANIAN, RENGASWAMY, YIN, *et al.* (2003), also emphasize the importance of the process nature, particularly regarding the presence of relevant dynamic or non-linear effects in the process. This is especially important during the selection of the best suited modeling techniques to be used, but also influences early decision-making steps, like the variable selection stage.

The next subsystem in Figure 6 highlights the selection of representative variables for an accurate model representation. In practice, this step is often delegated to plant operators or local engineers, based on the assumption that the operation team has the most detailed understanding of the process. However, operators and local engineers should not be expected to conduct extensive data analysis to define the final monitoring variable set. Instead, they can provide and suggest an initial set of tags for additional analysis.

Variable selection approaches aim to identify subsets of measured variables that optimize a chosen model performance metric and are typically grouped into three categories: filter-based, wrapper-based, and embedded methods (CLAVIJO, NAYHER, *et al.*, 2021).

Filter-based methods are among the simplest variable selection techniques, since they depend only on the data under analysis. For that reason, they are commonly used during preprocessing, after data acquisition. In this category, perhaps the most used approach is the Pearson correlation, which measures the degree of linear correlation between two variables. The general rule is that, if two measurements are strongly correlated, including both in subsequent analyses adds limited information to the model. The problem regarding this approach is that, even if two measurements present a low Pearson correlation, it is not possible to state that they are not correlated, as the Pearson correlation is essentially linear and bivariate, being unable to capture multivariate nonlinear correlations among variables (LIU, Nan, HU, *et al.*, 2022). Other common filter-based methods include Spearman correlation and mutual information analysis.

Wrapped-based methods use a learning algorithm iteratively to identify a subset of the original variables that optimizes a predetermined performance metric. Candidate subsets can be created by systematically adding or removing variables or by sampling subsets at random. Wrapper-based methods usually outperform filter-based approaches, since they can incorporate multivariate non-linearities in their predictions. Their main limitation is computational cost: depending on the learning algorithm and the size of the initial dataset, the search may require substantial time to converge. A practical advantage is that multiple learning algorithms can be evaluated and compared before selecting the final variable set (CHANDRASHEKAR, SAHIN, 2014, GUYON, ELISSEEFF, 2003).

Embedded methods perform variable selection as part of the model training procedure. A limitation of this approach is that it is restricted to specific model families, such as random forests and LASSO regression. On the other hand, they tend to provide better results than filter-based techniques with lower computational costs when compared to wrapper-based models (CHANDRASHEKAR, SAHIN, 2014).

CLAVIJO, MELO, *et al.* (2021) analyzed different techniques for variable selection. The authors concluded that all considered learning algorithms provided better performances when a causal-based selection procedure was used. This result

underscores the importance of using an appropriate variable selection strategy to improve model quality.

It is also important to note that not all the relevant information of a given process can be measured directly. For instance, if the system dynamics is important for monitoring activity, past information of the process should somehow be considered in the model structure to deal with autocorrelation. This feature selection procedure constitutes another reason why variable selection should be carried out by a broader group of technicians, and not only by plant operators or local engineers, as mentioned before (SIRCAR, YADAV, *et al.*, 2021).

A common feature of data-driven process monitoring is that incoming observations are always compared against an expected behavior for the same time point. If the observed deviation exceeds a certain threshold, it becomes reasonable to assume that the process is not performing the way it was supposed to be. This expected behavior of the process is modeled during the so-called training step, when the process model is built (RUSSELL, LEO, *et al.*, 2000).

Data-driven models are commonly classified by their training characteristics as supervised or unsupervised<sup>1</sup>. In an unsupervised learning framework, a set of unlabeled data, i.e.,  $\{\mathbf{X}\}$  — where  $\mathbf{X}$  is the input (or measured) vector,  $\mathbf{X} \in \mathbb{R}^M$ , and  $M$  is the number of measurements considered — is provided with no distinction between input or output variables. In supervised learning, on the other hand, the dataset comprises distinct set of inputs and outputs  $\{\mathbf{X}, \mathbf{y}\}$ , with  $\mathbf{y}$  representing the output (or predicted) vector,  $\mathbf{y} \in \mathbb{R}^p$ , and  $p$  is the number of outputs, or predicted variables. If the output is continuous, the problem is formulated as regression; when it is discrete, it is formulated as classification (ALDRICH, AURET, 2013). As

---

<sup>1</sup> This text will not cover other kind of learning techniques, such as reinforcement learning or semi-supervised learning. If the reader should be interested, however, a comprehensive text can be found in ALDRICH, AURET (2013).

already mentioned in Chapter 1, the results and discussions presented in this thesis focus primarily on regression-based monitoring problems.

In both supervised and unsupervised settings, historical process data are required to provide a representative snapshot of plant behavior. Indeed, traditional machine-learning algorithms can only be as good as the data they are presented in the first place, meaning that a proper selection of the training period is also of crucial importance.

The step described in Figure 6 as “Instance Selection” deals with the choice of a representative dataset for process monitoring. Just like the variable selection stage, the training period is often chosen by inspecting the plant historian and manually identifying time intervals considered representative. Because this search requires a clear set of variables to guide what “representative” means, instance selection is typically performed in parallel with variable selection (as outlined by the two-way array in Figure 6). In practice, these coupled decisions are among the most time-consuming parts of deploying a data-driven monitoring application. This is consistent with industry surveys and recent studies reporting that data scientists spend a substantial share of their time just cleaning and organizing data, with estimates typically ranging from 45% to 80% (ANACONDA INC., 2020, CROWDFLOWER, 2016, EL MORR, JAMMAL, *et al.*, 2022, GOLENDUKHINA, FELDERER, 2024, ZHAO, GAL, *et al.*, 2024).

As can be noticed, a considerable amount of time should be saved if the second and third steps from Figure 6 could somehow be automated. Finding a representative training period, however, might constitute a challenging task because it requires screening and mining large volumes of historical data. In applications with hundreds of measurements, it should be expected that some information might be left out or, what is worse from the model perspective, wrongly included in the final dataset.

Among the techniques to better condition the data, instance selection comprises a set of procedures designed to extract a representative subset from the original training dataset (ARNAIZ-GONZÁLEZ, DÍEZ-PASTOR, *et al.*, 2016b).

According to LIU, Huan, MOTODA, (2002), instance selection serves several purposes: it enables more computationally demanding models to handle larger datasets, focusing on the most relevant subsets of the data to potentially improve model performance, and removes outliers or redundant instances that add limited value to the learning task.

Historically, instance selection has been mostly used in classification problems, where outputs are discrete and the set of possible labels is limited (ALDRICH, AURET, 2013). In regression, different strategies are required because the output is continuous and, therefore, with an arbitrary number of possible values (ARNAIZ-GONZÁLEZ, DÍEZ-PASTOR, *et al.*, 2016b). Besides that, there also seems to be a consensus that the predictor becomes better when the number of observations used to train the regression model increases, in such a way that reducing the dataset size would not make sense for regression. This last statement, however, can only be true if the data quality remains the same through all observations. In real-world scenarios, with processes subject to random variations and unexpected disturbances (VENKATASUBRAMANIAN, RENGASWAMY, YIN, *et al.*, 2003), transmitters prone to losing calibration, communication failures, poor control design, among other known limitations, this quality can hardly be guaranteed. Moreover, taking into consideration that most of the monitoring activity will be conducted by engineers and not data scientists, it becomes imperative, for the sake of the success of any industrial deployment, that instance selection procedures are not only executed but also presented in the best way possible for the final users to interpret them and take actions.

When instance selection is applied appropriately, the training dataset can be reduced, and the predictive capability of the resulting models may even improve (ARNAIZ-GONZÁLEZ, DÍEZ-PASTOR, *et al.*, 2016a). In other words, after instance selection is applied on a tabular dataset  $\mathbf{X}^{N \times M}$  with  $N$  observations and  $M$  measurements, it is expected that  $\wp(h(\mathbf{X}^{N \times M})) \cong \wp(h(\mathbf{X}^{s \times M}))$ , where  $\wp$  is a performance measure for the learning algorithm  $h$ , and  $\mathbf{X}^{s \times M}$  is a subset of  $\mathbf{X}^{N \times M}$  with  $s < N$ .

As an approach to apply instance selection to regression problems, GUILLEN, HERRERA, *et al.*, (2010) used the concept of mutual information, normally applied in the field of feature selection, to obtain the best input vectors to be used during the model training. STOJANOVIĆ, BOŽIĆ, *et al.*, (2014) also used mutual information to select training instances for long-term time series prediction. By selecting instances that shared a large amount of mutual information with the current forecasting instance, the authors could reduce the error propagation and accumulation in their case studies. ARNAIZ-GONZÁLEZ, DÍEZ-PASTOR, *et al.*, (2016a) adapted the well-known DROP (Decremental Reduction Optimization Procedure) family of instance selection methods for classification to regression. In a different approach, ARNAIZ-GONZÁLEZ, DÍEZ-PASTOR, *et al.*, (2016b) applied the popular classification noise filter algorithm ENN (Edited Nearest Neighbor) to regression problems. To achieve this, the authors discretized the output variable, framing the problem as a classification task. SONG, LIANG, *et al.*, (2017) proposed a new method, called DISKR, based on the KNN (K-Nearest Neighbor) Regressor, which removes outlier instances and then creates a ranking sorted by the contribution to the regressor. DISKR was applied to 19 cases and showed similar prediction ability compared to the full-size dataset model. More recently, KORDOS, BLACHNIK, *et al.*, (2022) developed an instance selection procedure based on three serial steps. The first step decomposes the data using fuzzy clustering; the second uses a genetic algorithm to perform instance selection for each cluster separately; the combined results are weighted to provide a single output. The method was applied to several open benchmarks and generally improved the predictive model performance while reducing its size. LI, Chuang, MAO, (2023) presented a novel noise filtering technique specifically designed for regression problems with real-valued label noise. The proposed method introduces an adaptive threshold-based noise determination criterion and a noise score to identify and eliminate noisy samples while preserving clean ones. Its performance was evaluated across various controlled scenarios, including synthetic datasets and public regression benchmarks, demonstrating superiority over state-of-the-art methods.

It is worth noting that the studies mentioned above used well-known open datasets from various sources and fields. From the process monitoring perspective

and all the challenges it poses for data preprocessing, a relevant literature gap still exists when applying such methods to real industrial data. Dealing with a representative choice of periods for training purposes is one of the primary objectives of this thesis and shall be explored further in the next chapter.

As mentioned before, regression-based process monitoring typically relies on the difference between the model output and the corresponding plant measurement to infer process health. Consequently, different metrics can be used to quantify this error, and their choice can influence the detection performance (HALLGRÍMSSON, NIEMANN, *et al.*, 2020a, XIAO, WANG, *et al.*, 2013). During model development, differences between the model's output and the ground-truth values through all the samples are quantified by indices such as the Mean Squared Error (MSE) or the Mean Absolute Error (MAE). During online monitoring, deviations from normal behavior are typically quantified using indices such as the Squared Prediction Error (SPE), which measures the magnitude of residuals between observed and model-predicted values at a given time instant:

$$\text{SPE} = \sum_{i=1}^M (X_i - \hat{X}_i)^2 \quad \text{Eq. 1}$$

where, for unsupervised models,  $\mathbf{X}$  is the input vector and  $M$  is the number of considered measurements. In Eq. 1, the superscript  $\hat{\phantom{x}}$  indicates the predicted values from the learned model  $h$ , or, in other words:

$$\hat{\mathbf{X}} = h(\mathbf{X}) \quad \text{Eq. 2}$$

In a more concise form, Eq. 1 can be rewritten in matrix form as:

$$\text{SPE} = \mathbf{e}^T \mathbf{e} \quad \text{Eq. 3}$$

where  $\mathbf{e}$  is error the vector  $(\mathbf{X} - \hat{\mathbf{X}})$ , or the difference between the measured and predicted values, and  $\mathbf{e}^T$  is its transpose.

An abnormal event is detected if the SPE value exceeds a predefined threshold ( $\lambda$ ), normally defined as a percentile value from the training or validation

period (CLAVIJO, N., MELO, *et al.*, 2019). This limit value is then kept constant during the online monitoring and serves as a boundary for the normal operation condition or for an alarm detection heuristic. Within the workflow outlined in Figure 6, this step completes the definition of the monitoring index and its decision rule, enabling the critical stages related to data acquisition and model training.

As reflected in the literature, the “Techniques selection and/or model training” stage from Figure 6 is probably the step where most attention is paid to when referring to data-driven process monitoring literature. CLAVIJO, N., MELO, *et al.*, (2019) used a data-driven approach to create a real-time monitoring application of gas metering in an oil onshore process plant. OLIVEIRA-JUNIOR, DE ARRUDA PEREIRA, (2020) developed a soft sensor for the Total Oil and Grease (TOG) value using data from an oil production platform. LUO, XIE, *et al.*, (2020) proposed a Deep Neural Network for data-driven monitoring in an industrial methanol plant, effectively handling time dependency in the processes. CORTÉS-IBÁÑEZ, GONZÁLEZ, *et al.*, (2020) presented a preprocessing methodology for getting quality data in a crude oil refining process. In the framework proposed by the authors, data reduction was presented through the optics of outlier elimination and feature selection. LEMOS, CAMPOS, *et al.*, (2021) used an Echo State Network (ESN) in an oil and gas process plant to monitor the quality of critical flowmeters. KAZEMI, MASOUMIAN, *et al.*, (2024) introduced a neural-based adaptive PCA framework to mitigate limitations of traditional PCA in time-varying settings; the approach was validated on a CSTR and water resource recovery facilities and improved fault detection and diagnosis. CUI, LI, *et al.*, (2025) proposed a Mixture of Autoencoders for fault detection in multimode industrial processes characterized by complex chemical reactions and nonlinear behavior.

Accordingly, this thesis does not aim to provide an exhaustive comparison of learning algorithms or preprocessing alternatives. Instead, the selected methods are presented and justified as components of the overall monitoring pipeline. For completeness, the machine-learning methods used throughout the thesis are summarized in Appendix A, with descriptions of the principal models employed in this work in the context of process monitoring.

Once a model has been trained and deployed, the proposed workflow transitions from offline development to online monitoring. In this stage, the model effectively audits process observations and returns real-time performance indices such as SPE.

In an unsupervised learning algorithm, with all the  $\mathbf{X}$  inputs contributing to the fault detection index value, it is reasonable to think, by observing Eq. 1, that, once the SPE exceeds its limit, it is possible to somehow trace the input that had the most impact on the detection index value. This approach isolates abnormal process variables by analyzing the contribution of each measurement to the SPE (HALLGRÍMSSON, NIEMANN, *et al.*, 2020a), and is the main reason why residual statistics are widely used not only for detection, but also for diagnosis in multivariate process monitoring frameworks. Defining the contribution of a variable  $i$  as:

$$C_i = (X_i - \hat{X}_i)^2 \quad \text{Eq. 4}$$

Eq. 1 can be rewritten, for an unsupervised problem, as:

$$\text{SPE} = \sum_{i=1}^M C_i \quad \text{Eq. 5}$$

Variables with large contributions are assumed to no longer be consistent with normal operating conditions (HALLGRÍMSSON, NIEMANN, *et al.*, 2020a), and are, therefore, identified as potential responsible for the abnormal event.

Although fault diagnosis methods aim to identify the variables responsible for detected anomalies, it is important to note that not every alarm necessarily corresponds to a true process fault. In data-driven monitoring, the “Real time deploy” stage depicted in Figure 6 is commonly seen as the last stage of the monitoring application, meaning that every new measurement that has an SPE value beyond its thresholds is considered to be suspicious. This is, however, not necessarily true. Unlike classical parameter estimation — where experimental data serve as the reference to which the model must be adapted — most process monitoring applications invert this relationship: the model represents the expected normal behavior, and deviations in the data are interpreted as process anomalies

rather than model deficiencies. Since most of the processes in chemical engineering are dynamic and, thus, time dependent, it is reasonable to think that a model trained during the first deployment might not be able to perform the same way during the whole life span of the process. In fact, the performance of a model is likely to deteriorate (YAN, 2020). In the machine-learning literature, this change in data distribution over time is commonly referred to as concept drift (ALIPPI, BORACCHI, *et al.*, 2016).

Concept drift is defined as a change in the data-generating distribution over time, where the joint probability distribution ( $P(\mathbf{X}, \mathbf{y})$ ) of input features ( $\mathbf{X}$ ) and the target variable ( $\mathbf{y}$ ) shifts (HINDER, VAQUET, *et al.*, 2024, LOBO, LAÑA, *et al.*, 2023). Consequently, models trained on historical data may experience significant performance degradation when deployed in production, demanding continuous monitoring and retraining to maintain their predictive accuracy (PHAM, PREMKUMAR, *et al.*, 2025). In industrial process monitoring, this creates the challenge of distinguishing whether control limit violations indicate real process faults or operational changes that have turned the baseline model obsolete.

According to GAMA, ŽLIOBAITĖ, *et al.*, (2014), change detections algorithms can be classified as (i) methods based on sequential analysis; (ii) methods based on control charts; (iii) methods based on differences between windows and (iv) heuristic methods.

Among the methods based on sequential analysis, the CUmulative SUM (CUSUM) and its variant, the Page-Hinkley (PH) test, are among the most common (AGRAHARI, SINGH, 2021). The CUSUM method aims to detect changes in the variable mean and can be defined as:

$$g_{t+} = \max(0, (\text{SPE}_t - \delta) + g_{t-1}) \quad \text{Eq. 6}$$

where  $g_t$  is the value of the CUSUM test at instant  $t$ , and  $\delta$  is the magnitude of the allowable changes. Any residue above  $\delta$  generates a value that contributes positively to  $g_t$ . Thus, a low limit for  $\delta$  allows faster change detection but also results in more

false alarms. Negative deviations from the mean can also be calculated similarly through Eq. 7.

$$g_{t-} = \min(0, (\text{SPE}_t - \delta) + g_{t-1}) \quad \text{Eq. 7}$$

The CUSUM test generates an alarm when the absolute value of  $g_{t+}$  or  $g_{t-}$  exceeds a pre-established limit. The method is computationally efficient and only requires updating recursive cumulative statistics, rather than storing the full historical data. However, its accuracy depends on the choice of parameters  $\delta$  and the detection threshold (GAMA, ŽLIJBAITĚ, *et al.*, 2014).

The PH algorithm uses as a criterion the test defined by Eq. 8.

$$m_T = \sum_{t=1}^T (\text{SPE}_t - \overline{\text{SPE}}_t - \delta) \quad \text{Eq. 8}$$

where  $\overline{\text{SPE}}_t$  is the population mean of the residue between intervals  $t = 1$  and  $t = T$  and  $\delta$ , just like in CUSUM, specifies the tolerance to change. The PH test generates an alarm if the value of the difference between  $m_T$  and the minimum value of  $m_T$  between  $t = 1$  and  $t = T$  exceeds a certain threshold (GAMA, ŽLIJBAITĚ, *et al.*, 2014).

Methods based on SPC charts verify whether the residue, the error between predicted and measured values, is within a normal range. Like its application for univariate monitoring, SPC charts for drift detection use estimates of the time series variance to construct this range. The Exponentially Weighted Moving Average (EWMA) technique, adapted by ROSS, ADAMS, *et al.* (2012), for example, estimates the residue values by progressively down-weighting older data according to Eq. 9:

$$\begin{aligned} Z_0 &= 0 \\ Z_t &= (1 - \alpha)Z_{t-1} + \alpha\text{SPE}_t, \quad t > 0 \end{aligned} \quad \text{Eq. 9}$$

where  $Z_t$  is the EWMA estimator at instant  $t$  and  $\alpha$  is the parameter that controls the weight given to the most recent values of  $\text{SPE}_t$  compared to older ones. ROSS, ADAMS, *et al.* (2012) recommend  $\alpha$  values between 0.1 and 0.3.

It can be shown that, independent of the distribution of the  $SPE_t$ , the mean and standard deviation of  $Z_t$  are (ROSS, ADAMS, *et al.*, 2012):

$$\begin{aligned}\mu_{Z_t} &= \mu_t \\ \sigma_{Z_t} &= \sqrt{\frac{\alpha}{2-\alpha}(1-(1-\alpha)^{2t})}\sigma_{SPE}\end{aligned}\tag{Eq. 10}$$

The EWMA test generates an alarm when the estimate value of  $Z_t$  exceeds a given threshold based on the standard deviation.

$$Z_t > L\sigma_{Z_t}\tag{Eq. 11}$$

where  $L$  is the control limit and must be chosen in order to ensure the performance of the detector.

The third class method described by GAMA, ŽLIOBAITĚ, *et al.* (2014) is based on differences between windows. These methods usually utilize a fixed reference window for summarizing the past information and a sliding window for summarizing the most recent data (PESARANGHADER, VIKTOR, *et al.*, 2017). Distributions over the two windows are compared using statistical tests with the null hypothesis stating that the distributions are equal. If the null hypothesis is rejected, a change is declared at the start of the recent window (GAMA, ŽLIOBAITĚ, *et al.*, 2014).

The ADaptive sliding WINDOW (ADWIN) (BIFET, GAVALDÀ, 2007) is a change detector and estimator that keeps a sliding window that automatically adjusts its size based on the observed rate of change in the data. Unlike traditional approaches requiring fixed window sizes, ADWIN dynamically grows the window during stationary periods to improve estimation accuracy and shrinks it when distributional changes occur to discard obsolete data. The core algorithm operates by maintaining a window  $W$  of recent data points and applying a statistical test to detect when two sufficiently large subwindows exhibit distinct average values. When the observed difference exceeds a calculated threshold, the older subwindow is dropped, signaling a detected change. Figure 7 shows the algorithm for ADWIN and its output for a dataset with an abrupt change in  $t = 1000$ .

```

ADWIN: ADAPTIVE WINDOWING ALGORITHM
1 Initialize Window  $W$ 
2 for each  $t > 0$ 
3   do  $W \leftarrow W \cup \{x_t\}$  (i.e., add  $x_t$  to the head of  $W$ )
4   repeat Drop elements from the tail of  $W$ 
5     until  $|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}| \geq \epsilon_{cut}$  holds
6     for every split of  $W$  into  $W = W_0 \cdot W_1$ 
7   output  $\hat{\mu}_W$ 

```

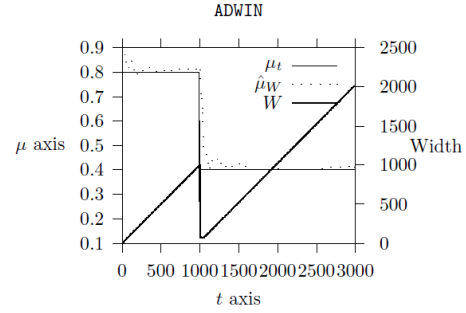


Figure 7: ADWIN algorithm (left) and output (right) for an abrupt change. Figure adapted from (BIFET, GAVALDÀ, 2007).

Drift detection in real-world applications is an expanding area, and although well-established techniques such as CUSUM, proposed back in 1954, are still applicable, many methods are being developed and/or adapted to the reality of drift detection. Comparative studies (GONÇALVES, DE CARVALHO SANTOS, *et al.*, 2014) and comprehensive surveys (BAYRAM, AHMED, *et al.*, 2022, GAMA, MEDAS, *et al.*, 2004, GEMAQUE, COSTA, *et al.*, 2020) have evaluated drift detectors across various contexts, noting that most methods deal with classification problems and that more research is needed for regression and process monitoring applications.

Through a Bayesian perspective, the drift problem can be viewed as the rupture in the joint probability of targets and features of a given dataset (ZACHOS, 2018). Since  $P(\mathbf{X}, \mathbf{y}) = P(\mathbf{y}|\mathbf{X}) \times P(\mathbf{X})$ , drifts can arise from changes in the feature distribution  $P(\mathbf{X})$  (virtual drift) or in the conditional probability  $P(\mathbf{y}|\mathbf{X})$  (real drift). Figure 8 illustrates these types of drift from a classification problem perspective.

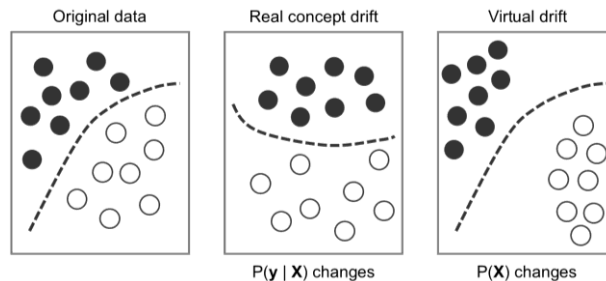


Figure 8: Types of drift in a classification setting. The axes represent the feature space  $\mathbf{X}$ , and  $\mathbf{y}$  denotes the class label (indicated by the circle colors). The dashed line represents the decision boundary of the model.

As shown in Figure 8, real drifts change the boundaries between classes (represented by differently colored circles), making the model  $h$  obsolete (GAMA, ŽLIOBAITĚ, *et al.*, 2014). This means that, for monitoring purposes, not all drifts should become a trigger for a new model to be trained. Additionally, not all joint probability changes are due to a new concept. For instance, outliers should not be considered as a change in the process, but could, otherwise, represent a problematic or undesired operation.

As depicted in Figure 9, fault detection systems are conventionally interpreted as process faults identifiers, while drift detection methods focus on model performance degradation. However, both types of alarms can arise from either root cause, creating diagnostic ambiguity. In other words, a fault detection alarm may indicate model inadequacy due to operational changes, while a drift detection alarm may signal an undetected process fault.

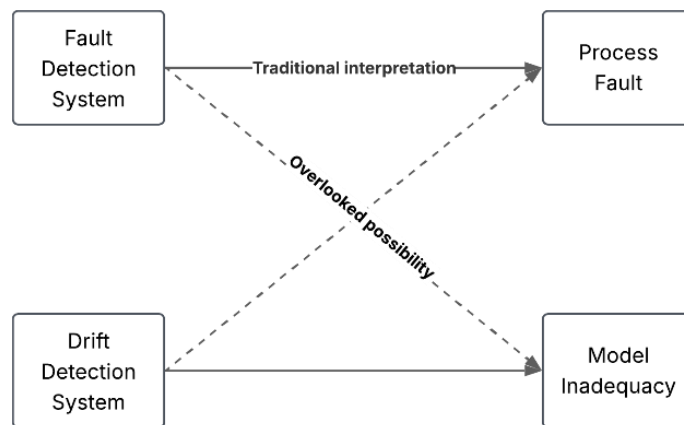


Figure 9: Traditional monitoring paradigm showing conventional interpretations (solid arrows) and overlooked diagnostic possibilities (dashed arrows).

For this reason, it is important to approach the retraining strategy as a part of the fault detection and diagnosis problem. To the best of the author’s knowledge, few works have explicitly examined the drift issue through the optics of fault diagnosis, by using the fault detection index as a trigger. For example, LUGHOFER, (2018) proposed drift-aware monitoring and evolving data-driven fault detection schemes, in which concept drift is handled through model adaptation and

incremental learning to preserve detection performance. RAMÍREZ, SILVA, *et al.*, (2025) introduced a method in which fault detection, model drift and change-point testing are unified via nonparametric mutual information estimators, providing a detection criterion that does not depend on specific learning algorithms. LOBO, LANA, *et al.*, (2023) investigated the relationship between concept drift and predictive uncertainty in industrial AI, showing that, in online regression tasks, an increase in prediction error tends to follow drift events. JOURDAN, BAYER, *et al.*, (2023) presented a framework for handling concept drift in deep learning applications for process monitoring, evaluated on a real-world CNC milling process. Their approach used accelerometer data and addressed multiple realistic drift types, demonstrating the necessity of model adaptation in industrial settings. YIN, Min, (2025) developed a drift-aware streaming predictive maintenance system specifically designed for semiconductor equipment. The system detected both rapid and slow drifts, with online drift monitors triggering cautious adaptation through shadow evaluation before any model update. This approach reflects the standard paradigm: drift is detected, and then the model is carefully adapted to maintain performance without overreacting to transient changes. BAYRAM, AHMED, *et al.*, (2024) proposed an Adaptive Data Quality Scoring Operations Framework using a drift-aware mechanism for industrial applications. The framework introduced a dynamic change detector that actively monitors and adapts to changes in data quality. In all these cases, however, drift is treated primarily as a modeling challenge to be compensated rather than as an explicit diagnostic hypothesis. In other words, drift is consistently framed as an undesirable deviation in process behavior that demands proactive detection followed by model adaptation to restore monitoring reliability. Most articles adopt this detect-and-adapt paradigm, viewing drift not as an inherent process evolution but as an anomaly to be corrected through data model recalibration. The available frameworks do not provide an operational procedure that uses residual indices and model information to decide whether the anomaly is more consistent with a true process or sensor fault, or with model inadequacy under a new operating mode. This problem, depicted by the “Diagnosis” box in Figure 6, is one of the main goals of this thesis.

## 2.3 Concluding remarks

This section presented the importance of process monitoring and the main stages of its deployment, from problem formulation to fault diagnosis and model retraining in data-driven applications. It also introduced a process monitoring framework that positions fault detection as one element within a broader workflow. The proposed strategy, therefore, provides an equivalent degree of importance to steps that precede and succeed the detection itself. In particular, steps related to model training and fault diagnosis were discussed in greater depth.

The analysis of all the works mentioned in this literature review allows for some conclusions:

- Most publications focus on the development and benchmarking of new fault detection techniques, with limited attention to critical decisions outside the detection algorithm itself.
- As highlighted by MELO, CÂMARA, *et al.* (2022), the Tennessee Eastman Process (TEP) database is the most used benchmark for performance quantification of techniques.
- Few studies report real industrial deployments, and among those, most devote little space to the steps that precede model training.
- When the importance of the dataset is discussed, it is usually framed as a feature-selection problem, while decisions about instance selection are treated less systematically.
- Only a small subset of works, such as the ones from BIRODKAR, MOBAHI, *et al.* (2019), CHITTA, ALVAREZ, *et al.* (2019) and VODRAHALLI, LI, *et al.* (2018) explicitly address temporal redundancy and its impact on learning quality.
- Contribution plot is the most used technique for fault diagnosis on regression models.
- Although concept drift is recognized as a source of model performance degradation, it is typically treated as a maintenance/retraining

problem rather than as a competing diagnostic hypothesis after an alarm.

- None of the evaluated works explore the presence of drift as an integral part of the fault diagnosis problem.

## Chapter 3

# Before Detection: Instance Selection for Model Training

By following the steps proposed in Figure 6, one can expect to achieve a process monitoring application running in real time and generating insights about the process. As outlined in the previous chapter, however, going through those steps does not constitute an easy task, since it involves making decisions that have not been usually tackled by most works in the literature. Nevertheless, choosing the most significant set of measurements of a given asset, or performing a proper instance selection procedure can have a huge impact on the final application.

In this section, the problem of data reduction in industrial settings is originally addressed through the lens of instance selection procedures. The proposed methodology, named Instance Selection Library (ISLib), is specifically designed to handle the unique challenges of industrial datasets, such as operational variability, sensor inaccuracies, and the need for interpretable results. ISLib was validated using three datasets, including the Tennessee Eastman Process (TEP) and two real-world oil and gas applications. A detailed technical description of ISLib, including preprocessing steps, and methods included, is presented in Appendix B.

Parts of the methodology and results presented in this chapter were previously published in the article “Instance Selection Strategies to Improve the Performance of Data-Driven Regression Models Applied to Industrial Systems”, by Thiago K. Anzai, Gabriel Marçal de Brito, Tiago S. M. Lemos, Jaqueline Sousa Santos, Pedro H. T. Furtado, and José Carlos Pinto, published in *Processes*, 2025, 13, 2187 (ANZAI, Thiago K., DE BRITO, *et al.*, 2025).

## 3.1 Instance Selection Library (ISLib)

Data reduction is one of the most critical steps inside a data pre-treatment framework. In general, data reduction aims to optimize the overall dimension of the dataset while maintaining or improving its quality. This can be achieved by manipulating the number of columns (or features) and/or rows (or instances) of the datasets. The former is usually referred to as feature selection, and the latter is known as prototype or instance selection. In the field of process monitoring, the amount of published works dedicated to feature selection is overwhelmingly more significant than the number of papers on instance or prototype selection, accounting for approximately 98% of the articles available, according to Google Scholar (GOOGLE LLC., 2024). This can be explained, among other things, by the fact that when dealing with tabular data, i.e., data organized in the form of observations as rows and attributes as columns, the impact of removing columns over rows on the size of the final dataset is more evident. In that way, if the goal is to reduce the dimension of the dataset, acting on the attributes instead of the observations seems more effective. Among the 2% dedicated to instance selection, most works published are concerned with classification problems (KORDOS, BLACHNIK, *et al.*, 2022), leaving a gap in the critical field of regression models.

In the present study, a novel Python library named Instance Selection Library (ISLib) was developed to identify the most important operating regions of a dataset focusing on regression problems. According to the taxonomy presented by GARCÍA, DERRAC, *et al.*, (2012), ISLib performs its search in two steps, as described below.

### 3.1.1 BATCH INSTANCE SELECTION USING RANKED CLUSTERS

During this first step, ISLib performs an unsupervised clustering algorithm to find similar subgroups within the initial data. The goal at this stage is to identify potential operational regions that should be included in the final dataset. Among the various techniques available, partitioning methods are known for their efficiency in

terms of computational complexity and memory requirements (KORDOS, BLACHNIK, *et al.*, 2022). The ISLib library uses the K-Means clustering algorithm (JAMES, WITTEN, *et al.*, 2013) due to its simplicity and fast convergence. Since K-Means assigns observations to the nearest centroid based on the squared Euclidean distance, all variables were previously scaled to avoid dominance by variables with larger numerical ranges. The optimal number of clusters is determined using the elbow method, balancing intra-cluster variance and computational efficiency. In other words, to overcome the drawback of having to define the number of clusters beforehand, in the present application,  $K$  was defined according to an objective function that minimizes the clustering loss function while keeping the number of clusters as small as possible.

The clusters obtained from K-Means are then ranked by their Mean Squared Error (MSE) when trying to predict the other regions. This is achieved by training a model ( $h$ ) with all the observations contained in region  $i$ , with  $i \in (1, \dots, K)$ , and tested against all clusters  $j$ , with  $j \in (1, \dots, K)$  and  $j \neq i$ . Figure 10 depicts an example of the results obtained from this analysis, showing the clusters ranked and one of the  $M$  measurements that compose the dataset to illustrate some key points. As one can see in Figure 10, the model trained only with data from Cluster 1 results in a better performance  $\wp$ , with  $\wp$  in this case being the MSE value when trying to predict data from Clusters 2 and 3. On the other hand, a model trained using only data from Cluster 3 seems to perform worse when presented with data from Clusters 1 and 2.

The model  $h$  used in ISLib assumes the form of a Principal Component Analysis (PCA), if the objective is to provide data for an unsupervised model, or a Regression Tree (RT), if the goal is a supervised model. PCA and RT were chosen due to their robustness and low computational complexity, which align well with industrial datasets characterized by large volumes and high variability. While other lightweight models such as Support Vector Machines (SVM) could be used, PCA has been widely applied in the field of process monitoring (MELO, CÂMARA, *et al.*, 2024b, XIAO, WANG, *et al.*, 2013, YIN, Shen, DING, *et al.*, 2012) due to its ability to effectively reduce data dimensionality while preserving essential

variability, and RT simplifies regression tasks in scenarios where interpretability and speed are important (HASTIE, TIBSHIRANI, *et al.*, 2009).

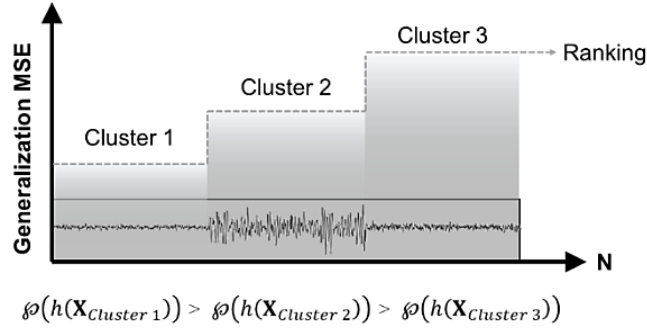


Figure 10: Illustration of the cluster-ranking procedure performed by ISLib during the batch instance selection stage, based on the generalization MSE obtained when models trained on each cluster are tested against the remaining operational regions.

The pseudocode that performs the previously described steps, is displayed in Algorithm 1.

---

**Algorithm 1** Pseudocode for ranking most important clusters in ISLib

---

**Require:** Initial dataset  $\mathbf{X}$ , configuration parameters  $C$

- 1: **for**  $K \leftarrow$  to  $C_{\max\_clusters}$  **do**
  - 2:     Define objective function  $F(K) = \text{Loss}(K - \text{Means}) + \beta K$   
   where  $\beta$  is a regularization parameter
  - 3:     Perform K-Means clustering on  $\mathbf{X}$
  - 4:     **if**  $F(K) > F(K - 1)$  **then**
  - 5:          $K_{\text{final}} \leftarrow K - 1$
  - 6:         Break
  - 7: **for**  $i \leftarrow$  to  $K_{\text{final}}$  **do**
  - 8:     **if** problem is supervised **then**
  - 9:          $h \leftarrow$  Regression Tree ( $C_{RT}$ )
  - 10:         Train  $h$  on  $(\mathbf{X}_i, \mathbf{y}_i)$
  - 11:         Test  $h$  on  $(\mathbf{X}_j)$  where  $j \neq i$
  - 12:          $\text{MSE}_i \leftarrow \frac{1}{N_j} \sum_{j=1}^{N_j} (\mathbf{y}_j - \hat{\mathbf{y}}_j)^2$  where  $N_j$  is the number of  
   observations in  $j$
  - 13:     **else if** problem is unsupervised **then**
  - 14:          $h \leftarrow$  PCA (a), with  $a < M$
  - 15:         Train  $h$  on  $\mathbf{X}_i$
  - 16:         Test  $h$  on  $\mathbf{X}_j$  where  $j \neq i$
  - 17:          $\text{MSE}_i \leftarrow \frac{1}{N_j} \sum_{j=1}^{N_j} (\mathbf{X}_j - \hat{\mathbf{X}}_j)^2$  where  $N_j$  is the number of  
   observations in  $j$
- return** Ordered  $K_{\text{final}}$  clusters by MSE values
-

In a supervised model, MSE is calculated based on the predicted values of  $\mathbf{y}$ , given  $\mathbf{X}$ , using a RT with hyperparameters  $C_{RT}$ . On the other hand, in an unsupervised model, the MSE is calculated using the reconstructed values of  $\mathbf{X}$  obtained from PCA, where  $a$  denotes the number of retained principal components.

The ranked clusters, along with the visual representation of the operational regions, assist in selecting the most desirable subsets to keep. Although ISLib automates the analysis, it is essential to ratify the conclusions with system knowledge. In other words, the final decisions still require human validation and evaluation.

### 3.1.2 INCREMENTAL INSTANCE SELECTION USING A CRESCENT WINDOW STRATEGY

Based on the subsets obtained in the previous step, the goal of this second stage is to determine the minimum training-window size beyond which adding more observations no longer improves the model's predictive performance. For this, several models are generated by increasing the training window size  $s$ , with  $0 < s < N$ , allowing each model to incorporate additional information from the dataset. During this process, the testing period does not remain fixed, and it is considered the fraction of the dataset not used by the training itself. Thus, with each new model, the data used for training has its size increased by a window, while the data for testing decreased by the same amount.

Figure 11 illustrates the adopted procedure, while Algorithm 2 shows its pseudocode. The employed terminology follows the same convention as presented in Algorithm 1. The overlapping time series on the M-axis indicate different features along  $N$  samples under the same training-testing configuration. The plot shows how the ratio between training and testing sets evolves along the  $s$ -axis while keeping their combined size constant.

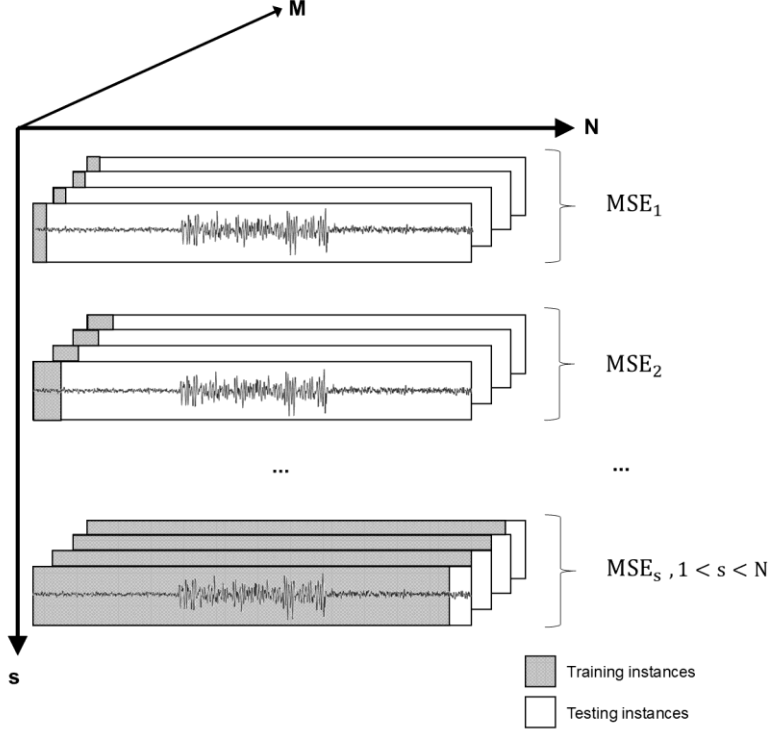


Figure 11: Enlarging window approach for instance selection.

---

**Algorithm 2** Pseudocode for finding the minimum window size in ISLib

---

**Require:** Dataset  $\mathbf{X}$ , configuration parameters  $C$

- 1: **for**  $s \leftarrow C_{min\_size\_window}$  to  $N$ , where  $N$  is the number of rows of the dataset, **do**
  - 2:   **if** problem is supervised **then**
  - 3:      $h \leftarrow$  Regression Tree ( $C_{RT}$ )
  - 4:     Train  $h$  on  $(\mathbf{X}[0:s], \mathbf{y}[0:s])$
  - 5:     Test  $h$  on  $(\mathbf{X}[s+1:N])$
  - 6:      $MSE_s \leftarrow \frac{1}{N-s} \sum (\mathbf{y}[s+1:N] - \hat{\mathbf{y}}[s+1:N])^2$
  - 7:   **else if** problem is unsupervised **then**
  - 8:      $h \leftarrow$  PCA ( $a$ ), with  $a < M$
  - 9:     Train  $h$  on  $(\mathbf{X}[0:s])$
  - 10:     Test  $h$  on  $(\mathbf{X}[s+1:N])$
  - 11:      $MSE_s \leftarrow \frac{1}{N-s} \sum (\mathbf{X}[s+1:N] - \hat{\mathbf{X}}[s+1:N])^2$
  - 12:   store  $MSE_s$
  - 13:  $min\_MSE \leftarrow \min(MSE_s)$
  - 14:  $min\_size\_window \leftarrow$  index of  $min\_MSE$
- return**  $min\_size\_window$
- 

Once the dataset is submitted to the steps presented in Algorithm 2, two general outcomes may occur. Firstly, if the model  $h$  displays strong generalization capabilities, the MSE profile is expected to exhibit a decaying pattern as the training

window expands, indicating improvement in model performance. Conversely, if the dataset presents regions too distinct to allow the model to generalize the process behavior well, the MSE profile is likely to reflect these regions. In such cases, the trend may show sudden decreases as the model learns each new state. These sharp variations in the MSE profile can be interpreted as an offline Change Point Detection (CPD) mechanism. Rooted in the statistical analysis of time series, CPD methods aim to identify time instants at which the underlying data-generating process undergoes structural changes, such as shifts in mean, variance, or correlation structure (TRUONG, OUDRE, *et al.*, 2020). From this perspective, the detected change points delineate segments of approximate stationarity corresponding to distinct operating regimes.

Although CPD originates from classical statistical signal processing and focuses primarily on changes in the distributional properties of the observed data, its use in this context is closely related to the concept of drift detection, which emerged from the machine learning and data stream literature. This conceptual alignment motivates the interpretation of the observed MSE transitions as a bridge between offline CPD and drift detection. More importantly, it establishes a link between the two parts of this thesis: while this first part relies on offline change detection to characterize regime transitions and provide relevant regions for model training, the second part explicitly aims at developing a drift-aware diagnostic framework, in which the presence of drift is directly accounted for in diagnostic decision-making. This connection is further explored and formalized in the second part of this thesis.

It is also worth noting that even though the two steps from ISLib were designed to be executed sequentially, it is possible to perform them either sequentially or independently, depending on the objectives and the initial data quality.

Figure 12 illustrates the structural framework of ISLib, highlighting its two main phases: clustering-based batch instance selection and the enlarging window strategy.

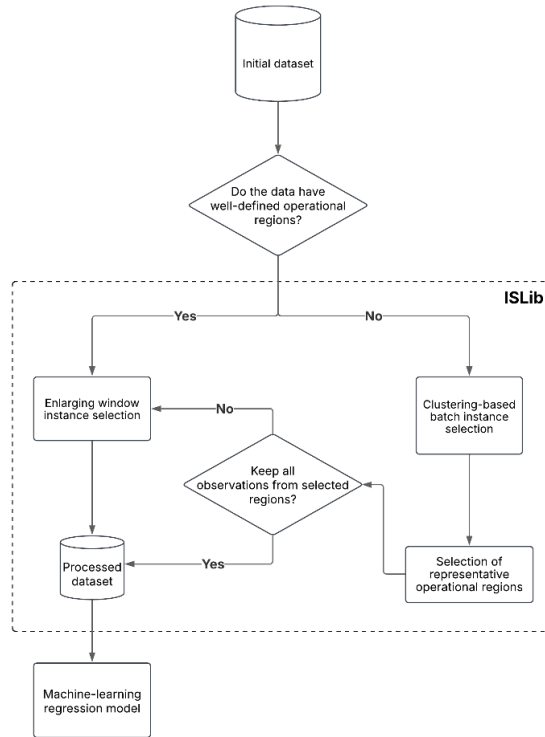


Figure 12: Workflow of dataset processing using the ISLib library, showcasing its two main phases: clustering-based batch instance selection and the enlarging window strategy.

The ISLib library was integrated into the standard process monitoring tool from Petrobras, called SmartMonitor (ANZAI, T. K., FURTADO, *et al.*, 2023), and has been used to provide users with quick in-sights about their datasets, speeding the preprocessing steps and obtaining optimized regression models tailored to their specific applications. The next section showcases some of the results obtained by ISLib.

## 3.2 Results and discussion

### 3.2.1 METHODOLOGY

Results from ISLib are presented for three different cases: (I) the Tennessee Eastman Process (TEP) dataset, (II) an offshore flare gas flowmeter fault detection

framework, and (III) an offshore oil fiscal meter soft sensor. The datasets used in this study differ in several aspects, including sensor types, environmental conditions, and operational modes. For instance, the flare gas flowmeter dataset involves high flow velocities and varying gas compositions, while the oil flowmeter dataset is influenced by temperature and pressure variations. These differences highlight the adaptability of the proposed methodology to diverse industrial scenarios. Table 1 presents the cases evaluated and some of their characteristics.

The model performances were evaluated in two stages: During ISLib processing — prior to final model training — the error across observations was quantified using the MSE metric as defined in Algorithms 1 and 2. After the final models listed in Table 1 were trained, their performances were compared using the SPE metric. For each case, two models were produced: one trained with the instances selected by ISLib and another trained with the full dataset. An abnormal event was flagged when the SPE value exceeded  $\lambda$ .

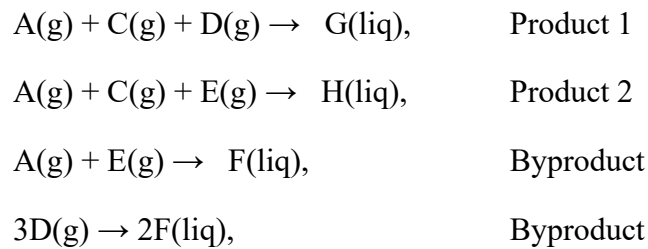
As summarized in Table 1, this study presents results from various algorithms in addition to the models used within ISLib for analysis (PCA or RT). Since the main goal of this part of the thesis is to address the importance and impact of pre-processing steps such as data reduction, detailed information regarding these models will not be extensively covered, given the relatively large number of surveys published in the open literature in this area. The interested reader might refer to MELO, CÂMARA, *et al.*, (2024b) and ALDRICH, AURET, (2013) for the employed PCA methodology; to HALLGRÍMSSON, NIEMANN, *et al.*, (2020b), ZHU, JIANG, *et al.*, (2022), and SPINA, DE O. CAMPOS, *et al.*, (2024) for the Autoencoder (AE) model detail; and to LEMOS, CAMPOS, *et al.*, (2021) for the Echo State Network (ESN) application.

Table 1: Summary of evaluated cases for Instance Selection.

<b>Dataset</b>	<b>Objective</b>	<b>Model (<math>h</math>)</b>	<b>Dataset initial size (<math>N</math>)</b>
TEP	Unsupervised model	PCA	500
Flare gas flowmeter	Unsupervised model	AE	30,775
Oil flowmeter	Supervised model	ESN	53,309

### 3.2.2 TENNESSEE EASTMAN PROCESS (TEP)

The first study used the well-known TEP dataset proposed by Eastman Chemical Company for evaluating process control and monitoring techniques (DOWNS, VOGEL, 1993, MELO, CÂMARA, *et al.*, 2022). The system comprises five major units: (I) a reactor, (II) a condenser, (III) a gas compressor, (IV) a separator vessel; and (V) a stripper. The fundamental purpose of this system, as indicated by the subsequent reactions, is to produce G and H from the gaseous reactants A, C, D, and E.



The remaining unit operations in the process aim to purify the products from the reaction, separating them from the generated byproducts and inert present in the system. The process has 11 manipulated variables and 41 measurements. Figure 13 shows the process flowchart with all variables and control loops.

The datasets used are available in RIETH, AMSEL, *et al.*, (2017) and can be divided into 2 groups: a normal operating period for model training and a second group containing 21 different failure modes. Table 2, Table 3 and Table 4 show, respectively, the manipulated variables, available measurements, and failure events with their descriptions.

As described in the methodology, the normal operation data from TEP was presented to ISLib to create an alternative reduced dataset. Since the data presented by RIETH, AMSEL, *et al.*, (2017) simulated just one operating point, with all the events listed in Table 4 being deviations from this steady-state condition, the first step from ISLib was bypassed, and the entire initial set of 500 observations was directly provided to the enlarging window strategy stage. The results from the reduced dataset model were then compared to those obtained from the full dataset model. Given that the ultimate objective was to create a PCA model for fault

detection, the ISLib analysis followed the unsupervised approach outlined in Algorithm 2.

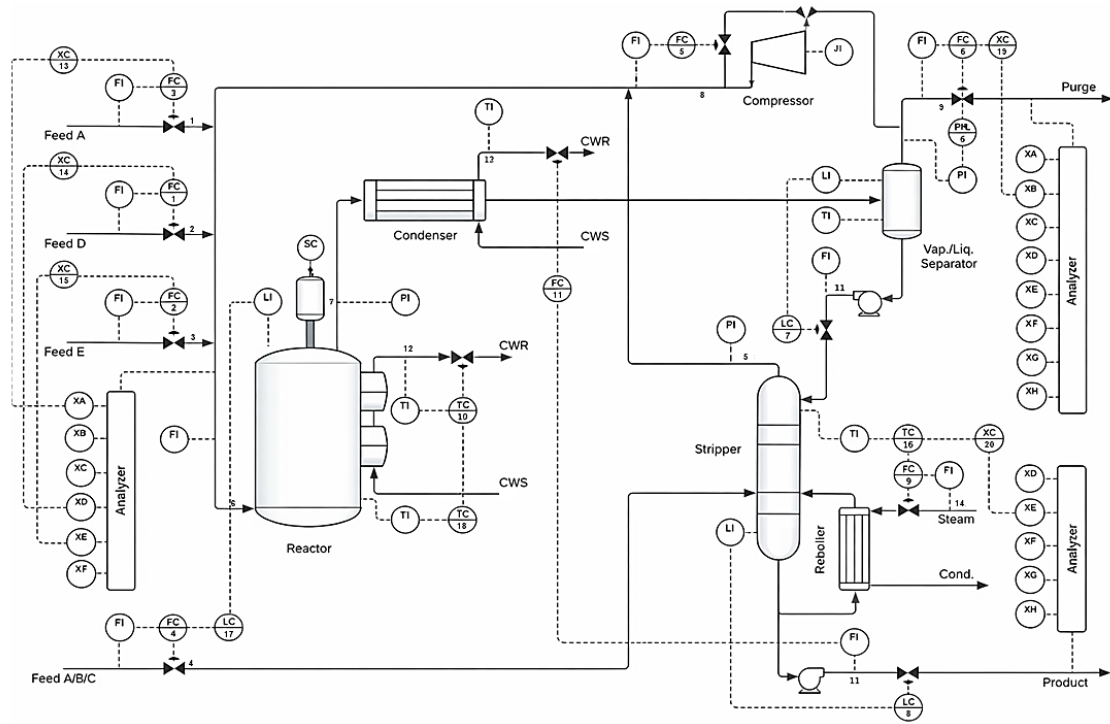


Figure 13: Process flowsheet of the Tennessee Eastman Process (TEP), depicting five main operational units: reactor, condenser, compressor, separator, and stripper (XAVIER, DE SEIXAS, 2018).

Table 2: Manipulated variables in the TEP process.

N°	Description	N°	Description
1	D Feed Flow (Stream 2)	7	Separator Pot Liquid Flow (stream 10)
2	E Feed Flow (stream 3)	8	Stripper Liquid Product Flow (stream 11)
3	A Feed Flow (stream 1)	9	Stripper Steam Valve
4	A and C Feed Flow (stream 4)	10	Reactor Cooling Water Flow
5	Compressor Recycle Valve	11	Condenser Cooling Water Flow
6	Purge Valve (stream 9)		

Table 3: Process measurements in the TEP process.

N°	Description	N°	Description
1	A Feed (stream 1)	22	Separator Cooling Water Outlet Temp
2	D Feed (stream 2)	23	Component A
3	E Feed (stream 3)	24	Component B
4	A and C Feed (stream 4)	25	Component C
5	Recycle Flow (stream 8)	26	Component D
6	Reactor Feed Rate (stream 6)	27	Component E

7	Reactor Pressure	28	Component F
8	Reactor Level	29	Component A
9	Reactor Temperature	30	Component B
10	Purge Rate (stream 9)	31	Component C
11	Product Sep Temp	32	Component D
12	Product Sep Level	33	Component E
13	Prod Sep Pressure	34	Component F
14	Prod Sep Underflow (stream 10)	35	Component G
15	Stripper Level	36	Component H
16	Stripper Pressure	37	Component D
17	Stripper Underflow (stream 11)	38	Component E
18	Stripper Temperature	39	Component F
19	Stripper Steam Flow	40	Component G
20	Compressor Work	41	Component H
21	Reactor Cooling Water Outlet Temp		

Table 4: Description of considered faults in the TEP.

<b>Fault number</b>	<b>Description</b>	<b>Type</b>
IDV(1)	A/C feed ratio, B composition constant (stream 4)	Step
IDV(2)	B composition, A/C ratio constant (stream 4)	Step
IDV(3)	D feed temperature (stream 2)	Step
IDV(4)	Reactor cooling water inlet temperature	Step
IDV(5)	Condenser cooling water inlet temperature	Step
IDV(6)	A feed loss (stream 1)	Step
IDV(7)	C header pressure loss-reduced availability (stream 4)	Step
IDV(8)	A,B,C feed composition (stream 4)	Random variation
IDV(9)	D feed temperature (stream 2)	Random variation
IDV(10)	C feed temperature (stream 4)	Random variation
IDV(11)	Reactor cooling water inlet temperature	Random variation
IDV(12)	Condenser cooling water inlet temperature	Random variation
IDV(13)	Reaction kinetics	Slow drift
IDV(14)	Reactor cooling water	Valve Sticking
IDV(15)	Condenser cooling water	Valve Sticking
IDV(16)	Variation coefficient of the steam supply of the heat exchange of the stripper	Random variation
IDV(17)	Variation coefficient of heat transfer in reactor	Random variation
IDV(18)	Variation coefficient of heat transfer in condenser	Random variation
IDV(19)	Unknown	Unknown
IDV(20)	Unknown	Unknown
IDV(21)	A feed (stream 1) temperature	Random variation
IDV(22)	E feed (stream 3) temperature	Random variation
IDV(23)	A feed flow (stream 1)	Random variation
IDV(24)	D feed flow (stream 2)	Random variation
IDV(25)	E feed flow (stream 3)	Random variation
IDV(26)	A and C feed flow (stream 4)	Random variation
IDV(27)	Reactor cooling water flow	Random variation
IDV(28)	Condenser cooling water flow	Random variation

Figure 14 illustrates the MSE values obtained by each PCA model as the training dataset size increases incrementally. Since the initial dataset corresponds to a clearly defined operating region, the MSE profile exhibits a consistent decreasing trend as the model retains more information from the provided data. The dashed line in Figure 14 represents the minimum MSE value obtained, accounting for a 10% tolerance. The red dot indicates the point at which the MSE reaches this minimum. In the context of the present study, this value — equivalent to 340 — represents a dataset reduction of 32%, when compared to the original data size of 500 observations.

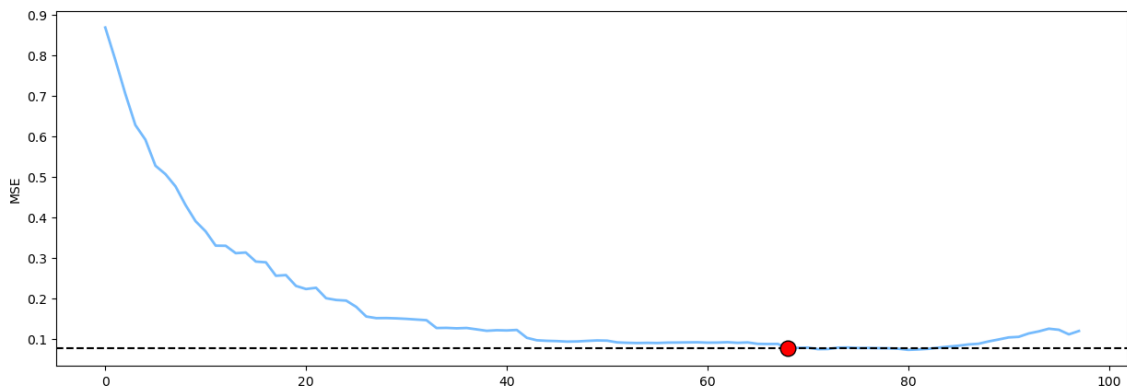


Figure 14: MSE values for the enlarging-window models generated from the TEP dataset using a resolution of 100. The x-axis represents the progressive enlargement of the training window in steps of  $N/100$ .

The obtained datasets generated two subsequent PCA models: one referred to as “reduced dataset,” which used only the first 340 measurements from the original training dataset, and the other as “full dataset model,” incorporating the entire data. In both models, the number of principal components was selected to explain 90% of the variance in the input data.

For this analysis, only a subset of fault scenarios was evaluated, focusing on those previously reported to be detectable by PCA-based monitoring. This criterion ensures consistency in the analysis and will also be applied later in Chapter 4 when evaluating the drift-aware diagnostic framework. Figure 15 shows the SPE profiles obtained for the two models. The dashed lines, calculated as the

99th percentile of the respective training SPE, serve as the basis for anomaly detection. The two models exhibited similar behavior, identifying the anomalies at nearly the same instants. In some cases, such as in IDV(2) and IDV(6), the reduced-dataset model produced a sharper separation between normal and faulty behavior, reflected in a larger contrast between SPE values before and after the fault. Nonetheless, in practical terms, for this case the response to the anomaly is quite evident in both models.

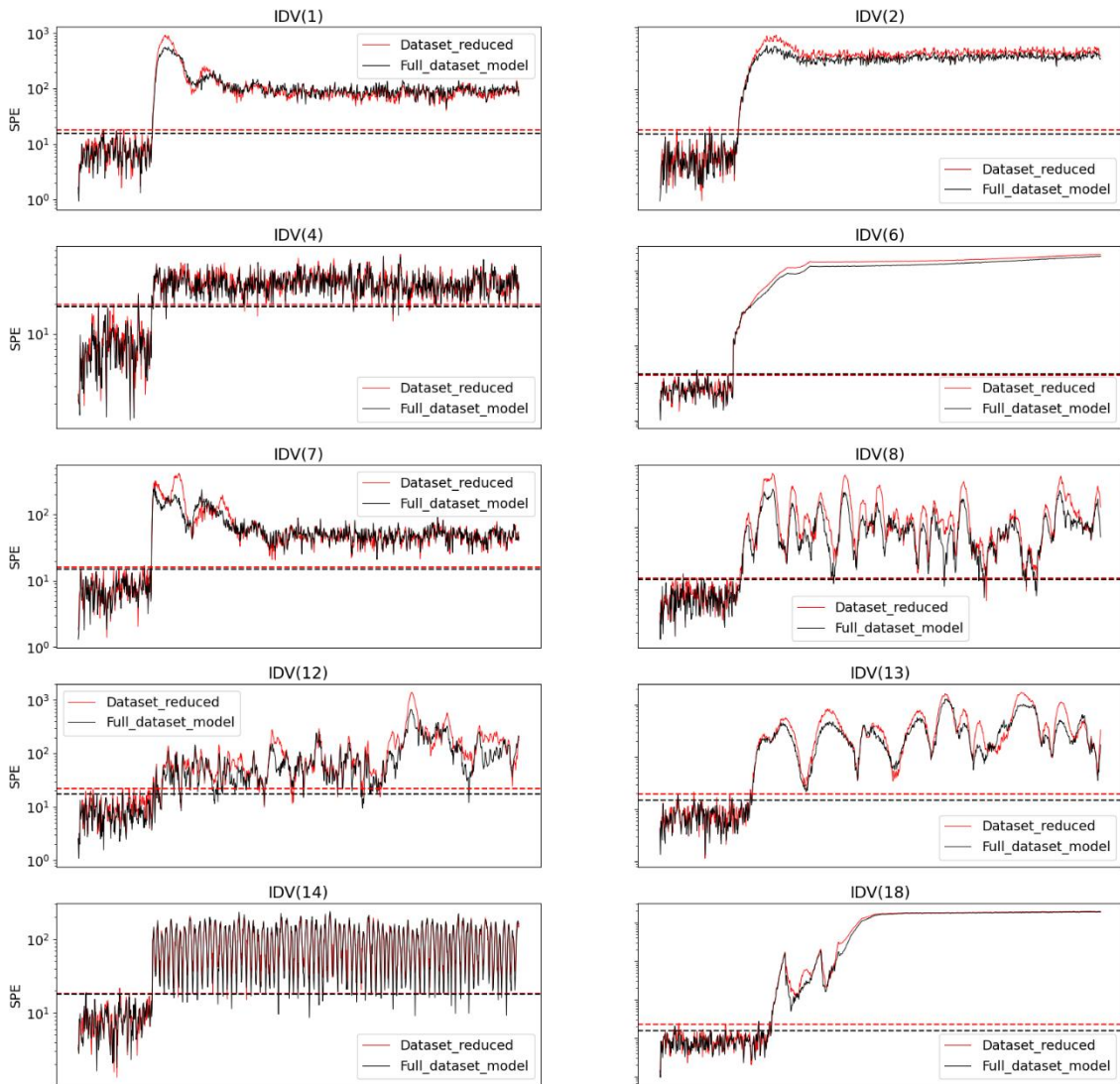


Figure 15: SPE profiles for the reduced and full model against different types of faults.

Table 5 presents similar results, but in a quantitative format. The fault indices evaluated from Table 4 are listed in the first column. The second and third columns compare the MSE values obtained with the full dataset PCA against the reduced dataset model. Both performances were computed using the initial portion of the test dataset, where the anomaly had not yet occurred, as outlined by the low SPE values in Figure 15. Columns four and five correspond to the period during the presence of the fault. In this scenario, by knowing exactly where the anomaly begins, it is possible to calculate the number of false negative (FN) points obtained for each model; or, in other words, how many observations rely below the respective  $\lambda$  after the fault initiation.

Averaging all cases, it is possible to notice the similar performance of both models in the test dataset. This is a notably interesting result because it demonstrates that even for a simulated scenario, confined to a single operational region, with known and constant uncertainties throughout the samples, it was possible to significantly reduce the dataset dimension without compromising the quality of the final model. As illustrated in the upcoming examples, in real-world scenarios where operational regions are not well-defined, making it challenging to identify a representative training dataset, the importance of employing instance selection techniques becomes even more pronounced.

Table 5: Results for anomaly detection using full and reduced datasets for training.

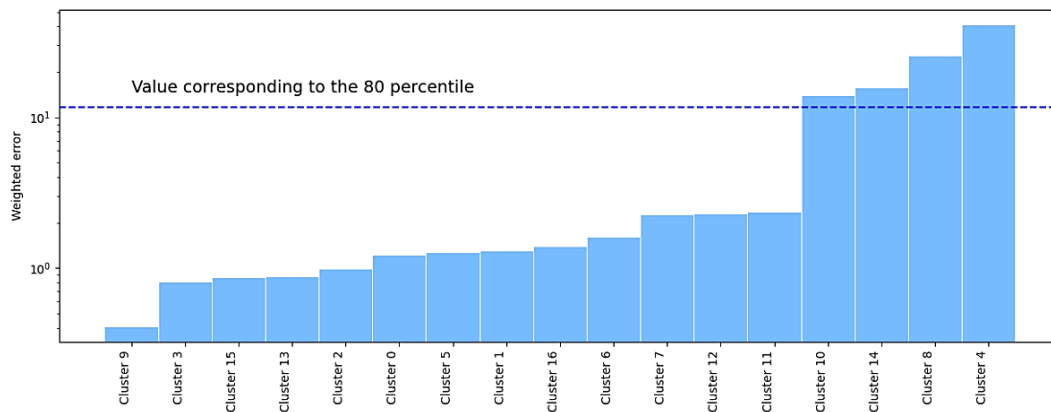
<b>Fault Index</b>	<b>MSE before anomaly</b>		<b>FN After anomaly</b>	
	<b>Full Dataset</b>	<b>Reduced Dataset</b>	<b>Full Dataset</b>	<b>Reduced Dataset</b>
1	6.8	6.7	2	2
2	7.9	7.8	11	11
4	9.1	8.7	9	21
6	8.7	8.6	0	0
7	7.6	8.5	0	0
8	8.6	8.1	31	18
12	8.4	9.2	31	44
13	7.0	7.7	39	39
14	8.1	8.7	36	7
18	8.7	8.2	78	80
Average	8.1	8.2	24	22

### 3.2.3 FLARE GAS FLOWMETER

In offshore oil facilities, flaring gas is the action of burning waste crude natural gas that is not possible to process or sell (ZARDOYA, LUCENA, *et al.*, 2022). Because of its impact on carbon emissions, in Brazil, ANP limits the authorized burning and losses, establishing parameters for its control (RODRIGUES, 2022).

Measuring the amount of gas sent to flare, however, does not constitute a simple task since it involves dealing with large pipe diameters, high flow velocities over wide measuring ranges, gas composition changes, condensate, etc. (EMAM, 2015). Despite that, it is common to install gas metering only on the main flare header (IOGP, 2024), making its reliability crucial for ensuring a safe operation.

In this application, a monitoring system using AE was specifically designed to detect anomalies in the flare gas meter of a Petrobras oil production platform. The dataset was obtained from historical operational data of a Petrobras oil production platform and comprised 30,775 observations and 5 features, including one volume flow measurement, one pressure transmitter, one level signal and two temperatures. Figure 16 shows the results obtained for the batch instance selection stage from ISLib. The colored trends represent the clusters identified by the K-Means algorithm, and the bar chart shows the MSE associated with each cluster when used to predict all other operational regions.



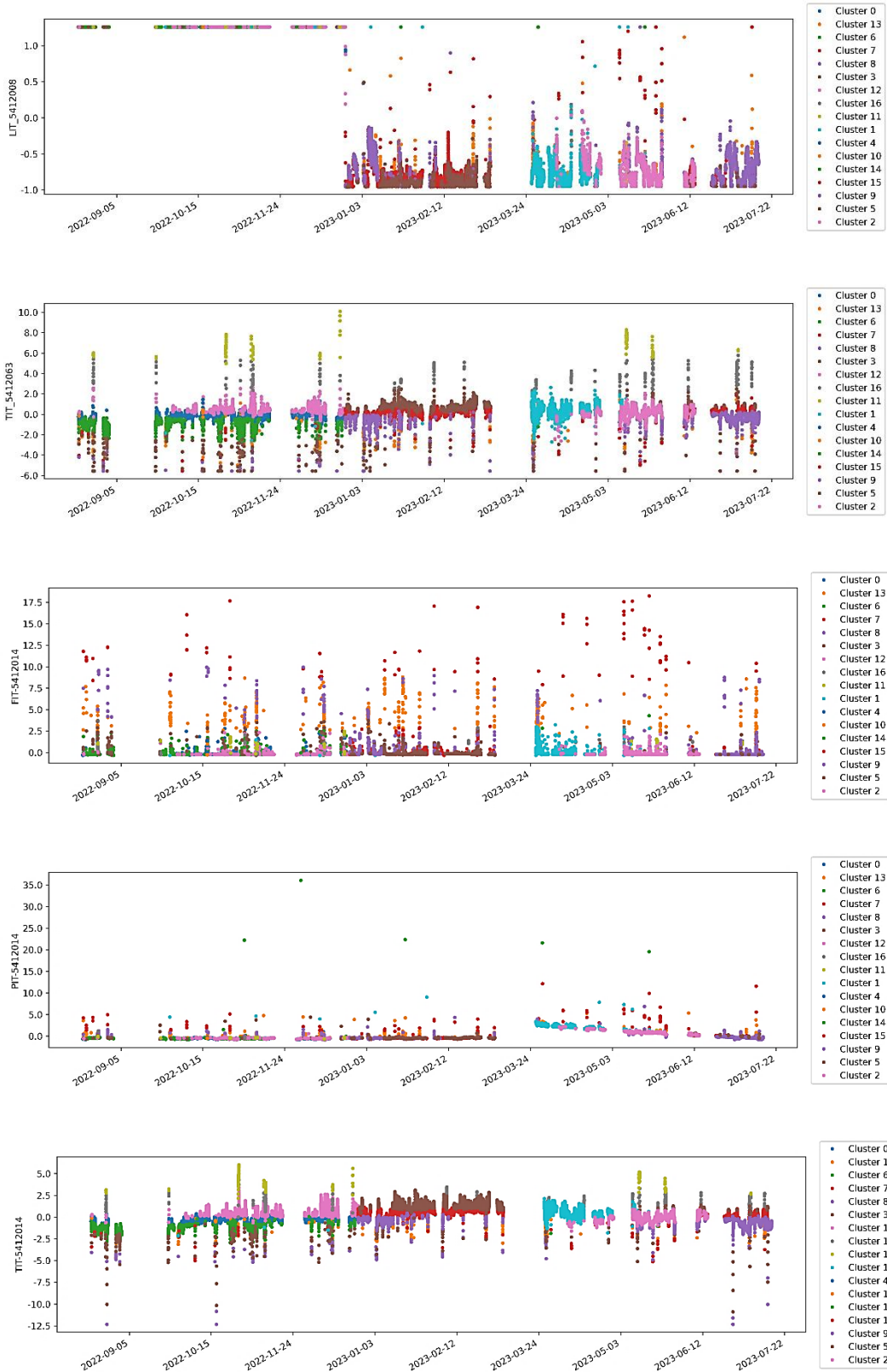


Figure 16: Cluster analysis applied to the gas flowmeter dataset encompassing visual representation of the operational regions and ranked clusters.

Based on the previous results, the monitoring team responsible for the system decided to exclude the regions associated with the highest MSE values, notably clusters 4, 8, 10, and 14, from subsequent analyses. This decision was made on the assumption that these regions were linked to production interruptions, outliers, or other non-representative operating modes.

Figure 17 shows the enlarging window approach from ISLib using the regions defined by the selected clusters.

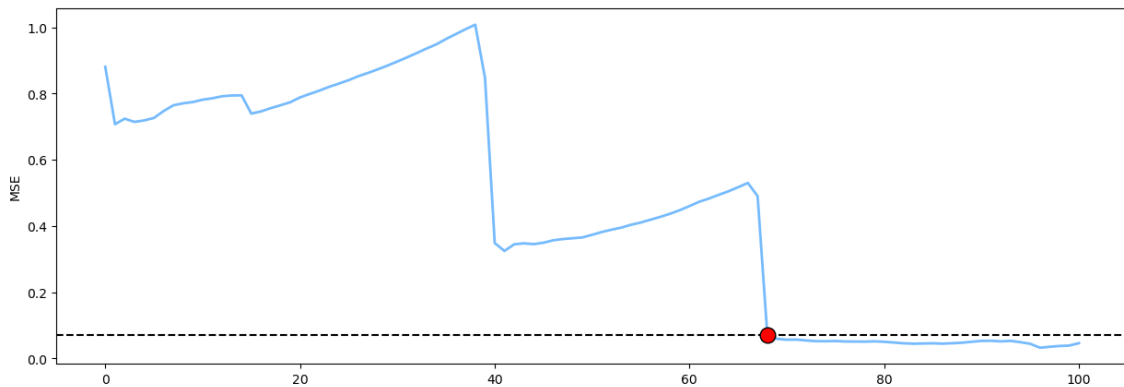


Figure 17: MSE values for the enlarging window models generated from the gas flowmeter dataset using a resolution of 100.

The trend observed in Figure 17 illustrates one of the most important outcomes of this analysis. As can be seen, the error profile has two significant discontinuities as the training window increases. Those steps mark the points in the dataset where major changes happen. In other words, if the dataset contains different operating regions — as suggested by the clustering analysis — as the training window increases, the testing window consequently decreases, becoming more concentrated in regions that might not have been presented to the model yet. Because of this poor generalization capacity from the model, the MSE tends to increase at first. As soon as a new condition, never experienced by the model, is incorporated into the training data, the resulting model can identify this new condition, yielding the steep drop observed in the MSE trend. The cycle restarts if new conditions are present in the remaining part of the dataset. If, however, the last unexplored subset is presented to the model, the MSE profile may show no more considerable

variations. At this stage, as already mentioned, including new instances for training would not increase the predictive power of the regression model, meaning that the minimum window size had been achieved.

Figure 18 shows the density distributions of the five features from the two datasets: the full dataset and the reduced dataset obtained through ISLib. Despite the reduction in data volume, the distributions of the reduced dataset closely align with those of the original dataset across all features. The similarity in distribution ensures that the reduced dataset remains representative of the original dataset, preserving its statistical properties and integrity for subsequent analysis or modeling.

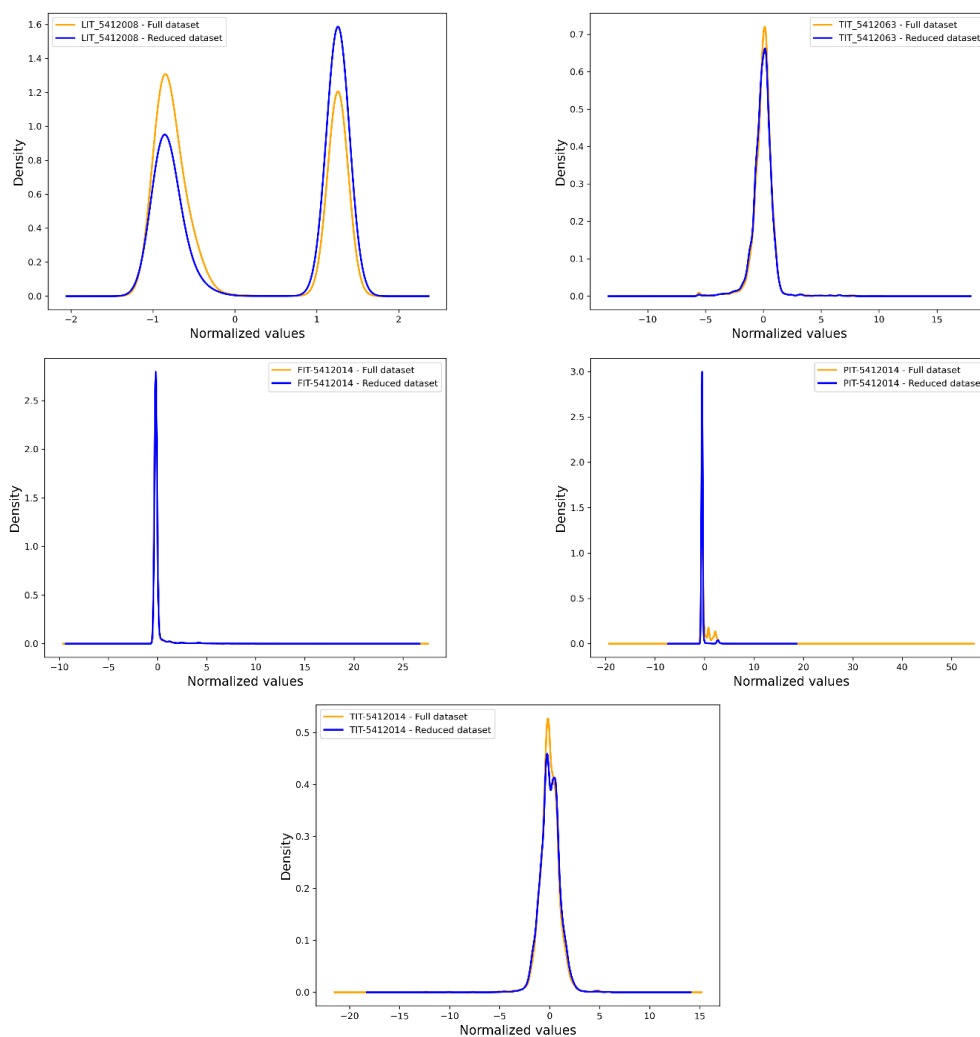


Figure 18: Density distributions for the full and reduced datasets, demonstrating that ISLib reduced data volume while preserving the original distribution characteristics.

The overall analysis by ISLib, in this case, took 7 seconds to converge in an Intel(R) Core(TM) i7-12800H 2.40 GHz with 32 GB RAM, speeding a process that would otherwise require long mining through historical data trends and information from different sources such as shutdowns occurrences and maintenance registers.

After the two sequential analyses from ISLib, the resulting dataset comprised 20,809 observations, a reduction of 32% from the initial size. Both datasets were then used to train two AE networks. In order to compare the best possible models generated from each dataset, a hyperparameter tuning procedure was conducted separately using the Python framework Optuna (AKIBA, SANO, *et al.*, 2019).

The Optuna's hyperparameter selection process involves employing the Tree Parzen Estimator (TPE), a Bayesian-inspired sequential optimization method (BERGSTRA, James, BARDENET, *et al.*, 2011). TPE reduces computational effort by minimizing the number of objective function evaluations, enabling real-time implementations. It formulates hyperparameter selection as an optimization problem, aiming to minimize the objective function evaluated on a validation set. This task becomes computationally challenging for models with numerous hyperparameters, such as neural networks. The TPE method addresses this issue by utilizing a surrogate objective function to approximate the true objective function. By iteratively maximizing the surrogate function, TPE identifies promising hyperparameter regions. The values with the highest surrogate function, called Expected Improvement, indicate regions for minimizing the original objective function. The probabilities of the surrogate model are updated accordingly, and the process continues until convergence or satisfactory hyperparameter values are found. The TPE method, as a Sequential Model-Based Optimization (SMBO) approach, reduces computational costs and offers advantages such as parallelizability and implementation in accessible numerical libraries. (AKIBA, SANO, *et al.*, 2019, BERGSTRA, J., YAMINS, *et al.*, 2012). For more information on TPE and the application of the Optuna framework to industrial process models, one should refer to the works of BERGSTRA, James, BARDENET, *et al.*, (2011) and LEMOS, CAMPOS, *et al.*, (2021), respectively. For the present study, the

hyperparameter tuning process using Optuna focused on optimizing key parameters for each model. For AE, parameters such as the number of hidden layers, activation functions, and learning rates were optimized. For ESN, presented in the next subsection, the reservoir size, spectral radius, and input scaling were fine-tuned to enhance predictive performance.

Hyperparameter tuning took 6,968 seconds for the full dataset and 4,838 seconds for the reduced dataset. The performance of the two AE models during an online monitoring period with abnormal events is illustrated in Figure 19. All SPE values are normalized by their respective threshold ( $\lambda$ ). As observed, during normal operation from the flare system, both models exhibit similar responses, with the AE generated using the reduced dataset displaying slightly lower reconstruction errors. During the abnormal events, both AE consistently show an increase in SPE values. However, since the model trained with selected instances appeared to be more sensitive to the presence of anomalies, as indicated by the higher peaks above  $\lambda$ , some of the faulty operations, such as the one observed on November 15th, could only be detected using the dataset provided by ISLib.

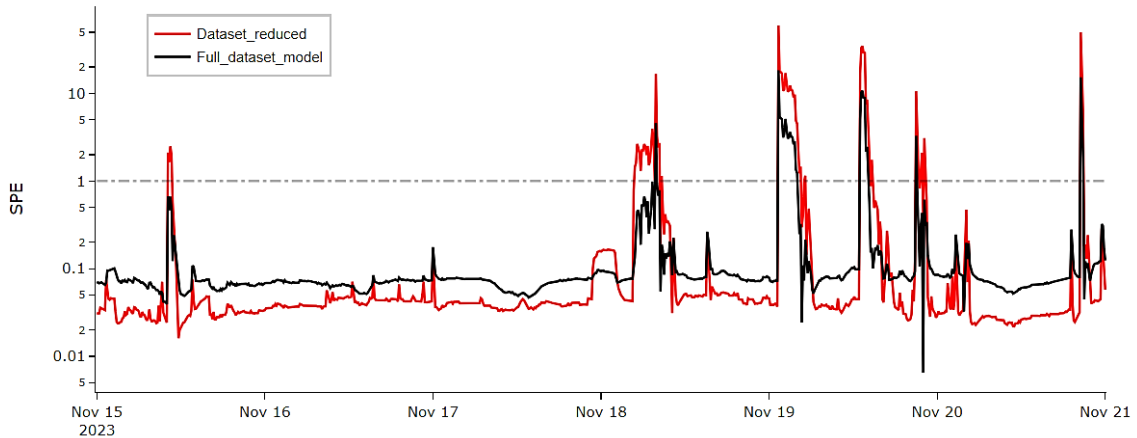


Figure 19: SPE profiles obtained from the AE model trained using the whole dataset (in black) and the AE generated from the data provided by ISLib (in red). The grey dashed line represents the  $\lambda$  value.

Figure 20 illustrates the same results as shown in Figure 19 while also presenting the corresponding trends from three of the features used as inputs for the

models as obtained from historical operational data. It can be observed that a typical failure mode consists of a depressurizing event, indicated by the sudden increase in the pressure curve (depicted in blue), that is somehow not followed by a corresponding change in volume flow (the black line in the features trends). In such abnormal cases, the signal from the flowmeter either remains at zero or stays constant at some level below the expected value, given the behavior of the other features. This difference between the expected values from the AE model and the actual measured flow results in the high SPE shown in Figure 20. In this case, the reduced AE model successfully detected all the faulty operations while maintaining a low number of false positive alarms.

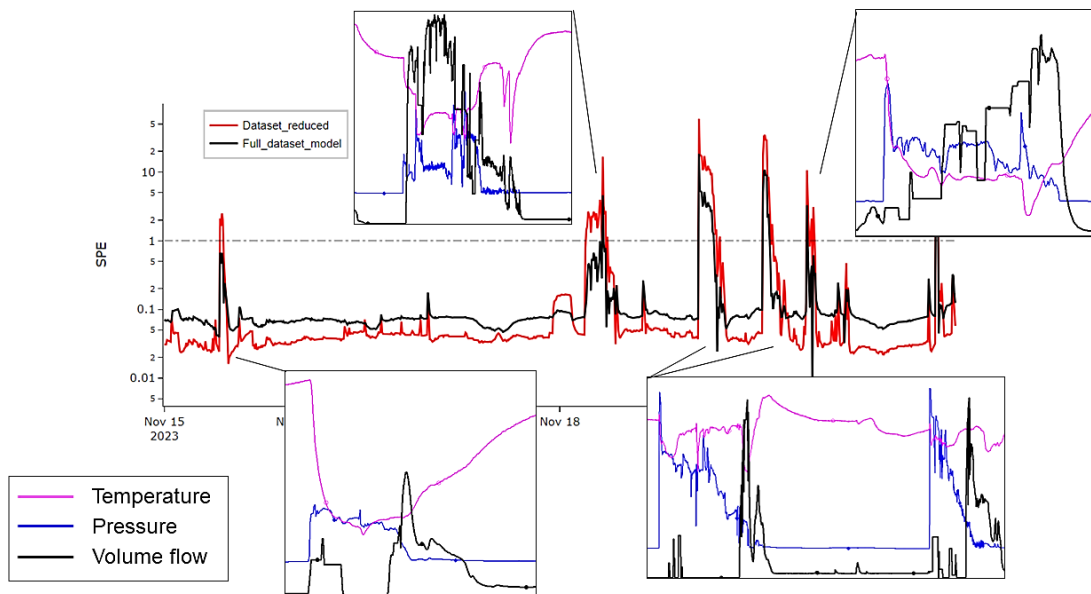


Figure 20: SPE profiles of both models and the measured values from the features during anomalies observed in the volume flow signal.

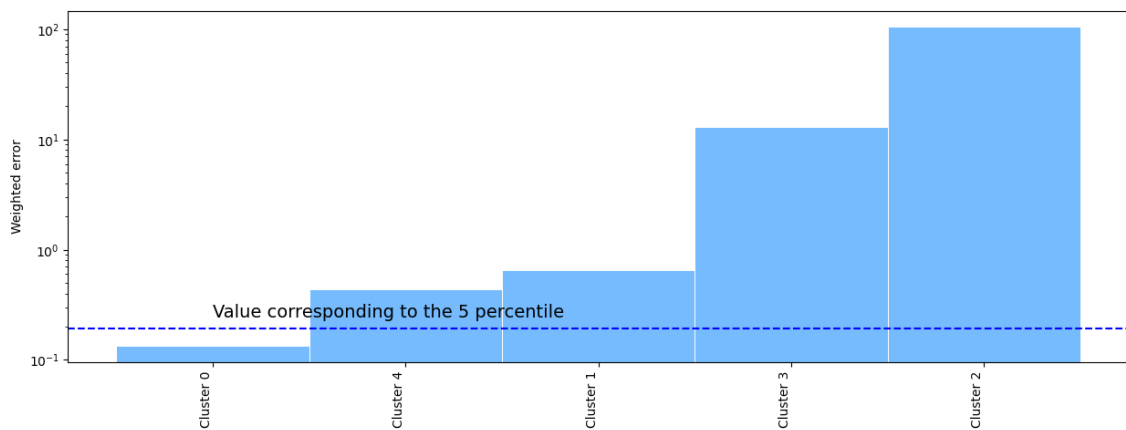
### 3.2.4 OIL FLOWMETER

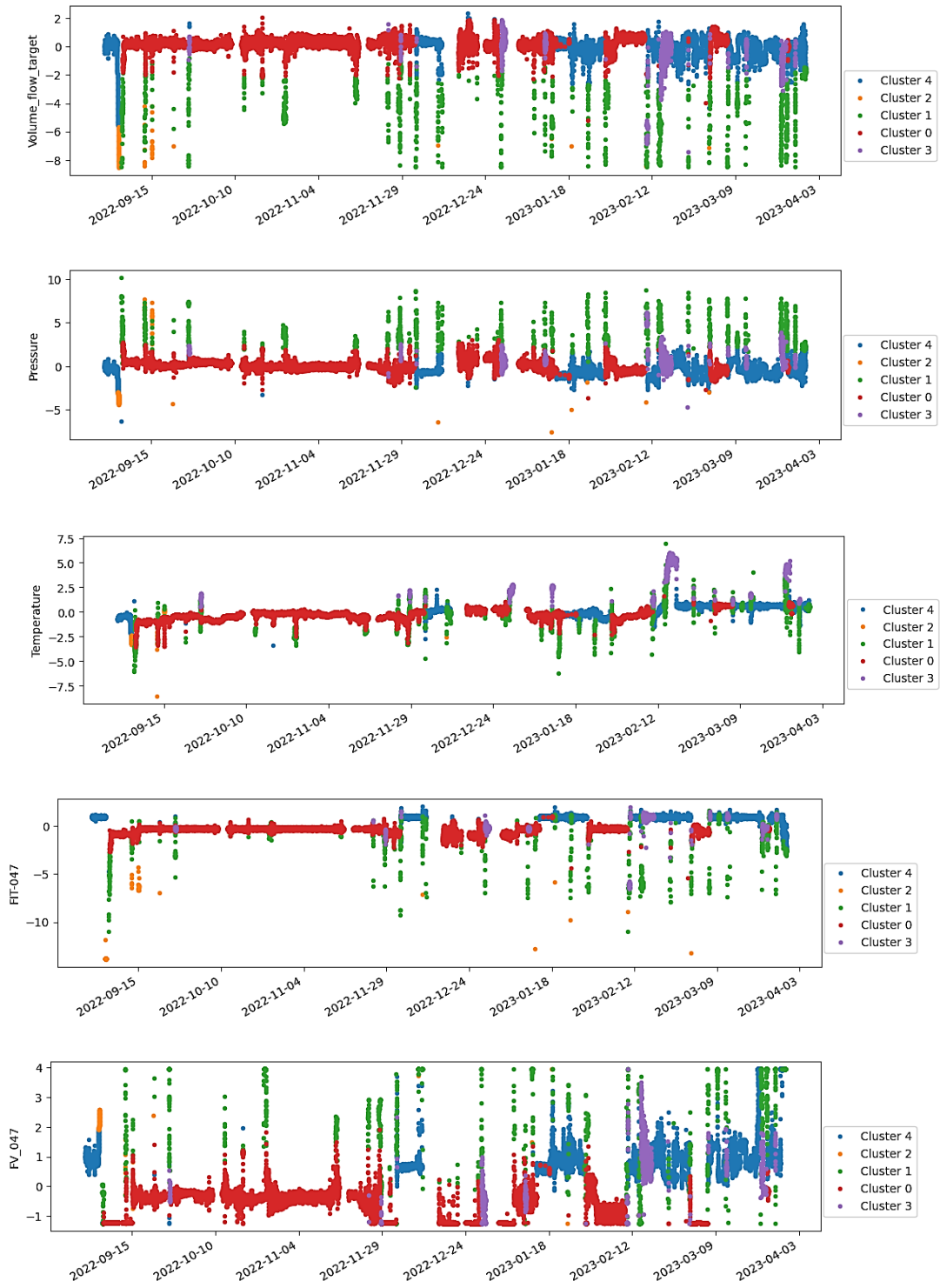
Fiscal meters are critical instruments on oil production platforms because they determine royalties and custody-transfer values (CLAVIJO, N., MELO, *et al.*, 2019). Measurement errors in these equipment can cause potential penalties and significant financial losses (BEKRAOUI, HADJADJ, *et al.*, 2019). For this reason,

regulatory agencies establish guidelines to ensure accurate and reliable measurements (BARATEIRO, FARIA, *et al.*, 2022). In Brazil, for instance, ANP has strict procedures and protocols that must be followed in case of abnormalities such as incorrect measurement and configuration errors (CLAVIJO, N., MELO, *et al.*, 2019).

In order to minimize downtime and ensure the continuous operation of an oil fiscal meter on a Petrobras platform, a monitoring system was developed using data from other process-plant instruments — such as temperature and pressure measurements — to predict the oil volume flow. The initial dataset consisted of nine features and 53,309 observations from months of operation obtained from historical operational data.

Figure 21 presents the results obtained from the batch instance selection step from ISLib when presented to all 53,309 observations. It is important to notice that, in this case, since the objective of the application was to produce a supervised model for a soft sensor, the supervised option was also applied in ISLib, meaning that, differently from the previous two cases, the results shown here were generated by an RT model instead of the unsupervised PCA approach described in Algorithm 1 and Algorithm 2.





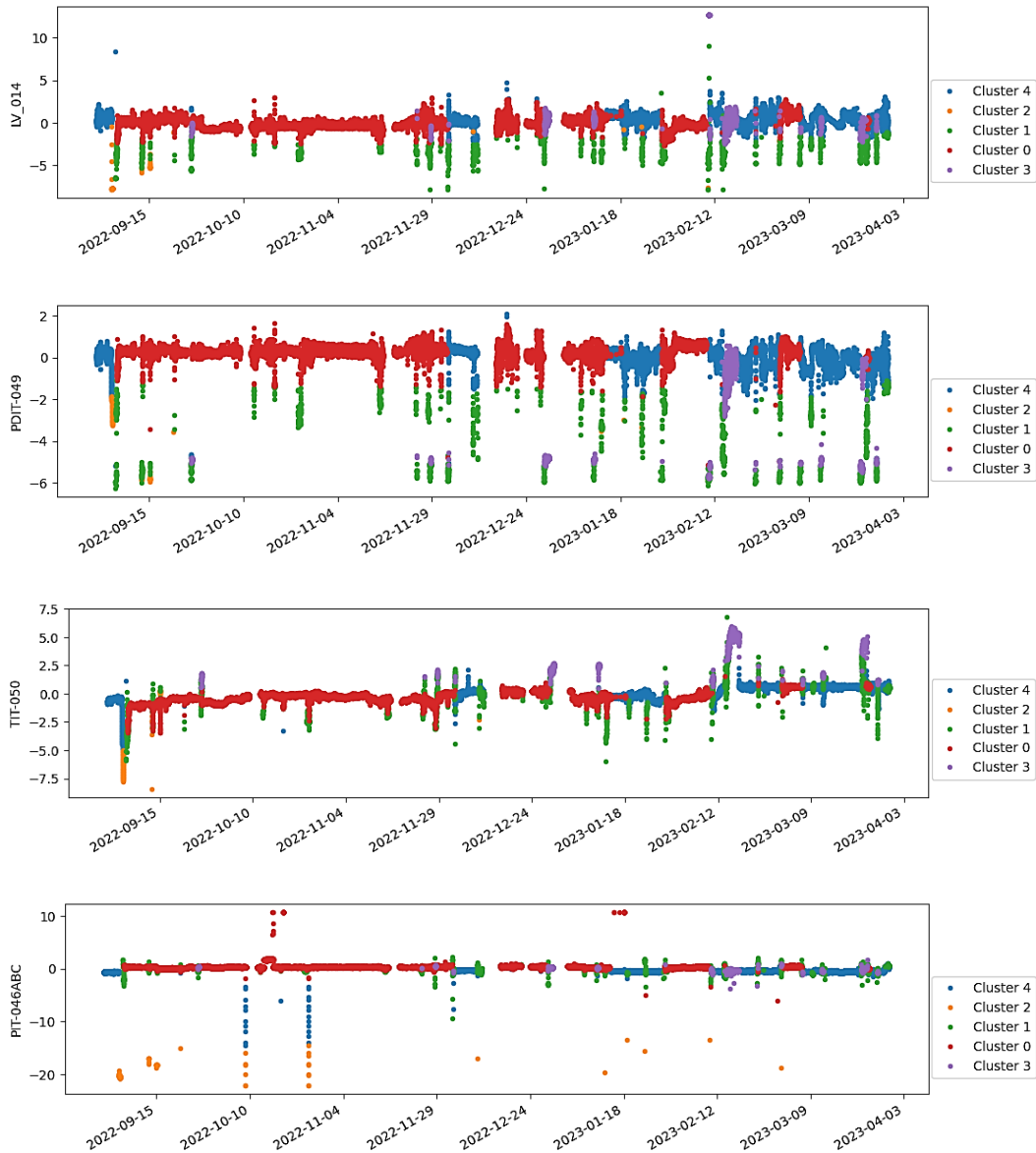


Figure 21: Cluster analysis applied to the oil flowmeter dataset encompassing visual representation of the operational regions and ranked clusters.

Following a similar approach to the previous subsection, the monitoring team decided to keep only the highest-ranked cluster for the remaining steps by examining the operating modes. This was because all the other regions seemed to show some undesirable characteristics, such as measurement errors, plant shutdowns, etc., being, thus, classified as a different cluster by ISLib.

Figure 22 presents the enlarging windows analysis using the previously selected region (Cluster 0) as input. The results along the 100 intervals in the x-axis

showed three pronounced drops in the MSE values in the first half of the dataset. This indicates that the enlarging window procedure could outline subregions within the cluster analysis.

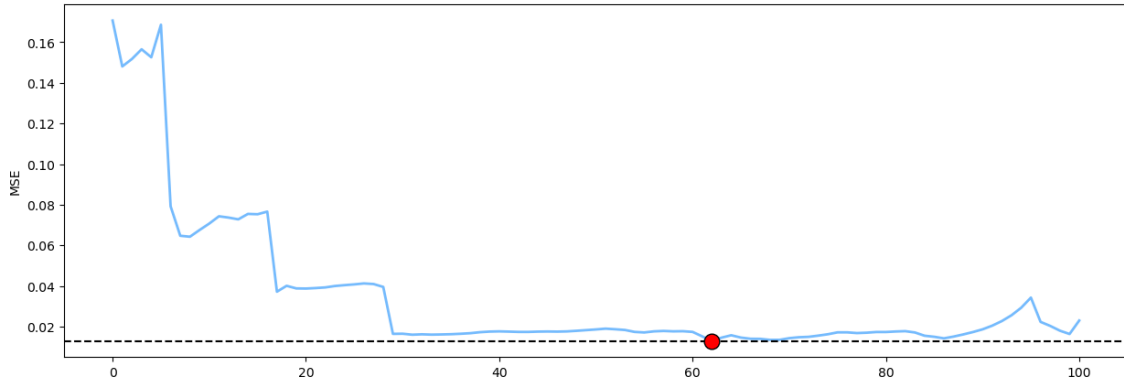


Figure 22: MSE values for the enlarging-window models generated from the TEP dataset using a resolution of 100. The x-axis represents the progressive enlargement of the training window in steps of  $N/100$ .

Preserving the data prior to the red circle in Figure 22, i.e., before the point at which the MSE reaches its minimum value, resulted in a dataset with 20,275 observations, a 62% reduction compared to the original dataset size of 53,309.

Using the previous methodology, the supervised model training was preceded by hyperparameter tuning in Optuna to obtain the best configuration for each ESN model. The total time required for optimization was 4,229 seconds for the full dataset and 739 seconds for the reduced dataset.

Both trained recurrent neural network models were deployed using the SmartMonitor platform and consistently provided accurate predictions for the oil volume flow. Figure 23 shows the predicted values from both models plotted against the measured flow during a selected test period, in which the measured values were known to be reliable. The scatterplot on the right reveals a higher concentration of points along the identity line during normal operation, while a more dispersed pattern appears at extreme values. Notably, this behavior persists even for the ESN model trained on all operational regions.

Figure 24 shows the residue distribution of the models during the same test period. As can be seen, both distributions show small offsets from zero. Notably, the ESN trained with only 38% of the original dataset presented better results, as evidenced by the higher concentration of residues closer to zero compared to the ESN trained with the whole dataset. The reduced dataset performs well due to the elimination of noisy and redundant data points, which enhances the model's ability to generalize.

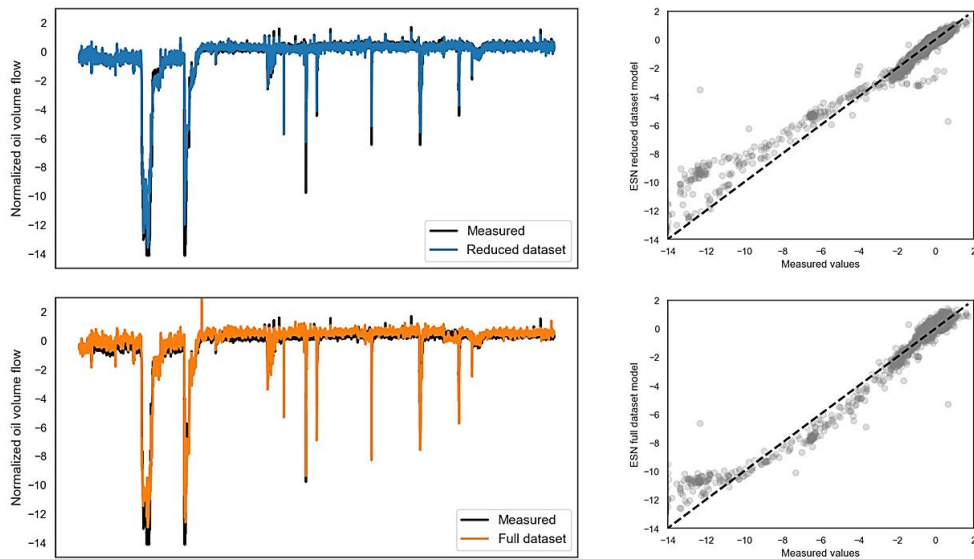


Figure 23: Predicted versus measured normalized values of both models during a test period, with the x-axis representing the sequential test samples.

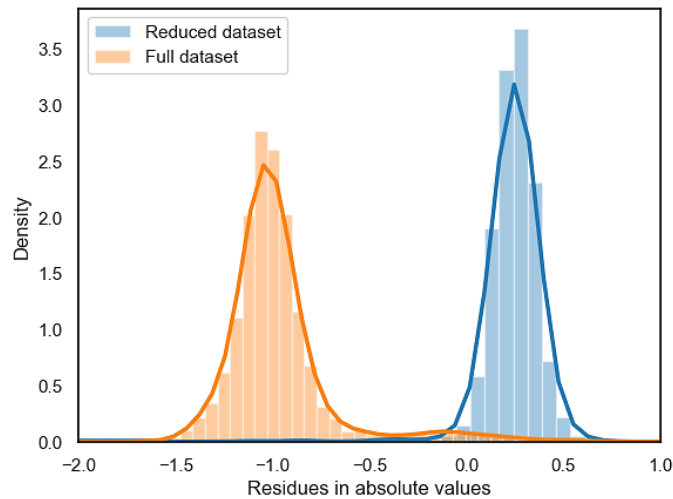


Figure 24: Residues distribution of both ESN models.

The results obtained across the evaluated case studies indicate that the benefits of the proposed instance selection strategy are not limited to reducing the size of the training dataset. In all considered scenarios, models trained on subsets identified by ISLib were able to preserve, and in some cases improve, monitoring performance when compared to models trained on the full dataset. This behavior suggests that the representativeness of the selected data plays a more critical role than the volume of available observations when learning normal process behavior.

From an operational perspective, the batch instance selection stage can be interpreted as an offline mechanism for identifying dominant operating regions in historical data. Clusters that generalize well across different regions tend to encode stable relationships among process variables, whereas poorly ranked clusters are often associated with transient conditions, measurement noise, or localized behaviors that are not representative of the overall process dynamics.

The enlarging window analysis further reinforces this interpretation. The observed saturation of predictive performance, as well as abrupt changes in error profiles as the training window expands (and, consequently, the testing window shrinks), indicate that adding more data does not necessarily translate into improved generalization. Instead, these transitions suggest the presence of distinct operating regimes within the historical dataset. From this perspective, the enlarging window strategy can be interpreted as an offline indicator of regime changes, bridging instance selection and operating mode characterization.

### **3.3 Concluding remarks**

During this part of the thesis, the focus was placed on the development and evaluation of ISLib as a systematic instance-selection methodology for data-driven regression problems in industrial monitoring. This study emphasized the critical yet often overlooked step of data reduction in process monitoring frameworks, demonstrating how instance selection can significantly impact model performance and operational efficiency in real-world scenarios. By employing a two-stage

approach, ISLib demonstrated its ability to improve model performance while reducing dataset size.

The sequential methodology for instance selection was evaluated in three distinct cases, covering a range of applications with specific goals, allowing the proposed procedure to be tested in various scenarios, including (i) independent evaluation of the two methodologies that constitute the library, (ii) application in a real fault detection case, (iii) application in a real soft-sensor generation case, and (iv) utilization of the results for training different data-driven models encompassing both supervised and unsupervised approaches. Furthermore, the impact of incrementally larger datasets on the execution time of ISLib was also quantified. It was shown that smaller datasets not only accelerated the training process, but also generated models that performed equally to or better than their counterparts trained with the full-size dataset.

Table 6 complements the information provided in Table 1 by adding the results obtained from ISLib for each evaluated case. Notably, a significant reduction in model training time was observed for both industrial cases. Moreover, it is worth noting that ISLib was able to complete its analysis in less than 20 seconds, even for the largest evaluated dataset. This performance can be attributed to the utilization of robust and fast-converging algorithms such as PCA and RT, as well as the incorporation of the K-Means technique for identifying operational regions. Based on the results obtained, it can be concluded that any potential gains in accuracy achieved using more powerful techniques were outweighed by the speed and responsiveness of the proposed methodology.

Table 6: Summary of the results obtained for the evaluated cases.

Dataset	Objective	Model ( $h$ )	Dataset initial size ( $M$ )	Dataset final size ( $S$ )	Training time reduction [%]	IS analysis time [s]
TEP	Unsupervised model	PCA	500	340	0	3
Gas flowmeter	Unsupervised model	AE	30,775	20,809	30	7
Oil flowmeter	Supervised model	ESN	53,309	20,275	82	17

The obtained results highlight the importance of a proper instance selection — even in regression problems — and encourage the adoption of simple and efficient solutions that deliver valuable insights to end-users in complex industrial environments.

# Chapter 4

## After Detection: Drift-Aware Fault Diagnosis

Conventional anomaly diagnosis methods generally regard process faults and model drift as distinct, independent issues. Anomalous behavior is frequently attributed to process problems, while model drift is often considered a secondary concern. This traditional perspective neglects the fact that, when deviations are detected between process model and process data (fault detection stage), the first diagnosis that must be provided regards the source of the observed deviation: a process fault or a model malfunction. This part of the thesis presents a novel perspective that characterizes model drift as a fault diagnosis problem, proposing that effective anomaly diagnosis should distinguish process faults from model inadequacies originating from operational changes.

To address this challenge, the Nearest Normal Value (NNV) contribution analysis technique was developed to quantify individual variable contributions through counterfactual analysis. Unlike conventional diagnostic methods that rely on static references, the NNV technique provides dynamic contribution profiles that characterize the operational state through adaptive reference tracking.

Figure 25 illustrates the proposed framework, where both fault and drift detection systems feed into a fault diagnosis module that aims to discriminate real process faults from model inadequacy, in contrast to the conventional framework presented in Figure 9. Within this structure, NNV serves as the diagnostic layer responsible for interpreting alarms and identifying whether they originate from true process faults or from model inadequacy caused by operational changes.

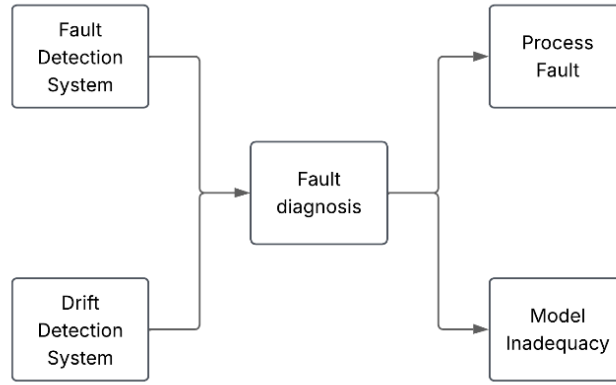


Figure 25: Proposed integrated framework using fault diagnosis as an intermediate layer.

Parts of the methodology and results presented in this chapter were previously published in the article “Distinguishing Process Faults from Model Drift Through Variable Contribution Analysis: A Novel Perspective on Anomaly Diagnosis”, by Thiago K. Anzai and José Carlos Pinto, published in *Processes*, 2026, 14, 859 (ANZAI, Thiago K., CARLOS COSTA DA SILVA PINTO, 2026).

#### 4.1 Nearest Normal Value (NNV) Contribution Method

As mentioned in section 2.2, the retraining dilemma requires systematic criteria to differentiate process faults from operational changes. This work proposes that variable contribution signatures might provide such criteria, by stating that faults produce concentrated and persistent contribution patterns that are localized in specific variables, while operational transitions produce distributed transient patterns that reflect coordinated shifts.

Contribution vectors can be interpreted as a nonnegative allocation of residual inconsistency across variables. Localized faults are expected to concentrate inconsistency on directly impacted variables, whereas drift-induced model mismatch tends to spread across variables and regimes, producing more diffuse profiles.

The underlying diagnostic idea is formalized by the following hypotheses:

- **H1 (Fault-concentrated hypothesis).** Let  $\mathbf{C} \in \mathbb{R}^M$  denote the vector of nonnegative, normalized variable contributions associated with SPE values above  $\lambda$ , with  $C_i \geq 0$  and  $\sum_{i=1}^M C_i = 1$ . When a process or sensor fault affects a subset of variables more than others, the contribution distribution becomes more heterogeneous. In this case, the contribution vector is expected to be more concentrated, such that a small subset of indices accounts for a larger fraction of the total SPE.
- **H2 (Regime-distributed hypothesis).** When an alarm is triggered by operating mode changes, the normalized contribution vector tends to be more evenly distributed across all variables. In this case, the increase in SPE reflects coordinated, system-wide adjustments, and the contribution profile is expected to be less concentrated than in the case presented by H1.

In traditional contribution analysis for regression problems, the results of each contribution, calculated according to Eq. 4, are sorted in descending order, being the first variable pointed out as the largest contributor to the abnormal event. Under normal operating conditions, one can expect all the normalized contributions to be close to each other, presenting an average value of  $1/M$ . In other words, each of the  $M$  features should contribute equally to the composition of the SPE index. When anomalies are present in the data, however, the tendency toward a uniform distribution does not apply. In such cases, the normalized contributions deviate from the expected baseline, typically concentrating on the variables whose behavior most strongly deviates from the learned normal relationship.

One way to account for the effect of an individual feature on the model prediction is to replace this feature with a different value and quantify the effect of this change on the predictions, or ultimately on the SPE. In explainable machine learning literature, this approach is commonly referred to as counterfactual analysis, and has been widely adopted for interpreting black-box predictors, providing insights from complex machine learning models (WACHTER, MITTELSTADT, *et al.*, 2018; VERMA, BOONSANONG, *et al.*, 2022; GUIDOTTI, 2024).

An important point derived from this statement is the choice of reference operation values. A natural decision would be to use the training period as a

representation of the normal condition. During an abnormal event, each input variable is sequentially replaced by its training average, and the resulting SPE reduction quantifies that variable's contribution. It is expected that when a faulty variable has its value replaced by the training reference, the resulting SPE value (given that an anomaly is in progress) will be reduced. For this to be true, however, the other variables must remain as close as possible to their respective training values. As noted in section 2.2, such an assumption is difficult to guarantee in complex industrial systems that evolve over time, where operational variability and drift may cause these reference conditions to become progressively misaligned with the actual process behavior.

A second way to select a normal representative condition of the process would be to keep track of the last normal operation window before the fault occurs. By tracking the last window where SPE remained below the threshold, it is possible to ensure that reference values always reflect the current operating regime. This brings adherence to operational reality, since, as already mentioned, not every change in the process demands model retraining.

This study defines the "nearest normal window" as the latest continuous segment of data during which the monitoring statistic stays under control ( $SPE \leq \lambda$ ).

The NNV technique integrates drift detection with counterfactual analysis to provide contribution profiles adherent to process changes. The algorithm proceeds as follows:

- Continuous drift monitoring: A sliding window tracks the SPE index profile, and a drift detector identifies change points in SPE behavior.
- Adaptive reference update: If a change point is detected but SPE remains below the fault threshold, the reference window is updated to the period immediately preceding the change point. This captures the "new normal" operating regime. In other words, in the proposed NNV procedure, the "nearest normal window" is not a fixed length hyperparameter. Instead, it is defined as the latest continuous in-control segment ( $SPE \leq \lambda$ ) and is updated online: the reference statistics are recursively updated while the process remains in-control, and they are reset when a change point is detected under

in-control conditions. As a result, the effective reference-window length adapts to the rate of process evolution, which reduces dependence on an arbitrarily chosen window size.

- Counterfactual fault diagnosis: If SPE exceeds the threshold, an anomaly is detected, and for each variable, a local counterfactual sample is generated by replacing the observed value with its reference counterpart drawn from the nearest normal operating window. The reference vector is computed as the mean over this segment. The resulting change in SPE quantifies the diagnostic relevance of the variable. The variable whose substitution produces the largest SPE reduction is identified as the primary contributor.

The pseudocode that performs the previously described steps is displayed in Algorithm 3. Additional implementation details of the NNV architecture, class structure, and diagnostic workflow can be found in Appendix C. Notably, no assumptions are made about the underlying model type or problem structure (supervised or unsupervised), making NNV a model-agnostic framework. The current formulation, however, assumes continuous process variables and a scalar error-based monitoring index, commonly observed in regression tasks.

---

**Algorithm 3** NNV Diagnosis algorithm

---

**Require:** Trained model  $h$   
**Require:** Training data  $\mathbf{X}_{\text{train}}$   
**Require:** SPE threshold  $\lambda$

- 1: Initialize  $\mathbf{X}_{\text{ref}} \leftarrow \text{mean}(\mathbf{X}_{\text{train}})$  (average of training period)
- 2: Initialize  $k \leftarrow \text{len}(\mathbf{X}_{\text{train}})$  (initial window size)
- 3: **while** time > 0 **do**
- 4:     Check for Drift
- 5:     **if** Drift detected AND  $\text{SPE}_{\text{time}} \leq \lambda$  **then**
- 6:          $\mathbf{X}_{\text{ref}} \leftarrow \text{mean}(\mathbf{X}[t - k : t])$
- 7:          $k \leftarrow 1$
- 8:     **if** Drift NOT detected AND  $\text{SPE}_{\text{time}} \leq \lambda$  **then**
- 9:          $k \leftarrow k + 1$
- 10:          $\mathbf{X}_{\text{ref}} \leftarrow \mathbf{X}_{\text{ref}} + (\mathbf{X}(t) - \mathbf{X}_{\text{ref}})/k$  (update reference recursively)
- 11:     **if**  $\text{SPE}_{\text{time}} > \lambda$  **then**
- 12:         **for**  $i = 1$  to  $M$  **do**
- 13:              $\mathbf{X}_{\text{temp}} \leftarrow \mathbf{X}_{\text{time}}$
- 14:              $X_{\text{temp}}(i) \leftarrow X_{\text{ref}}(i)$
- 15:              $\hat{\mathbf{X}}_{\text{temp}} \leftarrow h(\mathbf{X}_{\text{temp}})$
- 16:              $\text{SPE}_{\text{NNV}} \leftarrow \|\mathbf{X}_{\text{time}} - \hat{\mathbf{X}}_{\text{temp}}\|^2$
- 17:              $\Delta\text{SPE}_i \leftarrow \text{SPE}_{\text{time}} - \text{SPE}_{\text{NNV}}$
- 18:             Sort  $\Delta\text{SPE}$

In Algorithm 3,  $k$  is the length of the current in-control segment used to compute  $\mathbf{X}_{\text{ref}}$ . As  $k$  increases, the update gain ( $1/k$ ) decreases, which stabilizes the reference during gradual changes.

Figure 26 depicts the visual workflow model from Algorithm 3. The diagram shows how, at each time step ( $t$ ), the SPE is computed and simultaneously assessed by a drift detector. When a change point is detected while the process remains below  $\lambda$ , the reference is updated to track the current operating mode, whereas when SPE exceeds the threshold, the reference is held fixed and a counterfactual loop is executed in which each variable is replaced by its reference value to quantify the SPE reduction and rank the most influential contributors.

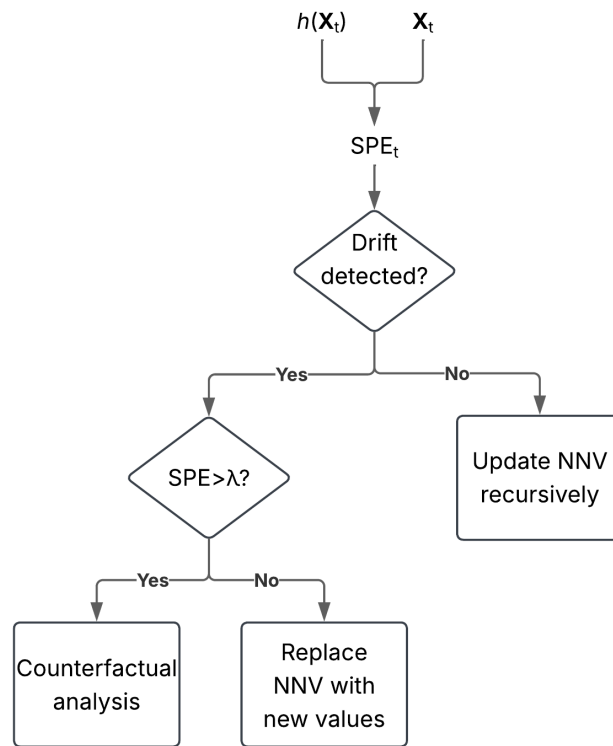


Figure 26: Workflow of NNV diagnosis procedure at a given time ( $t$ ).

To enable online identification of operating mode changes directly from the monitoring stage, the ADWIN method was adopted as the drift detector. As mentioned before, ADWIN relies on statistical tests to identify distributional changes between recent and past observations, without assuming a fixed window size or prior knowledge of data distribution (BIFET, GAVALDÀ, 2007). The

method was selected because it operates in a fully online, sample-by-sample way, which matches the sequential monitoring loop where each new observation is processed individually and a decision must be made before the next sample arrives. Unlike batch-based change-point methods, ADWIN does not require accumulating a fixed number of observations before testing for a distributional shift. Also, ADWIN automatically adapts its window size without requiring the user to specify a fixed window length a priori. This is particularly important in the present application because, upon detecting drift, the algorithm's internal window width is directly used to determine how many recent normal-operation samples should be included when resetting the reference values. A fixed-window approach would impose an arbitrary horizon that may be too short to capture the current operating regime or too long to respond promptly to process changes. Finally, ADWIN provides theoretical guarantees on both the false positive rate and the detection delay, ensuring that the monitoring framework has well-characterized statistical behavior regardless of the underlying distribution of the SPE values. Further algorithmic details, statistical guarantees on false alarms, and examples of use of ADWIN can be found in the literature (BIFET, GAVALDÀ, 2007).

The ADWIN algorithm implementation used in this work was provided by the River library for Python, version 0.22.0 (MONTIEL, HALFORD, *et al.*, 2020). ADWIN detector was configured with  $\delta=0.01$ , corresponding to a 1% bound on the false alarm probability per test. The SPE statistic was fed as a scalar input at each time step. Upon drift detection, if the current SPE did not exceed the control limit, the NNV reference values were reset using the  $w$  most recent samples, where  $w$  is the adaptive window width maintained by ADWIN; otherwise, only the detector was reinitialized. After each detection event, a new ADWIN instance was created with the same  $\delta$  to restart the monitoring window. All remaining parameters were kept at their default values in the River library.

By coupling ADWIN with counterfactual diagnosis, the reference state is continuously adapted to the most recent normal operating regime, preventing outdated baselines, such as data obtained from the training period, from biasing diagnostic conclusions.

## 4.2 Results and discussion

To validate the proposed NNV contribution methodology, evaluation was conducted using three datasets arranged in increasing order of complexity. The first dataset is a controlled, low-dimensional autoregressive benchmark (PORTNOY, MELENDEZ, *et al.*, 2016), which provides an interpretable environment to isolate and understand the behavior of contribution signatures under well-defined disturbances. The second dataset comprises the Tennessee Eastman Process (TEP), discussed in section 3.2.2. Finally, in order to evaluate the method under practical industrial variability and constraints, the third dataset consists of real operational measurements from an offshore oil fiscal metering system similar to the case presented in section 3.2.4.

For the present work, the BibMon library (MELO, LEMOS, *et al.*, 2024), implemented in Python, was used as the implementation framework. BibMon provides a platform featuring regression and reconstruction models, data preprocessing pipelines, alarm systems, and visualization tools including control charts and diagnostic maps (MELO, LEMOS, *et al.*, 2024). Its interface is inspired by well-known machine learning frameworks such as Scikit-Learn (PAPER, 2020, PEDREGOSA, WEISS, *et al.*, 2011) to facilitate usability within the data science community.

Among the techniques available in BibMon, PCA was chosen as the fault detection model for all cases presented in this study. This decision reflects the fact that the primary goal of this research is to address the diagnostic challenge that follows detection, rather than the fault detection performance itself. While more advanced models may improve detection sensitivity for specific fault types, the model-agnostic nature of the proposed NNV framework ensures that diagnostic principles validated with PCA are applicable to other model architectures. Thus, the use of PCA enables a simple proof of the essential proposition while avoiding additional methodological complication. A detailed description of PCA-based monitoring for industrial applications can be found in (MELO, CÂMARA, *et al.*, 2024a) and, more concisely, in Appendix A.

#### 4.2.1 LINEAR AUTOREGRESSIVE SYSTEM

Prior to validation on complex industrial benchmarks, simpler controlled models provide valuable insights into the fundamental behavior of monitoring methods. One example of a controlled environment is the autoregressive model presented by (KU, STORER, *et al.*, 1995) and adapted later by (PORTNOY, MELENDEZ, *et al.*, 2016) for researching adaptive monitoring strategies.

The model consists of two outputs ( $\mathbf{y}_1, \mathbf{y}_2$ ) and two control inputs ( $\mathbf{u}_1, \mathbf{u}_2$ ), governed by the following equations:

$$\mathbf{z}(t) = \mathbf{A}_1 \mathbf{z}(t-1) + \mathbf{B}_1 \mathbf{u}(t-1) \quad \text{Eq. 12}$$

$$\mathbf{y}(t) = \mathbf{z}(t) + \mathbf{V}(t) \quad \text{Eq. 13}$$

$$\mathbf{u}(t) = \mathbf{A}_2 \mathbf{u}(t-1) + \mathbf{B}_2 \mathbf{w}(t-1) \quad \text{Eq. 14}$$

$$\mathbf{X}(t) = [\mathbf{y}(t) \ \mathbf{u}(t)] \quad \text{Eq. 15}$$

where  $\mathbf{z}(t)$  is the latent state vector,  $\mathbf{V}(k)$  represents measurement noise (normally distributed with standard deviation 0.316), and  $\mathbf{w}(t)$  represents external disturbances. Both input  $\mathbf{u}$  and output  $\mathbf{y}$  are measured, but  $\mathbf{z}$  and  $\mathbf{w}$  are not. The final data vector consists of measurements  $\mathbf{y}$  and  $\mathbf{u}$ , concatenated by columns, as  $\mathbf{X}$ . The system matrices are:

$$\mathbf{A}_1 = \begin{bmatrix} 0.118 & -0.191 \\ 0.847 & 0.264 \end{bmatrix}, \quad \mathbf{B}_1 = \begin{bmatrix} 1 & 2 \\ 3 & -4 \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} 0.811 & -0.226 \\ 0.477 & 0.415 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} 0.193 & 0.689 \\ -0.32 & -0.394 \end{bmatrix}$$

This work updated the original model described in (PORTNOY, MELENDEZ, *et al.*, 2016) to provide two distinct anomalies that result in qualitatively different effects on the process. The first adaptation simulates a measurement system error that impacts only the observation of a single variable without affecting the process dynamics. This is accomplished by applying a continuous bias to the output variable  $\mathbf{y}_2$  after a specific time instant ( $t_{\text{faulty}}$ ):

$$\mathbf{y}_2^{\text{faulty}}(t) = \mathbf{y}_2(t) + \mathbf{b}, \quad t \geq t_{\text{faulty}} \quad \text{Eq. 16}$$

where  $\mathbf{b}$  is the bias magnitude. This sort of failure corresponds to common industrial scenarios such as sensor calibration drift, transmitter offset, and signal processing issues.

The second adaptation simulates a change in the underlying process dynamics, representing an operating mode transition. This is implemented by modifying the system matrices  $\mathbf{A}_1$ ,  $\mathbf{B}_1$ , and  $\mathbf{A}_2$ :

$$\mathbf{A}_1^{\text{new}} = \mathbf{A}_1 + \omega \cdot \Delta\mathbf{A}_1 \quad \text{Eq. 17}$$

$$\mathbf{B}_1^{\text{new}} = \mathbf{B}_1 + \omega \cdot \Delta\mathbf{B}_1 \quad \text{Eq. 18}$$

$$\mathbf{A}_2^{\text{new}} = \mathbf{A}_2 + \omega \cdot \Delta\mathbf{A}_2 \quad \text{Eq. 19}$$

where  $\omega$  is an intensity parameter, and the perturbation matrices are:

$$\Delta\mathbf{A}_1 = \begin{bmatrix} 0.15 & -0.10 \\ 0.12 & 0.08 \end{bmatrix}, \quad \Delta\mathbf{B}_1 = \begin{bmatrix} 0.40 & -0.25 \\ -0.25 & 0.60 \end{bmatrix}, \quad \Delta\mathbf{A}_2 = \begin{bmatrix} 0.04 & -0.06 \\ 0.03 & 0.06 \end{bmatrix}$$

The perturbation matrices, together with the intensity parameter  $\omega$ , were selected empirically to induce signal changes of a magnitude comparable to the bias case presented in Eq. 16, while remaining detectable by the monitoring model used in this study.

This form of adjustment exerts an impact on the dynamic interactions between all process variables since the matrices  $\mathbf{A}_1$ ,  $\mathbf{B}_1$ , and  $\mathbf{A}_2$  determine how the system evolves over time. This scenario is similar to operating mode transitions in industrial processes, in which setpoint changes or production regime alterations affect the correlation structure among process variables in a coordinated manner.

Figure 27 and Figure 28 show the datasets created for each faulty operation by setting the number of points simulated and a  $t_{\text{faulty}}$  as 50% of the total duration.

Figure 29 shows SPE values, generated by the PCA-based monitoring model developed in the BibMon library, indicating that both disturbances resulted in SPE values above  $\lambda$ . During this examination, the number of principle components was

chosen to account for 90% of the explained variation in the input data for both PCA models.

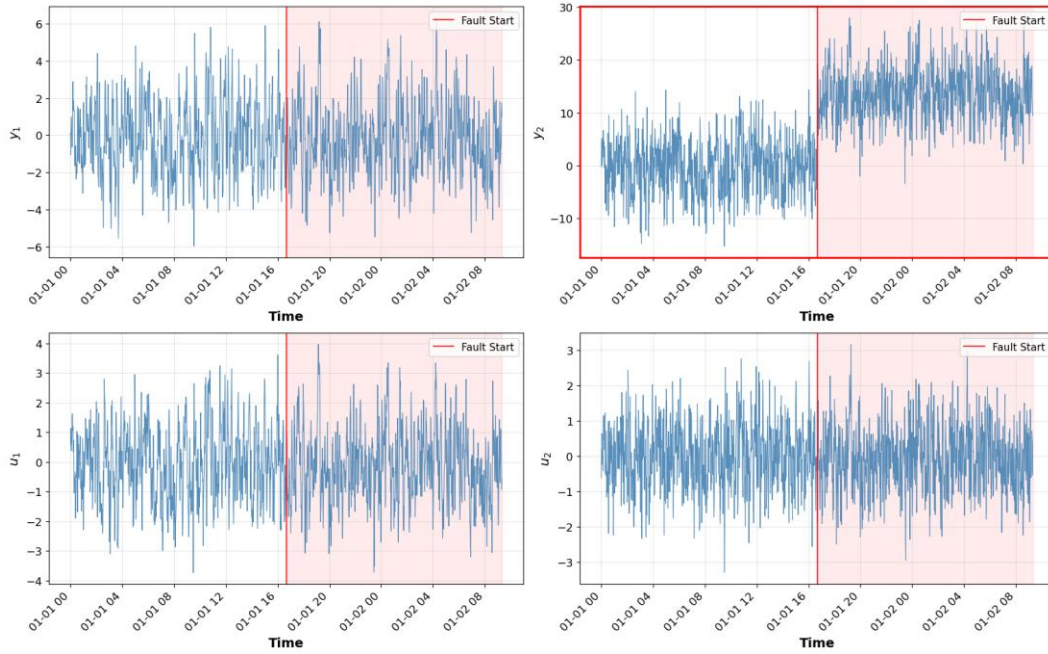


Figure 27: Time series of process variables under spurious fault. A bias introduced in  $y_2$  at sample 1000 affects only the measured variable while other variables remain unaltered.

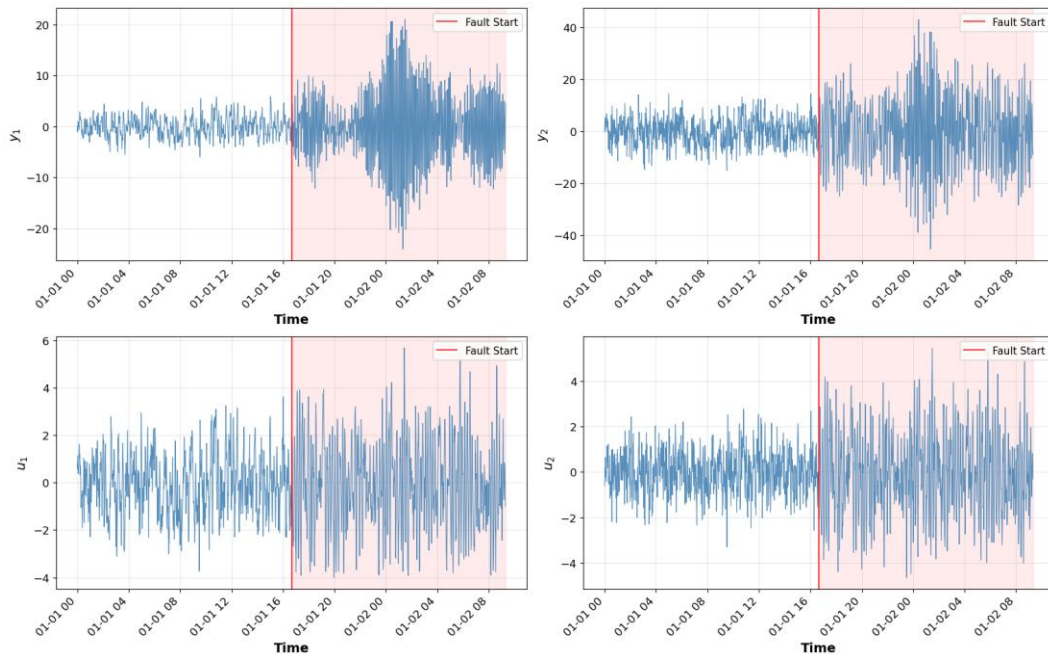


Figure 28: Time series of process variables under operating mode change. Matrix modifications at sample 1000 affect all variables through their dynamic interconnections.

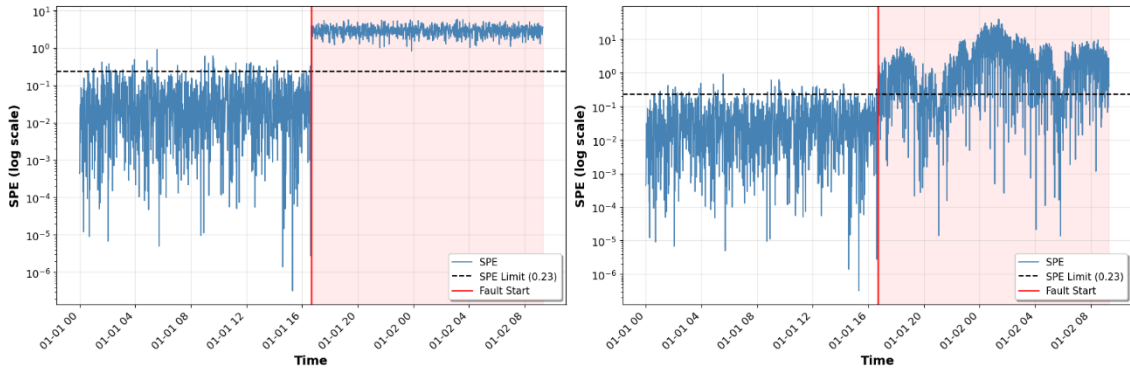


Figure 29: SPE values for the linear autoregressive system with bias (left) and with operating mode change (right). The horizontal dashed line indicates the SPE control limit  $\lambda$ , obtained from the training period.

As indicated by the SPE trends, both disturbance mechanisms trigger comparable detection behavior. Once SPE exceeds the threshold, diagnostic interpretation is transferred to the contribution stage. The SPE profiles were analyzed according to the procedures described in Algorithm 3 and depicted in Figure 26: SPE was monitored sequentially, drift events under in-control conditions updated the nearest-normal reference, and alarms ( $SPE > \lambda$ ) triggered the NNV counterfactual loop to rank variable contributions. Figure 30 depicts the average NNV contribution profiles computed for both datasets, comparing normal-operation contributions against those observed under the disturbances analyzed.

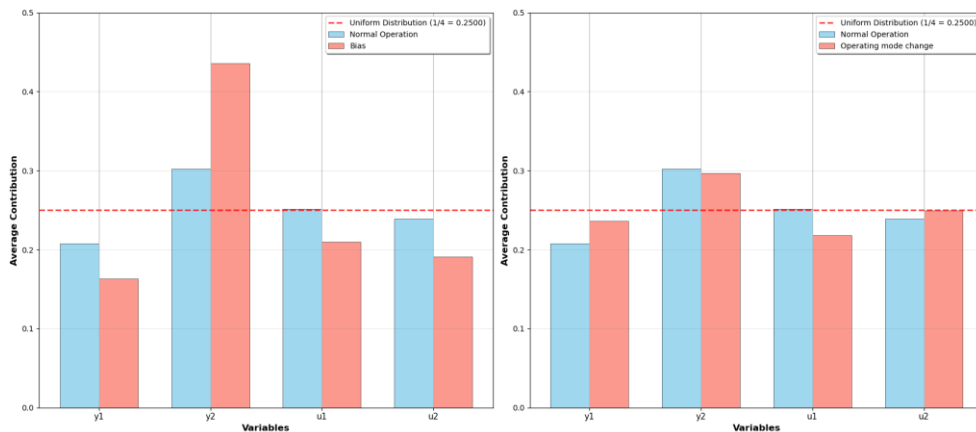


Figure 30: Side-by-side comparison of NNV contributions for the linear autoregressive system with bias (left) and with operating mode change (right).

As Figure 30 shows, for the faulty scenario, where a constant bias was introduced to the  $y_2$  measurement, the NNV analysis shows a concentration of contributions in the affected variable. In other words, replacing  $y_2$  by its nearest-normal reference value removes most of the inconsistency, concentrating the NNV contribution on that variable. This pattern reflects the localized behavior of sensor-related anomalies, where the fault affects a single signal without propagating through the process dynamics, according to H1.

In contrast, the operating mode change scenario resulted in a different contribution pattern. When the system matrices  $\mathbf{A}_1$ ,  $\mathbf{B}_1$ , and  $\mathbf{A}_2$  are modified to simulate a change in process dynamics, the NNV contributions are spread across multiple variables rather than concentrated in a single measurement. This distributed pattern occurs when changes to the system matrices affect the dynamic relationships among all process variables at the same time, as stated by H2, resulting in coordinated deviations from the normal operating region captured by the PCA model.

These findings from the linear autoregressive system model provided a controlled environment for testing the NNV methodology before its application to more complex industrial benchmarks, such as the TEP system.

#### 4.2.2 TENNESSEE EASTMAN PROCESS (TEP)

The Tennessee Eastman Process (TEP), previously detailed in Section 3.2.2, is employed again in this study using the extended dataset provided by REINARTZ, KULAHCI, *et al.*, (2021). While the physical process remains the same, the expanded dataset includes simulations across all six operating modes originally defined by DOWNS, VOGEL, (1993) and summarized in Table 7.

The final dataset comprises 500 independent simulations per scenario with varied random seeds, ensuring statistical robustness, and totaling over 100 GB of data to be processed and analyzed. This broader scope is particularly useful for evaluating monitoring approaches in realistic scenarios when operating mode changes coexist with real faults. This study used all five transitions from Mode 1

(Mode 1 to Mode 2, Mode 1 to Mode 3, Mode 1 to Mode 4, Mode 1 to Mode 5, and Mode 1 to Mode 6), since Mode 1 is the steady-state baseline most commonly adopted in conventional TEP studies and provides a consistent reference for quantifying model mismatch across regimes. Also, specific fault scenarios (IDVs 4, 7, and 14) were evaluated from all operation modes. These fault scenarios were selected based on their proven detectability using traditional multivariate statistical techniques (REINARTZ, KULAHCI, *et al.*, 2021, YIN, Shen, DING, *et al.*, 2012).

Table 7: Steady-state different operation modes of the TEP dataset.

<b>Mode</b>	<b>G/H mass ratio</b>	<b>Production (Stream 11)</b>
1	50/50	14076 kg/h
2	10/90	14076 kg/h
3	90/10	11111 kg/h
4	50/50	maximum
5	10/90	maximum
6	90/10	maximum

Following the procedure adopted in section 4.2.1, the same BibMon workflow was applied to the TEP dataset. This application produced trained models and corresponding diagnostic results for three selected fault scenarios (IDV4, IDV7, and IDV14) evaluated under each of the six steady-state operating modes available in the extended TEP benchmark (REINARTZ, KULAHCI, *et al.*, 2021).

It is important to note that, in order to provide a unified view of fault behavior across different operating regimes, the results presented consist of averaged values for each fault type. In other words, given an IDV, both the contribution profiles and the SPE values represent the average across all six TEP operating modes. Figure 31 shows the SPE charts obtained for the selected TEP fault scenarios (IDV4, IDV7, and IDV14). In this analysis, the number of principal components was chosen to retain 90% of the variance in the input data.

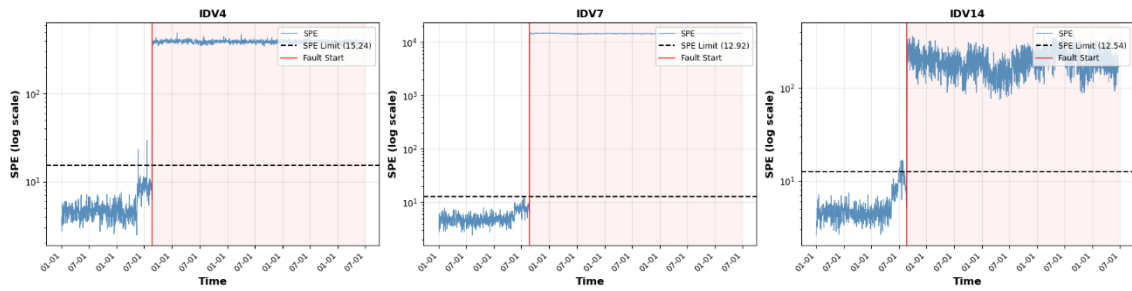
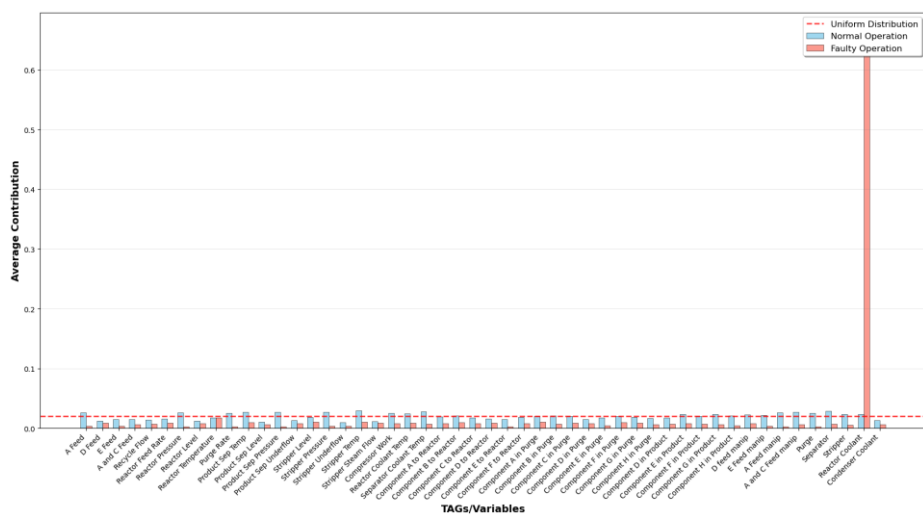


Figure 31: SPE values for the selected TEP fault scenarios considered in this study (IDV4, IDV7, and IDV14). The horizontal dashed line indicates the SPE control limit  $\lambda$ , obtained from the training period.

Figure 32 shows the NNV contribution profiles corresponding to each IDV scenario from Figure 31.

As one can see in Figure 32, a consistent observation across all three fault types is the change in the distribution of contributions when comparing the two periods. Under normal operating conditions (blue-colored bars), the contributions are distributed relatively uniformly across the 52 process variables. Most values remain close to the uniform distribution reference line ( $1/52 \approx 0.019$ ), indicating that no single variable dominates the deviation from the nominal state. This behavior is expected, as the NNV methodology computes contributions relative to the nearest normal reference, and during normal operation the process state remains close to this reference for all variables.



(a)

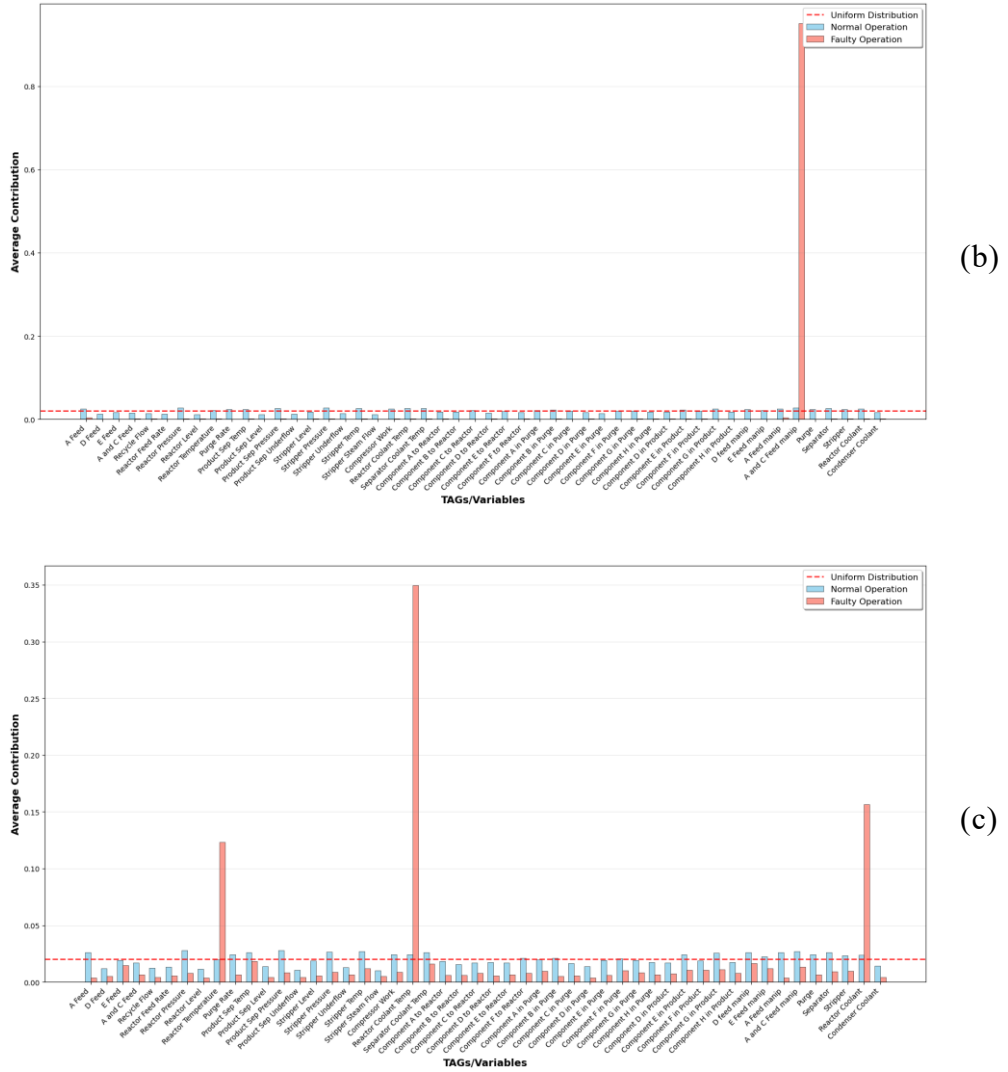


Figure 32: Average NNV contributions for: (a) IDV4, (b) IDV7 and (c) IDV14 through all six TEP operating modes.

When a fault is introduced, the contribution pattern changes, showing a redistribution of contributions where certain variables exhibit values substantially above the uniform baseline while others fall below it. This concentration effect happens because faults typically affect specific subsystems or variables in the process, causing those variables to deviate more from their normal reference values. Since the contributions are normalized, an increase in contribution from faulty variables necessarily reduces the relative contribution of others.

From a diagnostic perspective, the variables exhibiting elevated contributions during fault conditions are candidates for root cause investigation, as they are most

affected by or responsive to the introduced disturbance. Figure 33 shows the SPE values for the five TEP transitions starting from Mode 1, together with the control limit  $\lambda$ .

As shown in Figure 33, all the SPE trends associated with mode transitions exceeded the control limit  $\lambda$  obtained from the nominal (Mode 1) training regime. In all cases, SPE levels rise above this limit by orders of magnitude, indicating a mismatch between the baseline model and the new steady-state condition presented in Table 7. As a matter of fact, the SPE values in Figure 33 can reach even higher values than those observed in Figure 31. This comparison suggests that, for operating mode transitions, the high SPE value is due to the inability of the model trained under Mode 1 to represent the process behavior under the new regime. In this sense, the SPE profiles presented in Figure 33 can be interpreted as a model fault detected by a change in operating conditions.

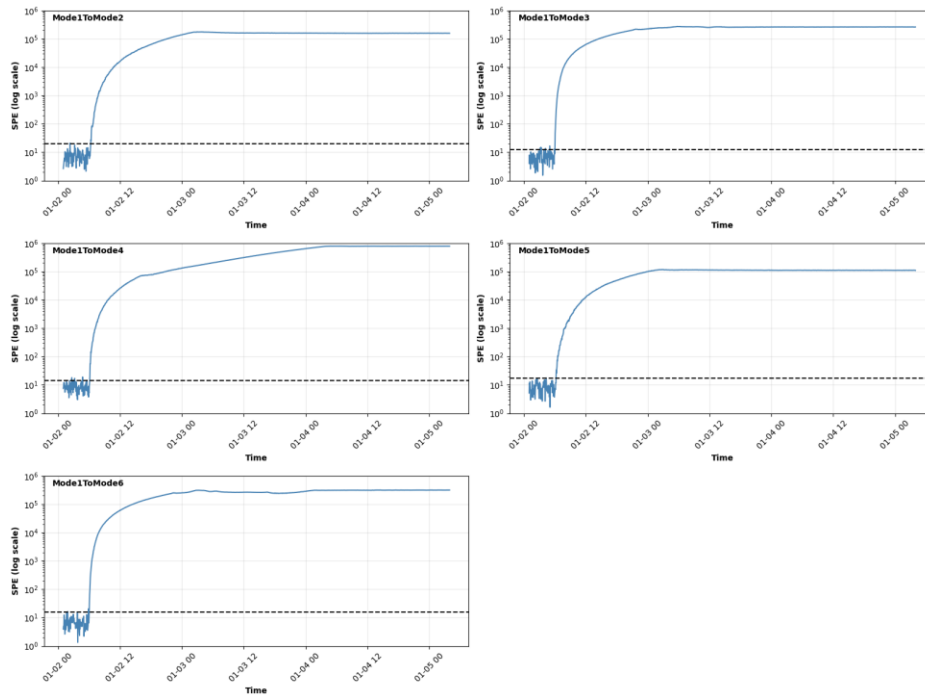
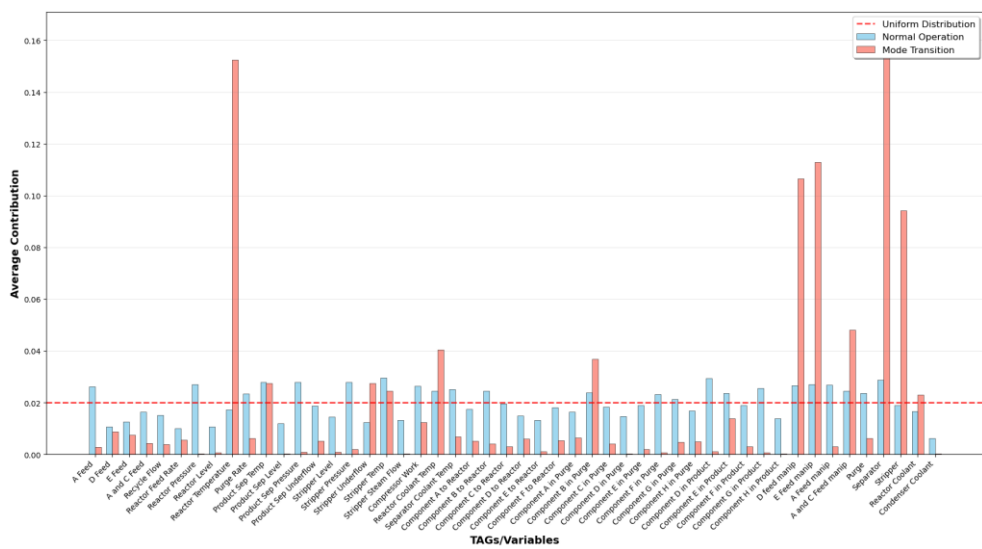
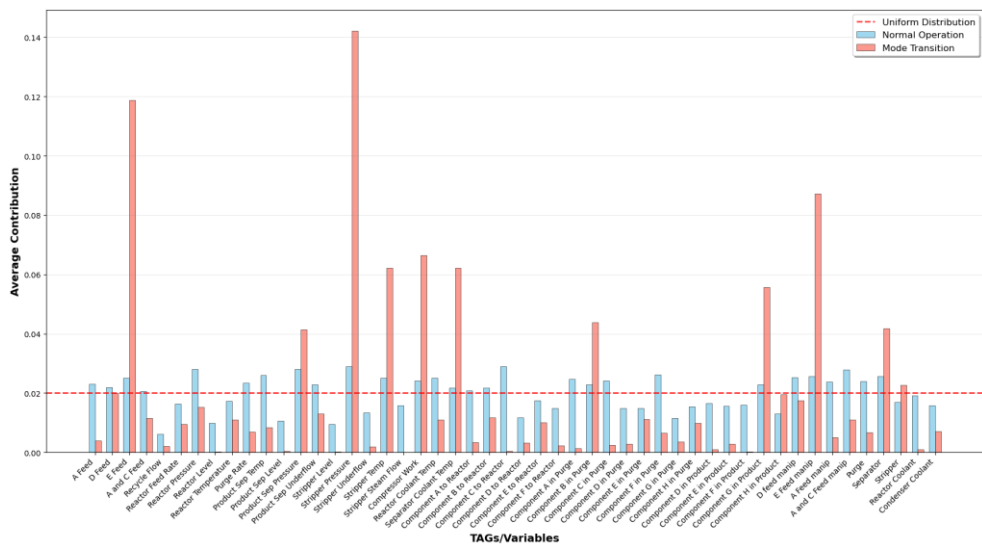
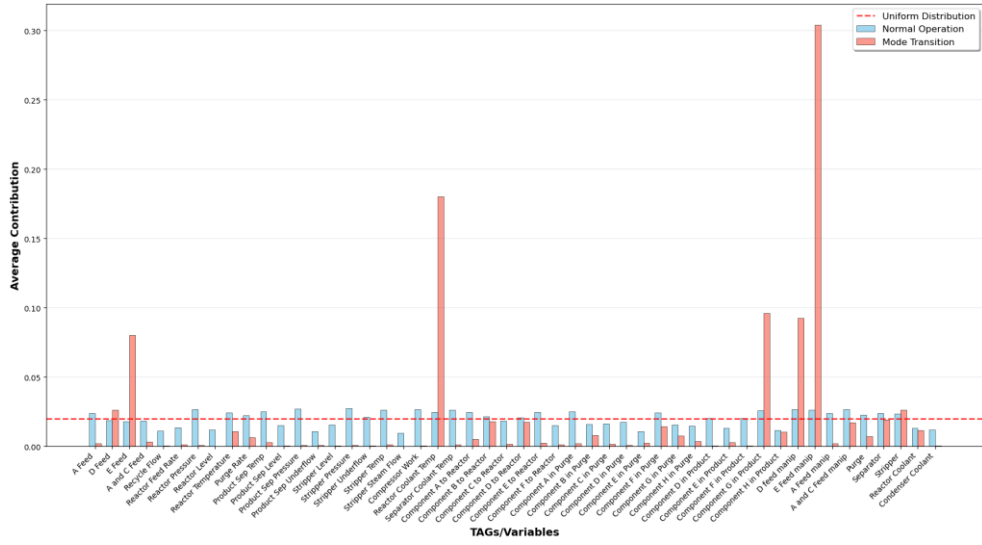


Figure 33: SPE values for TEP, during operating mode transitions from Mode 1. The horizontal dashed line indicates the SPE control limit  $\lambda$ , obtained from the training period.

Figure 34 shows the NNV contribution profile for transitions departing from Mode 1.



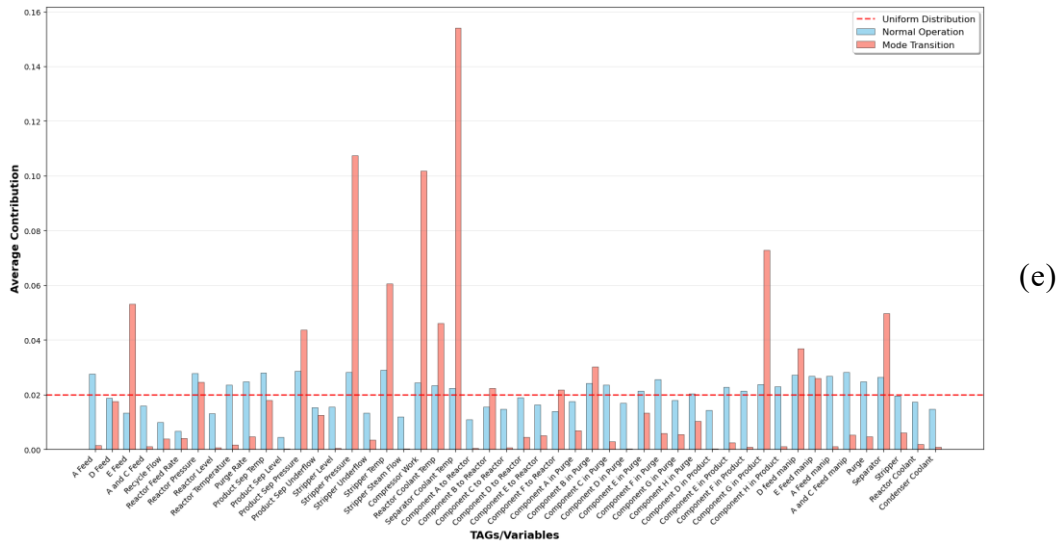
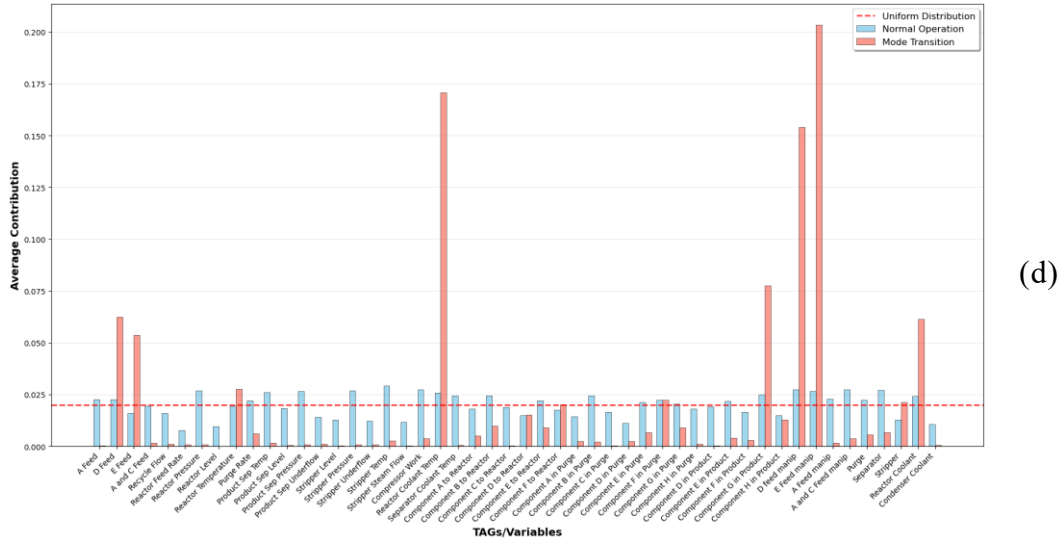


Figure 34: NNV contributions for operating mode transitions: (a) Mode 1 to Mode 2, (b) Mode 1 to Mode 3, (c) Mode 1 to Mode 4, (d) Mode 1 to Mode 5, and (e) Mode 1 to Mode 6.

Contrary to the fault cases presented in Figure 32, where contributions seemed to concentrate on a relatively small subset of variables, Figure 34 exhibits a different signature. In these cases, the contribution redistribution remains spread across many tags, with multiple variables showing moderate differences relative to the uniform reference line rather than a few dominant peaks. Once again, this distributed pattern is consistent with the interpretation given by H2, in which the

operating mode transitions correspond to coordinated, process-wide adjustments required to reach a new regime.

The uniform distribution reference line is used as an interpretive tool in these analyses. Under ideal normal operation, with no anomalies, the normalized contributions should approximate a uniform distribution. This means that deviations from the nominal state are evenly distributed across all measured variables. Deviations from this reference are caused by both process faults and changes in operating mode, but their contribution patterns differ fundamentally. Various metrics, such as entropy-based indices and sparsity metrics, can be used to measure the dispersion or concentration of numerical vectors. In the present study, the dispersion of NNV contribution was quantified using the  $L_1/L_2$  norm ratio because it provides an intuitive geometric interpretation, is scale-invariant, and can be computed with  $O(M)$  cost without requiring sorting or logarithms. The  $L_1/L_2$  ratio ranges from 1, corresponding to maximum sparsity with all contribution concentrated in a single variable, to  $\sqrt{M}$ , corresponding to a perfectly uniform distribution across all  $M$  variables. The normalized dispersion index  $S$  can be calculated by:

$$S = \frac{(L_1/L_2) - 1}{\sqrt{M} - 1} \quad \text{Eq. 20}$$

where:

$$L_1 = \sum_{i=1}^M |C_i| \quad \text{Eq. 21}$$

$$L_2 = \sqrt{\sum_{i=1}^M C_i^2} \quad \text{Eq. 22}$$

In Eq. 20, values close to zero indicate highly concentrated, fault-like signatures, and values close to one indicate distributed patterns typically associated with operating mode changes. Within this framework, the uniform reference line represents the case of maximum dispersion and provides a natural baseline for interpreting the contribution patterns. Table 8 summarizes the average values of  $S$

obtained for the fault scenarios (IDV4, IDV7, and IDV14) and for the operating mode transitions evaluated in the TEP dataset. To quantify the statistical uncertainty of  $S$ , a percentile bootstrap approach with 10,000 resamples was adopted. The resampling unit was defined at the operating mode level: at each bootstrap iteration, operating modes were resampled with replacement, the corresponding contribution profiles were aggregated to form a mean contribution vector, and  $S$  was evaluated on this bootstrap-aggregated vector. The 2.5th and 97.5th percentiles of the resulting bootstrap distribution were used to define the 95% confidence interval (CI).

Table 8: Average values of the normalized dispersion index  $S$  computed from NNV contribution patterns for the TEP dataset.

	$\bar{S}$	Deviation from normal operation (%)
Normal operation	$0.95 \pm 0.005$	0
Faulty operation	$0.11 \pm 0.046$	88
Mode transition (operational change)	$0.35 \pm 0.013$	63

As one can see, in the faulty cases, the mean values of  $S$  are lower than in normal operation, indicating smaller dispersion levels as expected under faulty conditions. In contrast, the operating mode change  $\bar{S}$  has higher dispersion values and thus smaller deviations from normal operation, indicating that contributions are more evenly distributed across variables. Unlike fault scenarios, these transitions do not cause sparsity in contribution patterns, but rather result in broader adjustments that are consistent with coordinated changes in the process. The reported 95% bootstrap confidence intervals quantify the uncertainty of these aggregate patterns and show that the separation between fault-like and transition-like signatures is not attributable to sampling variability.

Following these observations, two simple supervised classifiers, a Decision Tree (BREIMAN, FRIEDMAN, *et al.*, 2017) and a Logistic Regression model (COX, 1958), were trained using the NNV contribution vectors as input features to determine whether fault and transition scenarios could be systematically separated. The classifiers were trained using contribution samples from the fault analysis and the operating mode transition analysis combined into a single labeled dataset with two classes. Since the contribution vectors originate from time series data with

inherent temporal autocorrelation, a group-aware data splitting strategy was adopted to prevent temporal data leakage. The train/test partition was performed using GroupShuffleSplit with a 70%/30% ratio, ensuring that entire groups were kept exclusively in either the training or test set, thereby reducing the problems associated with temporal dependence or within-group correlation. Model generalization was assessed through 5-fold cross-validation using GroupKFold, where each fold preserved the integrity of the entire time series. Both methods are available in the Scikit-Learn package (PEDREGOSA, WEISS, *et al.*, 2011). A fixed random seed was employed to ensure reproducibility of all results. Figure 35 and Figure 36 show, respectively, the confusion matrices and ROC (Receiver Operating Characteristic curve) curves obtained from the classifiers.

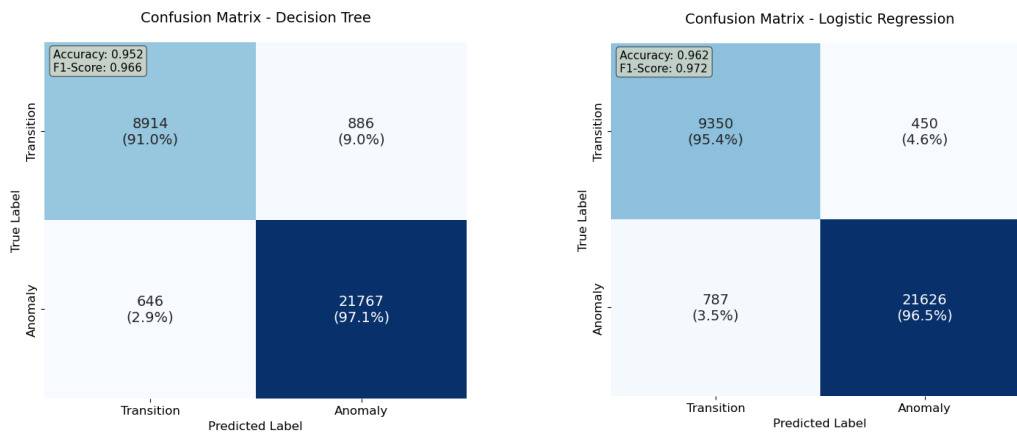


Figure 35: Confusion matrices for binary classification using NNV contributions.

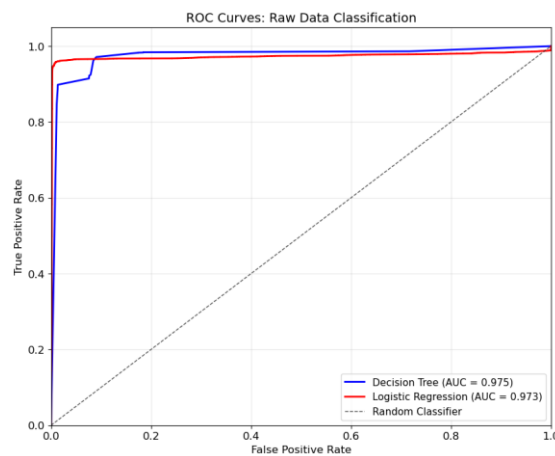


Figure 36: ROC curves for fault vs. operating mode changes classification using NNV contributions.

As one can observe, even when employing basic classifiers, such as Decision Tree or Logistic Regression trained directly on the NNV contribution vectors, the resulting confusion matrices and ROC curves show that the two classes are separable and perform effectively on the test set (30% of the NNV contribution dataset), with both false positive and false negative rates remaining below 10%. Overall, these findings suggest that NNV analysis contribution profiles contained discriminating information capable of distinguishing between equipment or sensor problems and operating mode changes.

The next subsection revisits this distinction using real-world industry datasets. It is worth noting, however, that because these industrial datasets do not provide a sufficiently large, consistently labeled sample to train a supervised model comparable to the TEP benchmark, the analysis in section 4.2.3 focuses on whether the expected distributed and concentrated contribution signatures remain consistent when comparing an operating mode change to a true fault occurrence.

### 4.2.3 OIL FLOWMETER

As discussed earlier in section 3.2.4, fiscal meters are critical instruments in offshore oil and gas operations, given their direct impact on custody transfer, revenue allocation, and regulatory compliance. In this second part of the thesis, the same fiscal metering system is revisited — now under an unsupervised monitoring framework — to investigate how the proposed drift-aware diagnostic approach performs when confronted with real industrial process variability.

For the present evaluation, datasets from two offshore platforms were used, referred to as Real Data 1 and Real Data 2. Real Data 1 comprises 9,583 observations from four process variables for model training, including liquid level, temperature, pressure, and flow measurements from a fiscal metering station. Real Data 2 includes 33,423 observations across five process variables for model training, consisting of flow rate, pressure, and temperature measurements from a fiscal metering system. The test period selected for Real Data 1 corresponds to a time interval in which a documented fault was developing in one of the monitored

variables. In contrast, the test period for Real Data 2 was taken from a distinct, yet operationally acceptable, regime in which the platform experienced an operating mode change. This shift altered the underlying process behavior and made the original model no longer representative of the process.

By applying the same methodology outlined in previous sections two independent monitoring models were produced. The results presented in this subsection compare NNV contribution patterns obtained under a fault scenario (Real Data 1) to those obtained under an operating mode transition scenario (Real Data 2). Figure 37 depicts the SPE profile produced for Real Data 1 dataset during the test period, showing how the monitoring index evolves during the fault period, moving from values below the control limit to levels potentially indicative of anomalous behavior. The PCA model retained the minimum number of principal components required to capture 95% and 90% of the total variance in the input space for Real Data 1 and Real Data 2, respectively. Figure 38 presents the NNV contribution results for Real Data 1, while Figure 39 and Figure 40 show, respectively, the SPE values for Real Data 2 during an operating mode change and its NNV contribution analysis.

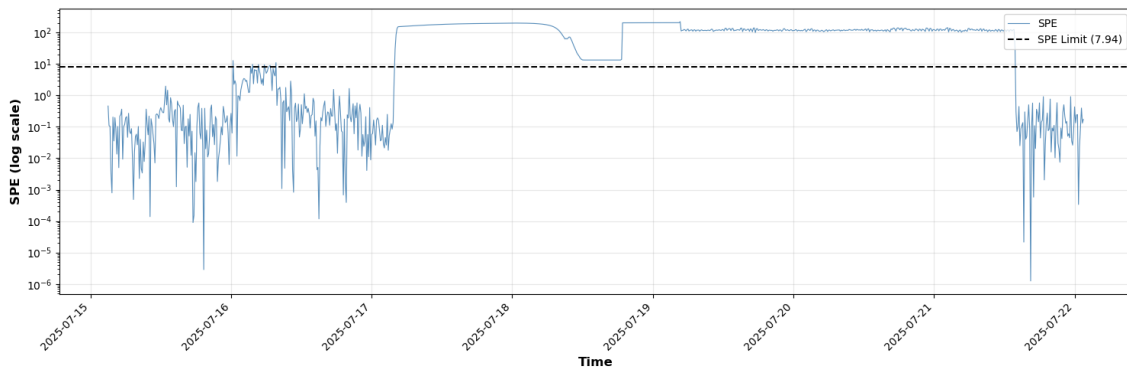


Figure 37: SPE values for Real Data 1, during a fault in one of the input variables. The horizontal dashed line indicates the SPE control limit  $\lambda$ , obtained from the training period.

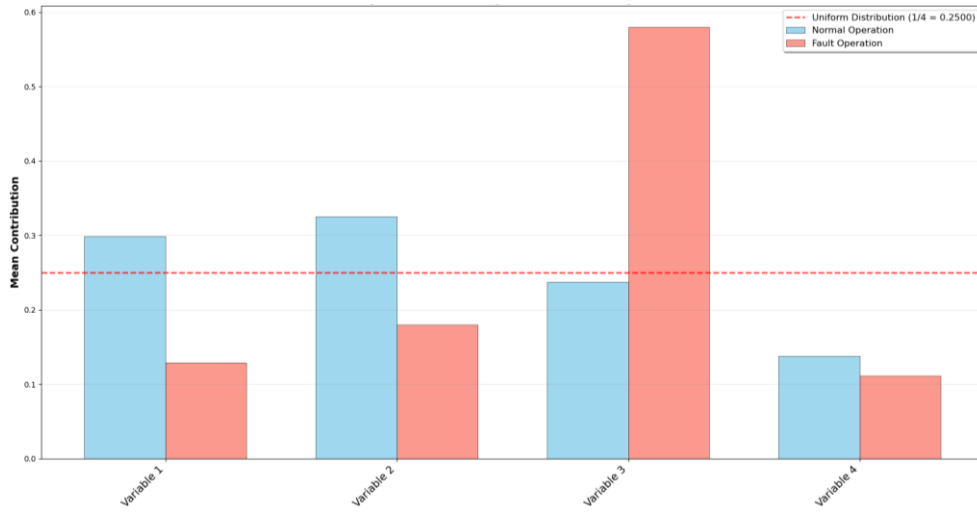


Figure 38: NNV contributions for Real Data 1.

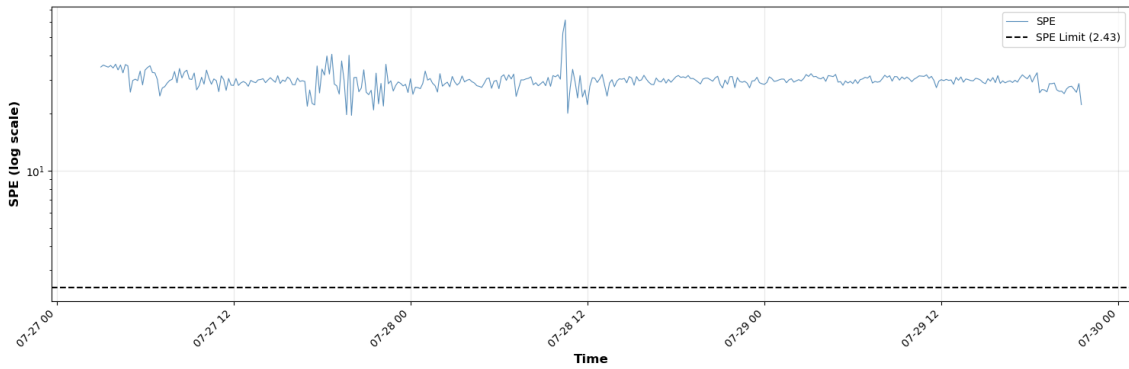


Figure 39: SPE values for Real Data 2, during an operating mode change. The horizontal dashed line indicates the SPE control limit  $\lambda$ , obtained from the training period.

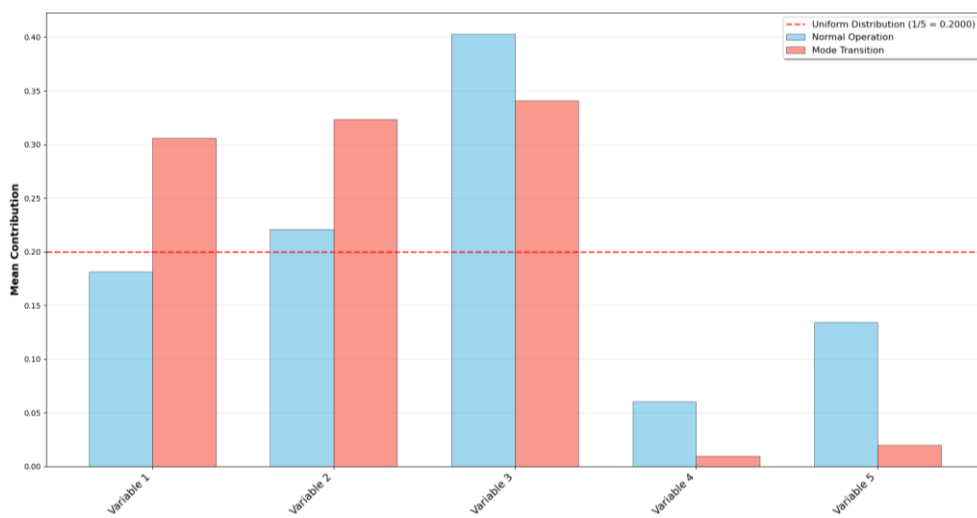


Figure 40: NNV contributions for Real Data 2.

The contribution pattern during the fault period differs between the two scenarios studied: while mode transitions in Real Data 2 caused distributed contribution changes that affected multiple variables, the fault in Real Data 1 produced a more concentrated contribution in Variable 3, which is the true source of the anomaly. Table 9 presents the dispersion index  $S$  for Real Data 1 and Real Data 2, calculated according to Eq. 20. It is important to note that, unlike the results presented in Table 8, where both normal and faulty conditions were derived from the same dataset and the normal operation reference was common through all cases, Table 9 reports dispersion values computed from two distinct industrial datasets. Therefore, two separate normal operation baselines are presented: one corresponding to Real Data 1 and another corresponding to Real Data 2.

Uncertainty quantification followed the same percentile-bootstrap principle described previously (10,000 resamples and 95% CI), but employed a different resampling strategy to account for temporal dependence in the industrial data streams. Specifically, a block bootstrap was used, in which contiguous time segments were resampled with replacement until the resampled dataset matched the original sample size. The mean contribution vector was computed for each resampled sequence, and the dispersion index  $S$  was then evaluated on that vector. This procedure was applied independently to the normal-operation and the test-period contribution matrices, so that each condition received its own confidence interval.

Table 9: Normalized dispersion index  $S$  computed from NNV contribution patterns for Real Data 1 and Real Data 2.

	<b>S</b>	<b>Deviation from normal operation (%)</b>
Normal operation / Faulty operation	$0.92 \pm 0.003$ / $0.58 \pm 0.034$	37
Normal operation / Operational change	$0.76 \pm 0.011$ / $0.63 \pm 0.001$	17

As Table 9 shows, even with the reduced dimensionality of these industrial datasets (4 and 5 variables, compared to 52 in the TEP dataset), the magnitude of

change differs substantially across scenarios. In Real Data 1,  $S$  drops from  $0.92 \pm 0.003$  to  $0.58 \pm 0.034$  (37% deviation), indicating a clear increase in contribution concentration during the documented fault. In contrast, in Real Data 2,  $S$  changes from  $0.76 \pm 0.011$  to  $0.63 \pm 0.001$  (17% deviation), reflecting a milder redistribution consistent with an operating-mode transition. The bootstrap confidence intervals show that these shifts are not attributable to sampling variability in the contribution estimates. This distinction, which has been observed systematically in simulated benchmarks and real industrial data, supports the hypothesis that the retraining dilemma can be addressed through a fault-diagnosis framework, with the NNV integration of drift detection and contribution analysis serving as a mechanism to connect model adequacy to real equipment and sensor faults scenarios.

### 4.3 Concluding remarks

This part of the thesis addressed the retraining challenge in data-driven process monitoring by reframing model drift as a diagnosable condition instead of a model maintenance concern. From this point of view, an apparent anomaly may reflect a mismatch between the trained model and the current operating mode, so that the detected fault can be interpreted as the inability of the model to represent the process under different conditions, not necessarily in the process itself.

To prove this viewpoint, the study introduced the NNV contribution analysis, a model-agnostic diagnostic framework that combines drift monitoring with counterfactual analysis. The technique was evaluated using three datasets of increasing complexity: a controlled linear autoregressive benchmark, the expanded TEP dataset, and real operational data from offshore fiscal measurement systems. These case studies were selected to determine if contribution patterns generated by NNV were comprehensible and consistent.

The findings corroborate hypotheses H1 and H2 presented in Section 4.1, indicating that retraining decisions may be addressed using a fault diagnosis framework. This work also demonstrated, in both TEP and real industrial case studies, that the normalized dispersion index  $S$  provides a compact quantitative

characterization of NNV contribution values. In this context, combining drift detection with contribution analysis, just as depicted in Figure 25, using the proposed NNV method creates a structured relationship between detection and diagnosis.

# Chapter 5

## Conclusions and Future Directions

This thesis addressed the practical and conceptual challenges of deploying data-driven monitoring in industrial environments by framing process monitoring as a workflow that goes beyond the detection algorithm itself. In this view, successful implementations depend on two stages that are often under-specified in the literature: (i) decisions made before online detection, notably the selection of representative training data; and (ii) actions taken after detection, especially diagnosis and the decision of whether an alarm reflects a process fault or model inadequacy caused by operating regime changes. Within this context, the main conclusions and contributions of this thesis are summarized below.

### 5.1 Contributions

#### 5.1.1 FORMALIZATION OF INSTANCE SELECTION AS A PRE-DETECTION PROBLEM IN INDUSTRIAL PROCESS MONITORING

This thesis establishes instance selection as a critical modeling decision that directly influences the quality and robustness of data-driven monitoring systems. A two-stage instance selection methodology, named Instance Selection Library (ISLib), was developed to identify representative training instances for regression problems under industrial variability. ISLib combines (i) a clustering-based batch analysis to rank operational regions according to their predictive relevance and (ii) an incremental enlarging-window strategy to determine the minimum training window beyond which additional data no longer improve model generalization. The proposed approach emphasizes interpretability, computational efficiency, and

compatibility with industrial workflows. Its effectiveness is demonstrated using both benchmark data and real industrial datasets.

### **5.1.2 REINTERPRETATION OF CONCEPT DRIFT AS A COMPETING DIAGNOSTIC HYPOTHESIS AFTER FAULT DETECTION**

This thesis introduces a diagnostic perspective in which operating mode changes and model inadequacy are treated as hypotheses that compete with genuine process faults once a detection alarm is triggered. A drift-aware diagnostic framework was proposed to reduce ambiguity between true process faults and model inadequacy due to operating mode changes. The Nearest Normal Value (NNV) contribution method integrates drift detection with counterfactual contribution analysis by updating the reference “nearest normal window” under in-control operation and computing diagnostic relevance through SPE reduction under controlled feature substitution. The results showed that explicitly accounting for drift at the diagnostic stage improves the interpretability and robustness of monitoring decisions.

### **5.1.3 AN INTEGRATED, DEPLOYMENT-ORIENTED MONITORING FRAMEWORK LINKING PRE-DETECTION AND POST-DETECTION DECISIONS**

By combining instance selection before detection and drift-aware diagnosis after detection, this thesis provides an integrated view of data-driven process monitoring that extends beyond algorithmic detection performance. The proposed framework underscores the interdependence between data representativeness, model validity, and diagnostic interpretation, showing that robust monitoring depends equally on decisions made during pre-monitoring data preparation and on the interpretation of alarms after deployment. In this view, effective monitoring is not achieved only by optimizing a detection algorithm, but by ensuring coherence between the data used to build the model and the diagnostic logic applied when that model is challenged in real industrial conditions.

## 5.2 Future Directions

Future research directions follow naturally from the thesis contributions and from the requirements of industrial deployment.

### **ISLib development:**

- **Alternative region discovery and validation:** Replace or complement K-means clustering with approaches better suited to more complex regimes and heterogeneous densities, while preserving interpretability for process engineers.
- **Scalable implementation and streaming integration:** Although ISLib already converges quickly in the evaluated cases, large-scale deployments would benefit from incremental updates and approximate clustering strategies compatible with historian-scale datasets.

### **NNV development:**

- **Real-time implementation and overhead characterization:** The next step is to deploy NNV in online monitoring environments and quantify computational overhead, latency, and reference update behavior under realistic alarm rates. This includes formalizing how the nearest normal window is stored and updated to avoid memory growth while preserving diagnostic relevance.
- **Quantitative decision rules for retraining:** While this thesis demonstrated that dispersion metrics can characterize contribution patterns, future work should establish interpretable thresholds and statistical guarantees for deciding when an alarm should trigger a process investigation versus model adaptation. This includes evaluating entropy-based or sparsity-based indices alongside the proposed normalized dispersion measure.

- Multiple faults, subtle drifts, and nonlinear models: The separability between fault-like and regime-like contribution signatures should be tested under more challenging scenarios, including multiple simultaneous faults, gradual drifts, and nonlinear monitoring models.

### 5.3 Final Remarks

From a practical standpoint, this thesis shifts the focus of data-driven process monitoring from improving detectors in isolation to improving the decisions surrounding detection. By formalizing how data are selected for model training and how alarms are interpreted under changing operating conditions, the proposed contributions bridge an important gap between theoretical monitoring methods and their industrial deployment. As such, the work provides both methodological advances and practical insights that can support the development of more robust and interpretable monitoring systems in real industrial settings.

# Bibliography

AGRAHARI, S., SINGH, A. K. "Concept Drift Detection in Data Stream Mining: A literature review", **Journal of King Saud University - Computer and Information Sciences**, dez. 2021. DOI: 10.1016/j.jksuci.2021.11.006. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1319157821003062>.

AKIBA, T., SANO, S., YANASE, T., *et al.* "Optuna: A Next-generation Hyperparameter Optimization Framework", 25 jul. 2019. Disponível em: <http://arxiv.org/abs/1907.10902>.

ALAUDDIN, M., KHAN, F., IMTIAZ, S., *et al.* "A Bibliometric Review and Analysis of Data-Driven Fault Detection and Diagnosis Methods for Process Systems", **Industrial and Engineering Chemistry Research**, v. 57, n. 32, p. 10719–10735, 2018. DOI: 10.1021/acs.iecr.8b00936.

ALCALA, C. F., JOE QIN, S. "Analysis and generalization of fault diagnosis methods for process monitoring", **Journal of Process Control**, v. 21, n. 3, p. 322–330, 2011. DOI: 10.1016/j.jprocont.2010.10.005. Disponível em: <http://dx.doi.org/10.1016/j.jprocont.2010.10.005>.

ALDRICH, C., AURET, L. **Advances in Computer Vision and Pattern Recognition Unsupervised Process Monitoring and Fault Diagnosis with Machine Learning Methods**. [S.l.: s.n.], 2013. Disponível em: <http://www.springer.com/series/4205>.

ALIPPI, C., BORACCHI, G., CARRERA, D., *et al.* "Change detection in multivariate datastreams: Likelihood and detectability loss", **IJCAI International Joint Conference on Artificial Intelligence**, v. 2016- Janua, p. 1368–1374, 2016.

ANACONDA INC. **The State of Data Science 2020 Moving from hype toward maturity**. 2020. Disponível em: [https://www.anaconda.com/state-of-data-science-2020?utm\\_medium=press&utm\\_source=anaconda&utm\\_campaign=sods-2020&utm\\_content=report](https://www.anaconda.com/state-of-data-science-2020?utm_medium=press&utm_source=anaconda&utm_campaign=sods-2020&utm_content=report). Acesso em: 18 ago. 2021.

ANP. **Projetos de PD&I**. 2023. Disponível em: <https://www.gov.br/anp/pt->

br/assuntos/pesquisa-desenvolvimento-e-inovacao/investimentos-em-pd-i/novo-projetos-de-pd-i. Acesso em: 1 fev. 2025.

ANZAI, T. K., FURTADO, P. H. T., DE BRITO, G. M., *et al.* "Catching Failures in 10 Minutes: An Approach to No Code, Fast Track, AI-Based Real Time Process Monitoring", **Offshore Technology Conference Brasil, OTCB 2023**, n. Cisco 2017, p. 1–11, 2023. DOI: 10.4043/32898-MS.

ANZAI, Thiago K., CARLOS COSTA DA SILVA PINTO, J. "Distinguishing Process Faults from Model Drift Through Variable Contribution Analysis: A Novel Perspective on Anomaly Diagnosis", **Processes**, v. 14, n. 5, p. 859, 7 mar. 2026. DOI: 10.3390/pr14050859. Disponível em: <https://www.mdpi.com/2227-9717/14/5/859>.

ANZAI, Thiago K., DE BRITO, G. M., LEMOS, T. S. M., *et al.* "Instance Selection Strategies to Improve the Performance of Data-Driven Regression Models Applied to Industrial Systems", **Processes**, v. 13, n. 7, p. 2187, 8 jul. 2025. DOI: 10.3390/pr13072187. Disponível em: <https://www.mdpi.com/2227-9717/13/7/2187>.

ARNAIZ-GONZÁLEZ, Á., DÍEZ-PASTOR, J. F., RODRÍGUEZ, J. J., *et al.* "Instance selection for regression: Adapting DROP", **Neurocomputing**, v. 201, p. 66–81, ago. 2016a. DOI: 10.1016/j.neucom.2016.04.003. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0925231216301953>.

ARNAIZ-GONZÁLEZ, Á., DÍEZ-PASTOR, J. F., RODRÍGUEZ, J. J., *et al.* "Instance selection for regression by discretization", **Expert Systems with Applications**, v. 54, p. 340–350, jul. 2016b. DOI: 10.1016/j.eswa.2015.12.046. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S095741741600049X>.

BARATEIRO, C., FARIA, A., FARIAS FILHO, J., *et al.* "Fiscal Measurement and Oil and Gas Production Market: Increasing Reliability Using Blockchain Technology", **Applied Sciences**, v. 12, n. 15, p. 7874, 5 ago. 2022. DOI: 10.3390/app12157874. Disponível em: <https://www.mdpi.com/2076-3417/12/15/7874>.

BAYRAM, F., AHMED, B. S., HALLIN, E. "Adaptive Data Quality Scoring Operations Framework using Drift-Aware Mechanism for Industrial Applications",

13 ago. 2024. Disponível em: <http://arxiv.org/abs/2408.06724>.

BAYRAM, F., AHMED, B. S., KASSLER, A. "From Concept Drift to Model Degradation: An Overview on Performance-Aware Drift Detectors", 21 mar. 2022. Disponível em: <http://arxiv.org/abs/2203.11070>.

BEKRAOUI, A., HADJADJ, A., BENMOUNAH, A., *et al.* "Uncertainty study of fiscal orifice meter used in a gas Algerian field", **Flow Measurement and Instrumentation**, v. 66, p. 200–208, abr. 2019. DOI: 10.1016/j.flowmeasinst.2019.01.020. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0955598618300554>.

BERGSTRA, J., YAMINS, D., COX, D. D. "Making a Science of Model Search", 23 set. 2012. Disponível em: <http://arxiv.org/abs/1209.5111>.

BERGSTRA, James, BARDENET, R., BENGIO, Y., *et al.* "Algorithms for Hyper-Parameter Optimization". 2011. **Anais [...]** Granada, Spain, [s.n.], 2011.

BIFET, A., GAVALDÀ, R. "Learning from time-changing data with adaptive windowing", **Proceedings of the 7th SIAM International Conference on Data Mining**, p. 443–448, 2007. DOI: 10.1137/1.9781611972771.42.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., *et al.* **Classification And Regression Trees**. [S.l.], Routledge, 2017. Disponível em: <https://www.taylorfrancis.com/books/9781351460491>.

CHANDRASHEKAR, G., SAHIN, F. "A survey on feature selection methods", **Computers & Electrical Engineering**, v. 40, n. 1, p. 16–28, jan. 2014. DOI: 10.1016/j.compeleceng.2013.11.024. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0045790613003066>.

CLAVIJO, N., MELO, A., CÂMARA, M. M., *et al.* "Development and application of a data-driven system for sensor fault diagnosis in an oil processing plant", **Processes**, v. 7, n. 7, 2019. DOI: 10.3390/pr7070436.

CLAVIJO, Nayher, MELO, A., SOARES, R. M., *et al.* "Variable Selection for Fault Detection Based on Causal Discovery Methods : Analysis of an Actual Industrial Case", p. 1–40, 2021.

CORTÉS-IBÁÑEZ, J. A., GONZÁLEZ, S., VALLE-ALONSO, J. J., *et al.*

"Preprocessing methodology for time series: An industrial world application case study", **Information Sciences**, v. 514, p. 385–401, 2020. DOI: 10.1016/j.ins.2019.11.027.

COX, D. R. "The Regression Analysis of Binary Sequences", **Journal of the Royal Statistical Society Series B: Statistical Methodology**, v. 20, n. 2, p. 215–232, 1 jul. 1958. DOI: 10.1111/j.2517-6161.1958.tb00292.x. Disponível em: <https://academic.oup.com/jrsssb/article/20/2/215/7027376>.

CROWDFLOWER. **Data Science Report**. . [S.l: s.n.], 2016. Disponível em: [https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower\\_DataScienceReport\\_2016.pdf](https://visit.figure-eight.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf).

CUI, G., LI, X., WEI, C., *et al.* "Data-Driven Fault Detection for Multimode Industrial Processes Based on Mixture of Autoencoders", **Asia-Pacific Journal of Chemical Engineering**, v. 20, n. 4, 23 jul. 2025. DOI: 10.1002/apj.70022. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/apj.70022>.

DOWNS, J. J., VOGEL, E. F. "A plant-wide industrial process control problem", **Computers and Chemical Engineering**, v. 17, n. 3, p. 245–255, 1993. DOI: 10.1016/0098-1354(93)80018-I.

EL MORR, C., JAMMAL, M., ALI-HASSAN, H., *et al.*, "Data Preprocessing". [S.l: s.n.], 2022. p. 117–163. DOI: 10.1007/978-3-031-16990-8\_4. Disponível em: [https://link.springer.com/10.1007/978-3-031-16990-8\\_4](https://link.springer.com/10.1007/978-3-031-16990-8_4).

EMAM, A. E. "GAS FLARING IN INDUSTRY: AN OVERVIEW", **Petroleum & Coal**, v. 57, n. 5, p. 532–555, 2015.

FEISA, T. T., GEBREMEDHEN, H. S., KIBRETE, F., *et al.* "Artificial Intelligence in Fault Diagnosis of Industrial Machinery: A Comprehensive Review", **Structural Control and Health Monitoring**, v. 2025, n. 1, 14 jan. 2025. DOI: 10.1155/stc/4640227. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1155/stc/4640227>.

GAMA, J., MEDAS, P., CASTILLO, G., *et al.*, "Learning with Drift Detection". [S.l: s.n.], 2004. p. 286–295. DOI: 10.1007/978-3-540-28645-5\_29. Disponível em: [http://link.springer.com/10.1007/978-3-540-28645-5\\_29](http://link.springer.com/10.1007/978-3-540-28645-5_29).

GAMA, J., ŽLIOBAITĖ, I., BIFET, A., *et al.* "A survey on concept drift adaptation", **ACM Computing Surveys**, v. 46, n. 4, p. 1–37, abr. 2014. DOI: 10.1145/2523813. Disponível em: <https://dl.acm.org/doi/10.1145/2523813>.

GARCIA, J., RIOS-COLQUE, L., PEÑA, A., *et al.* "Condition Monitoring and Predictive Maintenance in Industrial Equipment: An NLP-Assisted Review of Signal Processing, Hybrid Models, and Implementation Challenges", **Applied Sciences**, v. 15, n. 10, p. 5465, 13 maio 2025. DOI: 10.3390/app15105465. Disponível em: <https://www.mdpi.com/2076-3417/15/10/5465>.

GARCÍA, S., DERRAC, J., CANO, J. R., *et al.* "Prototype selection for nearest neighbor classification: Taxonomy and empirical study", **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 34, n. 3, p. 417–435, 2012. DOI: 10.1109/TPAMI.2011.142.

GEMAQUE, R. N., COSTA, A. F. J., GIUSTI, R., *et al.* "An overview of unsupervised drift detection methods", **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 10, n. 6, p. 1–18, 2020. DOI: 10.1002/widm.1381.

GOLENDUKHINA, V., FELDERER, M. "Unveiling Data Preprocessing Patterns in Computational Notebooks". 28 ago. 2024. **Anais [...]** [S.l.], IEEE, 28 ago. 2024. p. 114–121. DOI: 10.1109/SEAA64295.2024.00025. Disponível em: <https://ieeexplore.ieee.org/document/10803332/>.

GONÇALVES, P. M., DE CARVALHO SANTOS, S. G. T., BARROS, R. S. M., *et al.* "A comparative study on concept drift detectors", **Expert Systems with Applications**, v. 41, n. 18, p. 8144–8156, 2014. DOI: 10.1016/j.eswa.2014.07.019.

GOOGLE LLC. **Google Scholar**. 2024. Disponível em: [https://scholar.google.com/scholar?hl=pt-BR&as\\_sdt=0%2C5&q=%22Instance+selection%22%2B%22Process+monitoring%22&btnG=](https://scholar.google.com/scholar?hl=pt-BR&as_sdt=0%2C5&q=%22Instance+selection%22%2B%22Process+monitoring%22&btnG=). Acesso em: 1 mar. 2024.

GUIDOTTI, R. "Counterfactual explanations and how to find them: literature review and benchmarking", **Data Mining and Knowledge Discovery**, v. 38, n. 5, p. 2770–2824, 28 set. 2024. DOI: 10.1007/s10618-022-00831-6. Disponível em:

<https://link.springer.com/10.1007/s10618-022-00831-6>.

GUILLEN, A., HERRERA, L. J., RUBIO, G., *et al.* "New method for instance or prototype selection using mutual information in time series prediction", **Neurocomputing**, v. 73, n. 10–12, p. 2030–2038, 2010. DOI: 10.1016/j.neucom.2009.11.031. Disponible em: <http://dx.doi.org/10.1016/j.neucom.2009.11.031>.

GUYON, I., ELISSEEFF, A. "An Introduction to Variable and Feature Selection", **Journal of Machine Learning Research**, v. 3, p. 1157–1182, 2003. Disponible em: <http://www.ai.mit.edu/projects/jmlr/papers/volume3/guyon03a/source/old/guyon03a.pdf>.

HALLGRÍMSSON, Á. D., NIEMANN, H. H., LIND, M. "Improved process diagnosis using fault contribution plots from sparse autoencoders", **IFAC-PapersOnLine**, v. 53, n. 2, p. 730–737, 2020a. DOI: 10.1016/j.ifacol.2020.12.823. Disponible em: <https://doi.org/10.1016/j.ifacol.2020.12.823>.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. **The Elements of Statistical Learning**. New York, NY, Springer New York, 2009. Disponible em: <http://link.springer.com/10.1007/978-0-387-84858-7>. (Springer Series in Statistics).

HINDER, F., VAQUET, V., HAMMER, B. "One or two things we know about concept drift—a survey on monitoring in evolving environments. Part A: detecting concept drift", **Frontiers in Artificial Intelligence**, v. 7, 19 jun. 2024. DOI: 10.3389/frai.2024.1330257. Disponible em: <https://www.frontiersin.org/articles/10.3389/frai.2024.1330257/full>.

IOGP. **Guidelines for design and operations to minimize and avoid flaring**. [S.l.: s.n.], 2024. Disponible em: <https://statics.teams.cdn.office.net/evergreen-assets/safelinks/1/atp-safelinks.html>.

JAMES, G., WITTEN, D., HASTIE, T., *et al.* **An Introduction to Statistical Learning**. New York, NY, Springer New York, 2013. v. 103. Disponible em: <http://link.springer.com/10.1007/978-1-4614-7138-7>. (Springer Texts in Statistics).

JOURDAN, N., BAYER, T., BIEGEL, T., *et al.* "Handling concept drift in

deep learning applications for process monitoring", **Procedia CIRP**, v. 120, p. 33–38, 2023. DOI: 10.1016/j.procir.2023.08.007. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S2212827123006789>.

KAZEMI, P., MASOUMIAN, A., MARTIN, P. "Fault Detection and Isolation for Time-Varying Processes Using Neural-Based Principal Component Analysis", **Processes**, v. 12, n. 6, p. 1218, 14 jun. 2024. DOI: 10.3390/pr12061218. Disponível em: <https://www.mdpi.com/2227-9717/12/6/1218>.

KORDOS, M., BLACHNIK, M., SCHERER, R. "Fuzzy clustering decomposition of genetic algorithm-based instance selection for regression problems", **Information Sciences**, v. 587, p. 23–40, 2022. DOI: 10.1016/j.ins.2021.12.016.

KU, W., STORER, R. H., GEORGAKIS, C. "Disturbance detection and isolation by dynamic principal component analysis", **Chemometrics and Intelligent Laboratory Systems**, v. 30, n. 1, p. 179–196, 1995. DOI: 10.1016/0169-7439(95)00076-3.

LEITE, D., ANDRADE, E., RATIVA, D., *et al.* "Fault Detection and Diagnosis in Industry 4.0: A Review on Challenges and Opportunities", **Sensors**, v. 25, n. 1, p. 60, 25 dez. 2024. DOI: 10.3390/s25010060. Disponível em: <https://www.mdpi.com/1424-8220/25/1/60>.

LEMOS, T., CAMPOS, L. F., MELO, A., *et al.* "Echo State network based soft sensor for Monitoring and Fault Detection of Industrial Processes", **Computers and Chemical Engineering**, v. 155, p. 107512, 2021. DOI: 10.1016/j.compchemeng.2021.107512. Disponível em: <https://doi.org/10.1016/j.compchemeng.2021.107512>.

LI, C., MAO, Z. "A label noise filtering method for regression based on adaptive threshold and noise score", **Expert Systems with Applications**, v. 228, n. May, p. 120422, 2023. DOI: 10.1016/j.eswa.2023.120422. Disponível em: <https://doi.org/10.1016/j.eswa.2023.120422>.

LIU, H., MOTODA, H. "On issues of instance selection", **Data Mining and Knowledge Discovery**, v. 6, n. 2, p. 115–130, 2002. DOI:

10.1023/A:1014056429969.

LIU, N., HU, M., WANG, J., *et al.* "Fault detection and diagnosis using Bayesian network model combining mechanism correlation analysis and process data: Application to unmonitored root cause variables type faults", **Process Safety and Environmental Protection**, v. 164, p. 15–29, ago. 2022. DOI: 10.1016/j.psep.2022.05.073. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0957582022004761>.

LOBO, J. L., LAÑA, I., OSABA, E., *et al.* "On the Connection between Concept Drift and Uncertainty in Industrial Artificial Intelligence", 14 mar. 2023. Disponível em: <http://arxiv.org/abs/2303.07940>.

LUGHOFER, E., "Robust Data-Driven Fault Detection in Dynamic Process Environments Using Discrete Event Systems". **Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems**, Cham, Springer International Publishing, 2018. p. 73–116. DOI: 10.1007/978-3-319-74962-4\_4. Disponível em: [http://link.springer.com/10.1007/978-3-319-74962-4\\_4](http://link.springer.com/10.1007/978-3-319-74962-4_4).

LUO, L., XIE, L., SU, H. "Deep Learning With Tensor Factorization Layers for Sequential Fault Diagnosis and Industrial Process Monitoring", **IEEE Access**, v. 8, p. 105494–105506, 2020. DOI: 10.1109/ACCESS.2020.3000004. Disponível em: <https://ieeexplore.ieee.org/document/9108248/>.

MD NOR, N., CHE HASSAN, C. R., HUSSAIN, M. A. "A review of data-driven fault detection and diagnosis methods: Applications in chemical process systems", **Reviews in Chemical Engineering**, v. 36, n. 4, p. 513–553, 2020. DOI: 10.1515/revce-2017-0069.

MELO, A., CÂMARA, M. M., CLAVIJO, N., *et al.* "Open benchmarks for assessment of process monitoring and fault diagnosis techniques: A review and critical analysis", **Computers & Chemical Engineering**, v. 165, p. 107964, set. 2022. DOI: 10.1016/j.compchemeng.2022.107964. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0098135422003003>.

MELO, A., CÂMARA, M. M., PINTO, J. C. "Data-Driven Process Monitoring and Fault Diagnosis: A Comprehensive Survey", **Processes**, v. 12, n. 2,

p. 251, 24 jan. 2024b. DOI: 10.3390/pr12020251. Disponível em: <https://www.mdpi.com/2227-9717/12/2/251>.

MELO, A., LEMOS, T. S. M., SOARES, R. M., *et al.* "BibMon: An open source Python package for process monitoring, soft sensing, and fault diagnosis", **Digital Chemical Engineering**, v. 13, n. August, 2024. DOI: 10.1016/j.dche.2024.100182.

MILLER, P., SWANSON, R. E., HECKLER, C. E. "Contribution plots: a missing link in multivariate quality control", **Appl. Math. and Comp. Sci.**, v. 8, p. 775–792, 1998.

MONTIEL, J., HALFORD, M., MASTELINI, S. M., *et al.* "River: machine learning for streaming data in Python", 8 dez. 2020. Disponível em: <http://arxiv.org/abs/2012.04740>.

OLIVEIRA-JUNIOR, J. M., DE ARRUDA PEREIRA, M. "Forecasting Total Oil and Grease in produced water using Machine Learning methods in an oil extraction plant", **Marine Systems and Ocean Technology**, v. 15, n. 2, p. 124–134, 2020. DOI: 10.1007/s40868-020-00075-3. Disponível em: <https://doi.org/10.1007/s40868-020-00075-3>.

PAPER, D. "Scikit-Learn Classifier Tuning from Complex Training Sets", **Hands-on Scikit-Learn for Machine Learning Applications**, v. 12, p. 165–188, 2020. DOI: 10.1007/978-1-4842-5373-1\_6.

PARK, Y. J., FAN, S. K. S., HSU, C. Y. "A review on fault detection and process diagnostics in industrial processes", **Processes**, v. 8, n. 9, 2020. DOI: 10.3390/PR8091123.

PEDREGOSA, F., WEISS, R., BRUCHER, M., *et al.* "Scikit-learn: Machine Learning in Python", **Journal of Machine Learning Research**, v. 12, n. February 2011, p. 2825–2830, 2011. DOI: 10.5555/1953048.2078195. Disponível em: <http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html><http://arxiv.org/abs/1201.0490>.

PESARANGHADER, A., VIKTOR, H., PAQUET, E. "McDiarmid Drift Detection Methods for Evolving Data Streams", 5 out. 2017. Disponível em:

<http://arxiv.org/abs/1710.02030>.

PHAM, T. M. T., PREMKUMAR, K., NAILI, M., *et al.* "Time to Retrain? Detecting Concept Drifts in Machine Learning Systems", 3 ago. 2025. Disponível em: <http://arxiv.org/abs/2410.09190>.

PORTNOY, I., MELENDEZ, K., PINZON, H., *et al.* "An improved weighted recursive PCA algorithm for adaptive fault detection", **Control Engineering Practice**, v. 50, p. 69–83, 2016. DOI: 10.1016/j.conengprac.2016.02.010. Disponível em: <http://dx.doi.org/10.1016/j.conengprac.2016.02.010>.

RAMÍREZ, C., SILVA, J. F., TAMSSAOUET, F., *et al.* "Fault detection and monitoring using a data-driven information-based strategy: Method, theory, and application", **Mechanical Systems and Signal Processing**, v. 228, 2025. DOI: 10.1016/j.ymsp.2025.112403.

REINARTZ, C., KULAHCI, M., RAVN, O. "An extended Tennessee Eastman simulation dataset for fault-detection and decision support systems", **Computers and Chemical Engineering**, v. 149, p. 107281, 2021. DOI: 10.1016/j.compchemeng.2021.107281. Disponível em: <https://doi.org/10.1016/j.compchemeng.2021.107281>.

RIETH, C. A., AMSEL, B. D., TRAN, R., *et al.* **Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation**. 2017. Disponível em: <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6C3JR> 1. Acesso em: 7 jul. 2022.

RODRIGUES, A. C. C. "Decreasing natural gas flaring in Brazilian oil and gas industry", **Resources Policy**, v. 77, p. 102776, ago. 2022. DOI: 10.1016/j.resourpol.2022.102776. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0301420722002240>.

ROSS, G. J., ADAMS, N. M., TASOULIS, D. K., *et al.* "Exponentially Weighted Moving Average Charts for Detecting Concept Drift", 25 dez. 2012. DOI: 10.1016/j.patrec.2011.08.019. Disponível em: <http://arxiv.org/abs/1212.6018>.

RUSSELL, E. L., LEO, H. C., BRAATZ, R. D. **Data-Driven Techniques for**

**Fault Detection and Diagnosis in chemical Process.** [S.l.], Springer, 2000.

SCHLEGL, T., TOMASELLI, D., SCHLEGL, S., *et al.* "Automated search of process control limits for fault detection in time series data", **Journal of Process Control**, v. 117, p. 52–64, set. 2022. DOI: 10.1016/j.jprocont.2022.07.002. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S095915242200124X>.

SIRCAR, A., YADAV, K., RAYAVARAPU, K., *et al.* "Application of machine learning and artificial intelligence in oil and gas industry", **Petroleum Research**, v. 6, n. 4, p. 379–391, 2021. DOI: 10.1016/j.ptlrs.2021.05.009. Disponível em: <https://doi.org/10.1016/j.ptlrs.2021.05.009>.

SONG, Y., LIANG, J., LU, J., *et al.* "An efficient instance selection algorithm for k nearest neighbor regression", **Neurocomputing**, v. 251, p. 26–34, ago. 2017. DOI: 10.1016/j.neucom.2017.04.018. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0925231217306884>.

SPINA, D. E., DE O. CAMPOS, L. F., DE ARRUDA, W. F., *et al.* "Comparison of autoencoder architectures for fault detection in industrial processes", **Digital Chemical Engineering**, v. 12, p. 100162, set. 2024. DOI: 10.1016/j.dche.2024.100162. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S2772508124000243>.

STOJANOVIĆ, M. B., BOŽIĆ, M. M., STANKOVIĆ, M. M., *et al.* "A methodology for training set instance selection using mutual information in time series prediction", **Neurocomputing**, v. 141, p. 236–245, out. 2014. DOI: 10.1016/j.neucom.2014.03.006. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S092523121400410X>.

TIDRIRI, K., CHATTI, N., VERRON, S., *et al.* "Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges", **Annual Reviews in Control**, v. 42, p. 63–81, 2016. DOI: 10.1016/j.arcontrol.2016.09.008. Disponível em: <http://dx.doi.org/10.1016/j.arcontrol.2016.09.008>.

TRUONG, C., OUDRE, L., VAYATIS, N. "Selective review of offline change point detection methods", **Signal Processing**, v. 167, p. 107299, 2020. DOI:

10.1016/j.sigpro.2019.107299. Disponível em:  
<https://doi.org/10.1016/j.sigpro.2019.107299>.

VENKATASUBRAMANIAN, V., RENGASWAMY, R., KAVURI, S. N. "A review of process fault detection and diagnosis", **Computers & Chemical Engineering**, v. 27, n. 3, p. 313–326, 2003. DOI: 10.1016/s0098-1354(02)00161-8.

VENKATASUBRAMANIAN, V., RENGASWAMY, R., KAVURI, S. N., *et al.* "A review of process fault detection and diagnosis", **Computers & Chemical Engineering**, v. 27, n. 3, p. 327–346, 2003. DOI: 10.1016/s0098-1354(02)00162-x.

VENKATASUBRAMANIAN, V., RENGASWAMY, R., YIN, K., *et al.* "A review of fault detection and diagnosis. Part III: Process history based methods", **Computers and Chemical Engineering**, v. 27, p. 327–346, 2003. .

VERMA, S., BOONSANONG, V., HOANG, M., *et al.* "Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review", 15 nov. 2022. Disponível em: <http://arxiv.org/abs/2010.10596>.

WACHTER, S., MITTELSTADT, B., RUSSELL, C. "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR", 21 mar. 2018. Disponível em: <http://arxiv.org/abs/1711.00399>.

WIKIPEDIA, F. **Nelson rules**. 2015. Disponível em:  
[https://en.wikipedia.org/wiki/Nelson\\_rules#/media/File:Rule\\_8\\_-\\_Control\\_Charts\\_for\\_Nelson\\_Rules.svg](https://en.wikipedia.org/wiki/Nelson_rules#/media/File:Rule_8_-_Control_Charts_for_Nelson_Rules.svg). Acesso em: 16 jan. 2026.

XAVIER, G. M., DE SEIXAS, J. M. "Fault Detection and Diagnosis in a Chemical Process using Long Short-Term Memory Recurrent Neural Network". jul. 2018. **Anais [...]** [S.l.], IEEE, jul. 2018. p. 1–8. DOI: 10.1109/IJCNN.2018.8489385. Disponível em: <https://ieeexplore.ieee.org/document/8489385/>.

XIAO, Y., WANG, H., XU, W., *et al.* "L1 norm based KPCA for novelty detection", **Pattern Recognition**, v. 46, n. 1, p. 389–396, jan. 2013. DOI: 10.1016/j.patcog.2012.06.017. Disponível em:  
<https://linkinghub.elsevier.com/retrieve/pii/S0031320312002877>.

YAN, M. M. W. "Accurate detecting concept drift in evolving data streams", **ICT Express**, v. 6, n. 4, p. 332–338, 2020. DOI: 10.1016/j.ict.2020.05.011.

Disponível em: <https://doi.org/10.1016/j.ict.2020.05.011>.

YIN, M. "Drift-Aware Streaming Predictive Maintenance for Semiconductor Equipment", **Applied and Computational Engineering**, v. 197, n. 1, p. 180–188, 28 out. 2025. DOI: 10.54254/2755-2721/2025.28630. Disponível em: <https://www.ewadirect.com/proceedings/ace/article/view/28630>.

YIN, S., DING, S. X., HAGHANI, A., *et al.* "A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process", **Journal of Process Control**, v. 22, n. 9, p. 1567–1581, 2012. DOI: 10.1016/j.jprocont.2012.06.009. Disponível em: <http://dx.doi.org/10.1016/j.jprocont.2012.06.009>.

ZACHOS, I. "Bayesian On-line Change-point Detection Spatio-temporal point processes Ioannis Zachos", n. April, 2018.

ZARDOYA, A. R., LUCENA, I. L., BENGOETXEA, I. O., *et al.* "Research on an internal combustion engine with an injected pre-chamber to operate with low methane number fuels for future gas flaring reduction", **Energy**, v. 253, p. 124096, ago. 2022. DOI: 10.1016/j.energy.2022.124096. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0360544222009999>.

ZHAO, J., GAL, A., KRISHNAN, S. "A System for Quantifying Data Science Workflows with Fine-Grained Procedural Logging and a Pilot Study", 28 maio 2024. Disponível em: <http://arxiv.org/abs/2405.17845>.

ZHU, J., JIANG, M., LIU, Z. "Fault detection and diagnosis in industrial processes with variational autoencoder: A comprehensive study", **Sensors**, v. 22, n. 1, 2022. DOI: 10.3390/s22010227.

# Appendix A

## Modeling Techniques for Process Monitoring

This appendix is provided for completeness and clarity. The techniques described herein are included to contextualize their specific role within the proposed monitoring framework. The objective is not to present exhaustive theoretical developments, which are widely available in the literature, but to clarify how each technique is employed as a reference model for process monitoring and how it relates to the monitoring statistics and diagnostic procedures discussed in the main chapters.

Throughout this appendix, the notation and terminology are consistent with those used in the thesis. In particular,  $\mathbf{X}$  denotes the data matrix of measured variables,  $\mathbf{y}$  the measured output vector,  $\hat{\mathbf{y}}$  or  $\hat{\mathbf{X}}$  the predicted output (depending on whether the problem is supervised or unsupervised), and  $\lambda$  the control limit associated with the Squared Prediction Error (SPE).

### A.1 Squared Prediction Error (SPE) as a unified monitoring statistic

The Squared Prediction Error (SPE) is the primary monitoring statistic adopted throughout this thesis and constitutes the common link among the different modeling techniques employed. In general terms, the SPE quantifies the discrepancy between an observed vector and its model prediction.

An alarm is triggered whenever  $\text{SPE} > \lambda$ , where  $\lambda$  is a control limit estimated from data representing normal operation. Although the formulation of  $\hat{\mathbf{X}}$  varies depending on the underlying model, the SPE provides a model-agnostic basis for fault detection and diagnosis.

This unified formulation enables the drift-aware diagnostic analysis introduced in Chapter 4, independently of the specific learning algorithm used to generate predictions or reconstructions.

## A.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is one of the most established techniques in multivariate statistical process control and is used in this thesis as a baseline unsupervised model for monitoring normal operation, particularly in the instance selection and detection stages discussed in Chapters 3 and 4.

Given a mean-centered data matrix  $\mathbf{X} \in \mathbb{R}^{N \times M}$ , PCA decomposes the data as:

$$\mathbf{X} = \mathbf{T} \mathbf{P}^T + \mathbf{E} \quad \text{Eq. A.1}$$

where  $\mathbf{T}$  contains the principal component scores,  $\mathbf{P}$  is the loading matrix, and  $\mathbf{E}$  represents the residuals. Retaining the first  $a < M$  components yields the reconstructed matrix:

$$\mathbf{X} = \mathbf{T}_a \mathbf{P}_a^T \quad \text{Eq. A.2}$$

In the context of monitoring, PCA is used as a model of normal behavior rather than as a dimensionality reduction tool. The SPE is computed from the residual vector  $\mathbf{e}$  as:

$$\text{SPE} = \|\mathbf{e}\|^2 \quad \text{Eq. A.3}$$

A control limit  $\lambda$  is then estimated from the distribution of SPE values during the training (normal) period.

In this thesis, the monitoring methodology based on PCA follows the procedures established in MELO, CÂMARA, et al., (2024b) and ALDRICH, AURET, (2013). These references provide the methodological basis for PCA-based monitoring, including preprocessing, model construction, residual-space analysis, and decision rules.

### A.3 Autoencoders (AE)

Autoencoders (AE) are neural network models designed to learn nonlinear representations of data through an encoder-decoder architecture. In this thesis, AEs are employed as unsupervised models of normal operation, extending the reconstruction-based monitoring paradigm associated with PCA and supporting the monitoring experiments discussed in Chapter 3.

Let  $\mathbf{X} \in \mathbb{R}^{N \times M}$  denote the dataset of  $N$  observations and  $M$  measured variables. At each time step  $t$ , the autoencoder receives an input vector  $\mathbf{X}(t) \in \mathbb{R}^M$ , corresponding to one observation of the process. The encoder function  $f(\cdot)$  maps  $\mathbf{X}(t)$  to a latent representation  $\mathbf{z}(t)$ , which is subsequently decoded by  $g(\cdot)$  to produce:

$$\hat{\mathbf{x}}(t) = g(f(\mathbf{X}(t))) \quad \text{Eq. A.4}$$

The reconstruction residual is defined as:

$$\mathbf{e}(t) = \mathbf{X}(t) - \hat{\mathbf{X}}(t) \quad \text{Eq. A.5}$$

and the corresponding SPE is computed as:

$$\text{SPE}(t) = \|\mathbf{e}(t)\|^2 \quad \text{Eq. A.6}$$

AEs are particularly suitable for monitoring processes with nonlinear or multimodal behavior and have been widely adopted in industrial fault detection due to their ability to capture complex relationships among variables.

In this thesis, the AE-based monitoring methodology follows the procedures and recommendations presented by HALLGRÍMSSON, NIEMANN, et al., (2020b), ZHU, JIANG, et al., (2022), and SPINA, DE O. CAMPOS, et al., (2024).

### A.4 Echo State Networks (ESN)

Echo State Networks (ESN) belong to the class of reservoir computing models and are used in this thesis for supervised monitoring and soft-sensing applications involving dynamic processes, as discussed in Chapter 3.

Let  $\mathbf{X} \in \mathbb{R}^{N \times M}$  denote the input matrix and  $\mathbf{y}$  the measured output vector. The reservoir state  $\mathbf{r}(t)$  evolves according to:

$$\mathbf{r}(t) = \varphi(\mathbf{W}_{\text{in}}\mathbf{x}(t) + \mathbf{W}\mathbf{r}(t-1)) \quad \text{Eq. A.7}$$

where  $\varphi(\cdot)$  is a nonlinear activation function,  $\mathbf{W}_{\text{in}}$  is the input weight matrix, and  $\mathbf{W}$  is the recurrent weight matrix. The predicted output is given by:

$$\hat{\mathbf{y}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t) \quad \text{Eq. A.8}$$

with  $\mathbf{W}_{\text{out}}$  estimated through linear regression during training.

For monitoring purposes, the residual is defined as:

$$\mathbf{e}(t) = \mathbf{y}(t) - \hat{\mathbf{y}}(t) \quad \text{Eq. A.9}$$

and the SPE statistic is computed from the residual sequence and compared against a control limit  $\lambda$ .

In this thesis, the ESN-based monitoring methodology follows the procedures and recommendations presented by LEMOS, CAMPOS, et al., (2021). Their work provides practical guidance on ESN configuration for industrial applications, including reservoir design, hyperparameter selection, and the use of residual-based indicators for fault detection.

## A.5 Regression Trees (RT)

Regression Trees (RT) are nonparametric supervised learning models that partition the input space into regions with approximately constant output values. In this thesis, RTs are not used as final monitoring models, but as auxiliary tools within the Instance Selection Library (ISLib) described in Chapter 3.

Their role is to provide fast and interpretable estimates of generalization performance across different operational regions. For a dataset of size  $N$ , the Mean Squared Error (MSE) is defined as:

$$\text{MSE} = \left(\frac{1}{N}\right) \sum_{t=1}^N (\mathbf{y}(t) - \hat{\mathbf{y}}(t))^2 \quad \text{Eq. A.10}$$

The simplicity and low computational cost of RTs make them suitable for repeated evaluations during the batch instance selection stage, without imposing significant overhead on the monitoring workflow.

# Appendix B

## Instance Selection Library (ISLib) for Python

ISLib (Instance Selection Library) is a Python library designed to assist in determining the optimal periods for training regression models. It combines unsupervised clustering analysis with sliding window regression to identify the most representative data segments and the ideal training window size. The ISLib library is available to the scientific community on GitHub at <https://github.com/tkanzai/ISLib>.

The library provides three main analysis modes:

- **Cluster Analysis:** Identifies distinct operating regions using K-Means and evaluates each cluster's predictive capability.
- **Sliding Window Analysis:** Determines the minimum training window size that achieves the best prediction performance.
- **Full Analysis:** Combines both analyses into an end-to-end pipeline with automatic markdown report generation.

The following sections detail the main components of the ISLib, describing its internal structure, processing stages, and the functionalities that support the analyses presented in Chapter 3.

### B.1 Data Preprocessing

#### B.1.1 `preprocess_data()`

Applies preprocessing steps configured in `preprocess_list`. The original DataFrame is not modified and a copy is returned.

```

def preprocess_data(self, df: pd.DataFrame) -> pd.DataFrame:
    """Preprocess the DataFrame data.
    Args:
        df: DataFrame to preprocess.
    Returns:
        DataFrame with preprocessed data.
    Raises:
        ValueError: If the DataFrame is empty.
        TypeError: If the input is not a DataFrame.
    """

```

Processing steps (controlled by `preprocess_list`):

- 'transform' – Converts all columns to numeric (coercing errors to NaN) and fills missing values using forward-fill then back-fill.
- 'remove\_zero\_std' – Removes columns whose standard deviation is zero.

## B.2 Cluster Analysis

The cluster analysis pipeline identifies distinct operating regions in the data using K-Means, then evaluates how well each region can predict the others.

### B.2.1 `kmeans_optimize_clusters()`

Determines the optimal number of clusters using a modified elbow method. The objective function combines inertia with a penalty term proportional to the number of clusters, normalized by the initial inertia magnitude.

```

def kmeans_optimize_clusters(self, df: pd.DataFrame) -> pd.DataFrame:
    """Optimize the number of clusters in K-Means using the elbow method.
    Args:
        df: Data to be clustered.
    Returns:
        DataFrame with an additional 'Clusters' column.
    Raises:
        ValueError: If parameters are invalid or data is insufficient.
        TypeError: If the input is not a DataFrame.
    """

```

## B.2.2 cluster\_regressor()

Evaluates each cluster by training a model on its data and testing on all remaining clusters. Returns clusters sorted by ascending weighted error score.

```
def cluster_regressor(
    self, df: pd.DataFrame, target: Optional[str] = None
) -> List[Tuple[int, float]]:
    """Perform per-cluster regression and return clusters sorted by score.
    Args:
        df: DataFrame containing the data.
        target: Target variable name, or None for PCA.
    Returns:
        List of tuples (cluster_id, score) sorted ascending.
    Raises:
        ValueError: If target is not in DataFrame columns.
    """
```

## B.2.3 Cluster Evaluation (PCA & Regression)

Two evaluation strategies are available:

- `_evaluate_cluster_pca()`: When target is None, the cluster is evaluated using PCA reconstruction error. A PCA model is fitted on the cluster data, then the test data (all other clusters) is projected and reconstructed. The MSE between original and reconstructed data, weighted by the log-frequency of the cluster, gives the final score.
- `_evaluate_cluster_regression()`: When a target column is specified, a `DecisionTreeRegressor` is trained on the cluster data and tested on the remaining clusters. The score combines MSE, Pearson correlation, and log-frequency.

## B.3 Sliding Window Analysis

The sliding window analysis evaluates how the prediction error varies as the training window grows, identifying the point where adding more data no longer improves (or starts degrading) model performance.

### B.3.1 sliding\_window\_regression()

```
def sliding_window_regression(
    self, df: pd.DataFrame, target: Optional[str] = None
) -> Tuple[List[float], int, np.ndarray]:
    """Perform sliding window regression analysis.
    For each window size (from min_window up to n_rows, with step
    determined by resolution), trains a model on the initial data
    and computes the prediction error on the remaining data.
    When target is None, uses PCA reconstruction error;
    otherwise uses DecisionTreeRegressor.
    Args:
        df: Input DataFrame (without 'Clusters' column).
        target: Target column name, or None for PCA.
    Returns:
        Tuple: (test_errors, min_error_idx, spike_idx)
    Raises:
        ValueError: If target not found, min_window <= 0,
        insufficient rows, or resolution <= 1.
    """
```

## B.4 High-Level Analysis Methods

Three convenience methods combine the low-level building blocks into complete analysis workflows.

### B.4.1 cluster\_analysis()

Runs preprocessing, cluster optimization, per-cluster regression, plotting, and report generation. Does not include sliding window analysis.

```
def cluster_analysis(
    self, df: pd.DataFrame, target: Optional[str] = None
) -> Tuple[pd.DataFrame, Optional[List], Optional[pd.DataFrame], Optional[str]]:
    """Perform cluster analysis only.
    Returns:
        Tuple: (df_processed, sorted_clusters, df_results, results_markdown)
    """
```

Usage example:

```
islib = InstanceSelectionLib(show_figures=False)
df_proc, clusters, df_dates, report = islib.cluster_analysis(df)
```

## B.4.2 window\_analysis()

Runs sliding window regression on a (possibly pre-filtered) DataFrame. Automatically removes the Clusters column if present.

```
def window_analysis(
    self, df: pd.DataFrame, target: Optional[str] = None
) -> Tuple[Optional[List], Optional[int], Optional[np.ndarray],
          Optional[pd.DataFrame], Optional[str]]:
    """Perform sliding window analysis only.
    Returns:
        Tuple: (test_errors, min_error_idx, spike_idx,
               df_optimized, results_markdown)
    """
```

Usage example:

```
# First run cluster analysis
df_proc, clusters, _, _ = islib.cluster_analysis(df)
# Then run window analysis on the processed data
errors, min_idx, spikes, df_opt, report = islib.window_analysis(df_proc)
```

## B.4.3 full\_analysis()

Executes the complete pipeline: preprocessing, clustering, per-cluster regression, sliding window regression, plotting, and report generation. The pipeline steps are controlled by pipeline\_list.

```
def full_analysis(
    self, df: pd.DataFrame, target: Optional[str] = None
) -> Tuple[pd.DataFrame, Optional[List], Optional[List], Optional[int],
          Optional[pd.DataFrame], Optional[pd.DataFrame], Optional[str]]:
    """Perform a complete data analysis.
    Returns:
        Tuple: (df_processed, sorted_clusters, test_errors,
               min_error_idx, df_results, df_optimized, results_markdown)
    """
```

# Appendix C

## Nearest Normal Value (NNV) for fault diagnosis

The module Nearest Normal Value (NNV) is a fault diagnosis, model-agnostic library. The diagnostic algorithm is decoupled from any particular monitoring model through an abstract interface, enabling its application to PCA, autoencoders, or any other statistical model that exposes a monitoring statistic and a corresponding control limit.

The module comprises two classes and one utility function:

- `ProcessModel`: An abstract base class that defines the contract any monitoring model must satisfy.
- `NNVFaultDiagnosis`: The core class that implements the NNV contribution algorithm, adaptive reference updating, and the complete analysis pipeline.
- `create_data_stream`: A helper function that extracts a single observation from a dataset, preserving the original column names and index.

The sections below present the core components of the NNV diagnostic framework, outlining its architectural structure, class interfaces, and operational workflow as used in the methodology introduced in Chapter 4.

### C.1 ProcessModel

The `ProcessModel` class defines the minimal contract that any monitoring model must fulfil to be used with the NNV diagnosis system.

```

class ProcessModel(ABC):
    """
    Abstract base class for process monitoring models.
    This defines the interface that monitoring models must implement.
    """
    @abstractmethod
    def predict(self, data: pd.DataFrame, **kwargs) -> None:
        """
        Predict using the trained model.
        Args:
            data: Input data for prediction
            **kwargs: Additional arguments for prediction
        """
        pass
    @property
    @abstractmethod
    def monitoring_statistic(self) -> float:
        """Return the current monitoring statistic value."""
        pass
    @property
    @abstractmethod
    def control_limit(self) -> float:
        """Return the control limit for the monitoring statistic."""
        pass

```

The interface prescribes three capabilities:

- `predict(data, **kwargs)`: Receives a single-observation `DataFrame` and computes the model's internal state (residuals, scores, etc.).
- `monitoring_statistic`: A read-only property that returns a scalar value representing the current monitoring statistic (for instance, SPE). This value is expected to reflect the most recent call to `predict`.
- `control_limit`: A read-only property that returns the statistical control limit against which the monitoring statistic is compared. Observations whose statistic exceeds this threshold are considered anomalous.

This three-member interface is intentionally minimal. Any monitoring model — whether PCA-based, autoencoder-based, or otherwise — can be adapted to this contract.

## C.2 Class Initialisation and Private Helpers

The `NNVFaultDiagnosis` class is the central element of the module. Its constructor performs interface validation and stores the model reference together with an optional preprocessing identifier.

```
class NNVFaultDiagnosis:
    """
    Nearest Normal Value (NNV) Fault Diagnosis Class
    This class implements the NNV technique for fault diagnosis in
    industrial processes. The method calculates contribution values
    for each process variable by replacing each variable with its
    reference (normal) value and measuring the impact on the
    monitoring statistic.
    The NNV technique helps identify root causes of process faults
    by quantifying how much each variable contributes to the overall
    process deviation.
    """
    def __init__(self, model: ProcessModel,
                 preprocessing_function: Optional[str] = None):
        """
        Initialize the NNV fault diagnosis system.
        Args:
            model: A trained process monitoring model that implements
                the ProcessModel interface (or an adapter).
            preprocessing_function: Preprocessing function identifier
                passed to the model's predict method.
        Raises:
            TypeError: If model doesn't implement the required interface.
        """
        if not hasattr(model, 'predict'):
            raise TypeError("Model must implement 'predict' method")
        has_monitoring_stat = (
            hasattr(model, 'monitoring_statistic')
            or hasattr(model, 'SPE_teste')
            or (hasattr(model, 'bibmon_model')
                and hasattr(model.bibmon_model, 'limSPE'))
        )
        if not has_monitoring_stat:
            raise TypeError(
                "Model must have monitoring statistic capability")
        self.model = model
        self.preprocessing_function = preprocessing_function
        self._reference_values = None
```

### C.3 Core NNV Contribution Algorithm

The `calculate_contributions` method implements the core NNV algorithm. The rationale is that if replacing a value from input vector  $\mathbf{X}$  by its reference causes a large reduction in the monitoring statistic, then this input is a major contributor to the anomaly. Conversely, if the replacement has little effect, the variable is operating close to its normal condition and contributes minimally to the fault.

```
def calculate_contributions(self, current_data: pd.DataFrame,
                           reference_values: Optional[Union[pd.Series, Dict]] = None
                           ) -> Tuple[List[float], float]:
    """
    Calculate NNV contributions for each variable.
    Implements the core NNV algorithm:
    1. For each variable, replace its current value with the
       reference value.
    2. Calculate the monitoring statistic for the modified data.
    3. Compute the absolute difference from the original.
    Args:
        current_data: Current process data (single observation
                     as a one-row DataFrame).
        reference_values: Reference values to use. Falls back to
                        stored values when None.
    Returns:
        Tuple of (contributions, total_contribution).
    Raises:
        ValueError: If inputs are invalid.
        RuntimeError: If the monitoring statistic cannot be
                     retrieved.
    """
    if current_data.empty:
        raise ValueError("Current data cannot be empty")
    if current_data.shape[0] != 1:
        raise ValueError(
            "Current data must contain exactly one observation")
    # Resolve reference values
    if reference_values is not None:
        ref_vals = (pd.Series(reference_values)
                    if isinstance(reference_values, dict)
                    else reference_values)
    else:
        if self._reference_values is None:
            raise ValueError(
```

```

        "No reference values set. "
        "Use set_reference_values() first.")
    ref_vals = self._reference_values
# Fill in any missing variables with their current value
missing_vars = (set(current_data.columns)
                - set(ref_vals.index))
if missing_vars:
    warnings.warn(
        f"Missing reference values for: {missing_vars}")
    for var in missing_vars:
        ref_vals[var] = current_data[var].iloc[0]
# Original monitoring statistic
self.model.predict(current_data.copy(),
                   f_pp=self.preprocessing_function)
original_statistic = self._get_monitoring_statistic()
# Per-variable contributions
contributions: List[float] = []
for variable in current_data.columns:
    try:
        modified_data = current_data.copy()
        modified_data.loc[
            modified_data.index[0], variable
        ] = ref_vals[variable]
        self.model.predict(
            modified_data,
            f_pp=self.preprocessing_function)
        modified_statistic = (
            self._get_monitoring_statistic())
        contributions.append(
            abs(modified_statistic - original_statistic))
    except Exception as e:
        warnings.warn(
            f"Error calculating contribution "
            f"for '{variable}': {e}")
        contributions.append(0.0)
total_contribution = float(np.sum(contributions))
if total_contribution == 0:
    warnings.warn("Total contribution is zero.")
return contributions, total_contribution

```

## C.4 Normalised Contributions

For interpretability and cross-observation comparability, contributions are typically normalised to sum to unity. A uniform distribution indicates that no variable is preferentially responsible, whereas a concentrated distribution points to specific root causes.

```
def calculate_normalized_contributions(self,
    current_data: pd.DataFrame,
    reference_values:
        Optional[Union[pd.Series, Dict]] = None
) -> pd.Series:
    """
    Calculate normalized NNV contributions (summing to 1.0).
    Args:
        current_data: Current process data (one observation).
        reference_values: Reference values to use (optional).
    Returns:
        Series with normalised contribution per variable.
    Raises:
        ZeroDivisionError: If total contribution is zero.
    """
    contributions, total_contribution = (
        self.calculate_contributions(
            current_data, reference_values))
    if total_contribution == 0:
        raise ZeroDivisionError(
            "Cannot normalize contributions: "
            "total contribution is zero")
    normalized = np.array(contributions) / total_contribution
    return pd.Series(
        normalized,
        index=current_data.columns,
        name='nnv_contributions')
```

The method raises a `ZeroDivisionError` when all contributions are zero — a situation that arises when the observation is identical to the reference or when numerical precision prevents any measurable difference.

## C.5 run\_analysis

The `run_analysis` method orchestrates the complete NNV diagnosis procedure over a sequential dataset, integrating three mechanisms:

- Sequential monitoring: Each observation is fed to the model one at a time, simulating an online data stream;
- Adaptive reference updating: When the monitoring statistic falls within normal limits, the reference values are updated via exponential moving average, allowing the method to track slow, legitimate process drifts;
- Concept drift detection: The ADWIN algorithm monitors the stream of monitoring statistic values; when a distributional change is detected, the reference window is reset to the most recent normal observations, preventing stale references from inflating contributions.

```
def run_analysis(self, dataset: pd.DataFrame,
                 adwin_delta: float = 0.01
                 ) -> Tuple[pd.DataFrame, pd.Series]:
    """
    Execute complete NNV analysis on a dataset with adaptive
    reference updating.
    Uses ADWIN drift detection to adaptively update reference
    values during normal operation and reset them when concept
    drift is detected.
    Note: This method mutates the stored reference values
    as part of the adaptive updating procedure.
    Args:
        dataset: Complete dataset (training + monitoring).
        adwin_delta: Sensitivity for ADWIN drift detector.
    Returns:
        Tuple of (contributions, spe_series) where
        contributions is a DataFrame of normalized NNV values
        and spe_series holds the monitoring statistic per
        sample.
    Raises:
        ValueError: If dataset is empty or reference values
                    are not set.
        AttributeError: If the model has no control limit.
    """
    if dataset.empty:
```

```

        raise ValueError("Dataset cannot be empty")
    if self._reference_values is None:
        raise ValueError(
            "Reference values must be set before running "
            "analysis. Use set_reference_values() first.")
    control_limit = self._get_control_limit()
    spe_values: list[float] = []
    spe_index: list = []
    contrib_NNV = pd.DataFrame(
        index=dataset.index,
        columns=dataset.columns,
        dtype=float)
    reference = self._reference_values.copy()
    n_window = len(dataset) // 2
    adwin = drift.ADWIN(delta=adwin_delta)
    for i in range(dataset.shape[0]):
        df_now = create_data_stream(dataset, i)
        # Prediction and monitoring statistic
        self.model.predict(
            df_now, f_pp=self.preprocessing_function)
        spe_value = self._get_monitoring_statistic()
        spe_values.append(spe_value)
        spe_index.append(df_now.index[0])
        # Update ADWIN
        adwin.update(spe_value)
        # Update reference if within normal limits
        if spe_value <= control_limit:
            self.update_reference_values(
                df_now,
                update_method='exponential',
                alpha=1 / (n_window + 1))
            reference = self._reference_values.copy()
            n_window += 1
        # Drift detection -- reset ADWIN and reference
        if adwin.drift_detected:
            if spe_value <= control_limit:
                width = getattr(adwin, "width",
                                n_window)
                recent_width = (int(width)
                                if np.isfinite(width)
                                else n_window)
                recent_start = max(
                    0, i - recent_width)
                recent_window = dataset.iloc[

```

```

        recent_start:i + 1]
    if not recent_window.empty:
        self.set_reference_values(
            recent_window)
        reference = (
            self._reference_values.copy())
        n_window = len(recent_window)
        adwin = drift.ADWIN(delta=adwin_delta)
# NNV contributions
try:
    contrib_norm = (
        self.calculate_normalized_contributions(
            df_now, reference))
    contrib_NNV.loc[df_now.index[0]] = (
        contrib_norm.values)
except Exception:
    contrib_NNV.loc[df_now.index[0]] = (
        np.zeros(len(dataset.columns)))
spe_series = pd.Series(
    spe_values, index=spe_index, name="SPE")
contrib_NNV = contrib_NNV.fillna(0.0)
return contrib_NNV, spe_series

```

## C.6 create\_data\_stream

```

def create_data_stream(data: pd.DataFrame,
                      index: int) -> pd.DataFrame:
    """
    Extract a single observation from a dataset as a one-row
    DataFrame.
    Args:
        data: Input dataset.
        index: Positional index of the observation to extract.
    Returns:
        One-row DataFrame containing the selected observation.
    Raises:
        ValueError: If data is empty.
        IndexError: If index is out of bounds.
    """
    if data.empty:
        raise ValueError("Data cannot be empty")
    if index < 0 or index >= len(data):
        raise IndexError(

```

```
f"Index {index} is out of bounds for data "  
f"with {len(data)} rows")  
return pd.DataFrame(data.iloc[index, :]).T
```

This module-level function provides a clean interface for extracting a single observation from a DataFrame while preserving the original column names, data types, and the observation's timestamp index.