Learning-based approaches for automatic fault detection and diagnosis in industrial systems

Politecnico
di Bari

Doctoral Dissertation
Doctoral Program in Electrical and Information Engineering (36^{th}cycle)

Learning-based Approaches for Automatic Fault Detection and Diagnosis in Industrial Systems

By

Bahman Askari

******

Supervisor(s):
Prof. Engr. Mariagrazia Dotoli, Supervisor
Dr. Engr. Raffaele Carli, Co-Supervisor
Dr. Engr. Graziana Cavone, Co-Supervisor

Doctoral Examination Committee:
Prof. Engr Andrea BONCI, Università Politecnica delle Marche
Dr. Engr Claudio SAVAGLIO, Università della Calabria

Politecnico di Bari
2024

I would like to dedicate this thesis to my beloved family.

Acknowledgements

I would like to express my heartfelt gratitude to the following individuals whose support and guidance have been instrumental during my Ph.D. journey. First and foremost, my deepest appreciation goes to Prof. Mariagrazia Dotoli, who was my supervisor during the Ph.D. program at the Polytechnic University of Bari in Italy. Her unwavering support, invaluable guidance, and generous assistance have been fundamental in shaping the direction of my research. I am truly grateful for the supervision she has provided throughout my academic pursuit. I would also like to extend my gratitude to Dr. Raffaele Carli and Dr. Graziana Cavone for their insightful technical discussions, which have significantly contributed to the development of my work. Their expertise and feedback have been invaluable in advancing my research. I am indebted to Prof. Antoine Grall for granting me the remarkable opportunity to join the Computer Science and Digital Society (LIST3N) laboratory and his research group at the University of Technology of Troyes (UTT) in France. The collaborative environment within the group, particularly in association with Dr. Yves Langeron, has greatly enriched my research experience and broadened my horizons. Last, but certainly not least, I would like to express my profound gratitude to my friends and colleagues who have provided unwavering support and assistance throughout my academic journey. I am grateful to each of you for your contributions to my academic and personal growth. Your guidance and encouragement have been pivotal to my success, and I am truly fortunate to have had you by my side.

List of Publications

Publications Related to This Thesis:

- B. Askari, R. Carli, G.Cavone, M. Dotoli, "Data-Driven Fault Diagnosis in a Complex Hydraulic System based on Early Classification", 1st IFAC Workshop on Control of Complex Systems (COSY 2022), November 24-25, 2022, Bologna, Italy, IFAC-PapersOnLine 55 (40), 187-192.
  DOI: 10.1016/j.ifacol.2023.01.070.

- B. Askari, G. Cavone, R. Carli, A. Grall, and M. Dotoli, "A Semi-Supervised Learning Approach for Fault Detection and Diagnosis in Complex Mechanical Systems", 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Auckland, New Zealand, August 2023, pp. 1-6.
  DOI: 10.1109/CASE56687.2023.10260469.

- B. Askari, Y. Langeron, G. Cavone, R. Carli, A. Grall, and M. Dotoli, "An Integrated Approach for Failure Diagnosis and Analysis of Industrial Systems Based on Multi-Class Multi-Output Classification: A Complex Hydraulic Application", the 33rd European Safety and Reliability (ESREL) Conference. University of Southampton, United Kingdom, September 2023.
  DOI: 10.3850/978-981-18-8071-1-P456-cd.

- B. Askari, A. Bozza, G. Cavone, R. Carli, and M. Dotoli, "An adaptive constrained clustering approach for real-time fault detection of industrial systems", European Journal of Control, 2023, 100858, pp 1-10.
  DOI: 10.1016/j.ejcon.2023.100858.

Additional Paper (Not Strictly Related to Learning-Based Approach):

- A. Bozza, B. Askari, G.Cavone, R. Carli and M.Dotoli, "An Adaptive Model Predictive Control Approach for Position Tracking and Force Control of a Hydraulic Actuator", 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 2022, pp. 1029-1034.
  DOI:10.1109/CASE49997.2022.9926645.

Abstract

The timely detection and diagnosis of faults are crucial for maintaining the safety and reliability of industrial systems, as they help prevent severe damage and unexpected disruptions in operations. In pursuit of this objective, learning-based approaches have emerged as powerful tools, harnessing various machine learning techniques to detect potential faults and diagnose their root causes in the systems under study. This thesis delves into the application and advancement of machine learning methods, especially in the context of hydraulic and pneumatic systems, vital components at the heart of industrial machinery. The primary focus is to enhance the fault detection and diagnosis capabilities within these critical domains, thereby contributing to the overall performance and longevity of industrial machinery. The hydraulic system serves as a primary focus of investigation, where an early time series classification method is applied to detect faults as early as possible. Leveraging data-driven approaches, the aim is to detect potential faults in the multi-component hydraulic system and diagnose their underlying causes using a multi-class multi-output classification method. The study encompasses the development of algorithms capable of recognizing deviations from normal system behavior and, in turn, determining the specific issues that trigger these deviations. Moving forward, my exploration extends to both pneumatic and hydraulic systems in case of label scarcity in the dataset. To this aim, semi-supervised learning approaches are combined with conventional classification methods to harness the power of unlabeled data and improve model generalization and performance in fault detection and diagnosis. Finally, by employing adaptive machine learning methods in this context, an adaptive constraint clustering algorithm is presented for real-time fault detection in the pneumatic system. The results of this thesis are anticipated to provide practical solutions for maintaining the safety and reliability of complex industrial systems.

Keywords: Fault Detection and Diagnosis, Hydraulic Systems, Industrial Systems, Machine Learning, Pneumatic Systems

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, I delve into the world of fault detection and diagnosis (FDD) in complex industrial systems. To build a strong foundation for my exploration, I start by thoroughly examining the background and motivations driving the various methods and approaches used in FDD. I take a close look at the unique challenges posed by complex systems and the crucial need for proactive fault detection. Then, I articulate the precise problem statements that each of my projects tackles, shedding light on the intricacies and complexities of fault detection in industrial systems. As I progress, the research objectives for each activity become clearer, guiding us towards innovative solutions. Finally, I highlight the contributions of my research, all aimed at enhancing the performance and maintenance of manufacturing lines and industrial systems. In this comprehensive chapter, I prepare the groundwork for an in-depth exploration of my research journey, centered around the core themes of comprehension, addressing challenges, and advancing the field of FDD in smart manufacturing.

1.1 Background and Motivation

Within the scope of my research, I have made substantive contributions through the examination of two distinct industrial case studies, categorizing my endeavors into three distinct sections. In each of these sections or activities, I have judiciously employed a variety of ML methodologies, thereby ensuring the most suitable analytical tools are applied to the specific tasks at hand. This approach allows us to leverage the

full spectrum of ML capabilities in my efforts to address the multifaceted challenges presented by these industrial contexts. Broadly, ML methods can be categorized into four types: supervised, unsupervised, semi-supervised, and reinforcement learning. Given the characteristics of the dataset, I concentrate on the first three methods while excluding reinforcement learning from the thesis.

In the first two activities, I harnessed the capabilities of supervised ML methodologies. Since the complexity of industrial systems in various applications necessitates efficient fault diagnosis and conventional methods typically detect faults at the end of working cycles, which can lead to costly downtime and potential damage. Chapter 3 addresses the critical need for early fault diagnosis in a hydraulic system, motivated by the desire to strike a balance between the accuracy of fault detection and the earliness of detection.

On the other hand, in chapter 4, the failure of individual components of complex systems may not immediately lead to system failure but can eventually disrupt system operations. The motivation behind this research is to provide a comprehensive analysis of potential failures and their causes, addressing the need for a holistic understanding of complex systems' reliability using a specific supervised ML method called multi-class multi-output (MCMO) classification.

In chapter 5, my approach shifted towards the utilization of semi-supervised learning (SSL) techniques, a strategic response to the scarcity of labeled data where the integration of artificial intelligence into mechanical FDD is crucial for enhancing reliability and reducing costs in Industry 4.0 applications. However, dynamic environments often pose challenges, including limited labeled data and the presence of rare or difficult-to-detect faults. The motivation here is to leverage SSL to address these challenges.

Finally, in chapter 6, my methodology culminated in the application of unsupervised ML techniques, adeptly employed to process incoming unlabeled data streams in real-time where the widespread deployment of sensors in Industry 4.0 has elevated the role of data-driven methods in FDD. This research activity introduces an Adaptive Constrained Clustering Algorithm (ACCA) for real-time fault detection in industrial machines, driven by the need for responsive and accurate fault detection methods.

1.2 Problem Statement

Regarding the reasons for my previous activities, I will present the problem statements in three different aspects.

In the first supervised activity, the problem revolves around detecting various fault types in hydraulic system components before completing a full working cycle. Early detection of these faults is essential to minimize downtime, reduce repair costs, and prevent further damage to the system. Traditional methods are often reactive, whereas this research activity seeks proactive solutions for early fault classification.

Then the second activity tackles the challenge of diagnosing potential failures in multi-component systems and delving deeper into the root causes of these failures. The focus is on preventing system disruptions, minimizing downtime, and ensuring smooth system operations, especially in scenarios where component failures can cascade into critical issues.

Subsequently, the semi-supervised learning (SSL) method addresses the problem of detecting potential failures in dynamic mechanical systems where labeled data are scarce and expensive. Faults in such systems may be rare or challenging to identify and might not be fully represented in the labeled dataset.

Finally, the adaptive constrained approach addresses the problem of real-time fault detection in industrial machines using continuously incoming monitoring data. It focuses on achieving results comparable to offline methods but with higher responsiveness.

1.3 Research Objectives

In this section, I outline the research goals and objectives for the activities mentioned above.

The primary research objective is to develop an early time-series classification (ETSC) algorithm tailored to hydraulic systems. This algorithm aims to classify the system's state early in the working cycle while maintaining high classification accuracy. By achieving this objective, this work seeks to provide a practical solution for early fault detection in complex hydraulic systems.
The second research objective is to propose a data-driven model with a two-step

decision approach. In the first step, the MCMO classification technique is employed to diagnose potential failures based on sensor signals. In the second step, failure analysis (FA) is applied to investigate the root causes of these failures. This approach aims to comprehensively analyze potential failures in complex systems.
In the case of SSL, the research objective is to combine graph-based SSL (GSSL), particularly relying on label propagation, with conventional classification algorithms to detect potential failures in complex mechanical systems. This approach aims to enlarge labeled datasets and improve the accuracy of identifying non-nominal conditions.

Lastly, the research objective of the unsupervised method is to develop the ACCA, which consists of a two-stage procedure: micro-clustering and constrained macro-clustering. The micro-clustering stage groups batches of work-cycle data into micro-clusters in real-time, while the macro-clustering stage groups micro-cluster features into macro-clusters offline. This approach aims to provide real-time fault detection with high responsiveness and accuracy.

1.4 Contribution of the Thesis

In this section, I elucidate my contributions and innovations within each research direction.
The first contribution to the doctoral thesis is the introduction of the ETSC method, which demonstrates superior accuracy and earlier fault detection capabilities compared to baseline methods. This advancement holds promise for optimizing the performance and maintenance of manufacturing lines, reducing costly downtime, and enhancing overall productivity.
The second contribution is the introduction of a two-step approach that combines MCMO classification and FA. The approach enhances system reliability, reduces downtime, and minimizes the impact of failures on system operations. It offers a promising alternative to conventional methods and contributes valuable insights into improving the performance and maintenance of manufacturing lines.
The third contribution lies in the application of GSSL to expand labeled datasets and effectively identify various types of non-nominal conditions in complex mechanical systems. It provides a valuable addition to the toolbox of fault detection methods,

contributing to the optimization of manufacturing lines' performance and maintenance.
The last contribution is the ACCA, which achieves results equivalent to offline baseline methods with improved responsiveness and processing speed. It represents an innovative approach to real-time fault detection and contributes to the optimization of manufacturing lines by providing timely insights for maintenance and performance improvement.

1.5 Description of Case Studies Conducted in the Thesis

In this thesis, I focus on two mechanical datasets, namely hydraulic and pneumatic. The multi-component hydraulic system (HS) dataset is centered around the condition assessment of a hydraulic test rig, utilizing multi-sensor data. Additionally, the pneumatic dataset revolves around a single-component pneumatic system (PS) configuration, consisting of a rotary gripper controlled by pneumatic actuators that operates using linear pneumatic pistons and 4-way valves.

1.5.1 Hydraulic System

The first dataset concerns a real complex and multi-component HS, whose predictive maintenance experimental data set is available in [13]. The observable data are related to the failures due to different causes across different working cycles. The considered HS consists of a test rig equipped with a primary working and a secondary cooling-filtration circuit [13], which are connected via an oil tank, as shown in Fig. 1.1. The primary working circuit consists of the main pump $MP1$, switchable accumulators $A1-A4$, filter $F1$, and a set of valves. The secondary cooling-filtration circuit –which is responsible for cooling and filtering the hydraulic oil– consists of the hydraulic pump $SP1$, a three-way solenoid valve, filter $F2$, cooler $C1$, and a variety of sensors [1].

In this HS, the state of the four main components ($C1$, $V10$, $MP1$, $A1-A4$) varies dynamically, while the monitoring system cyclically records process values

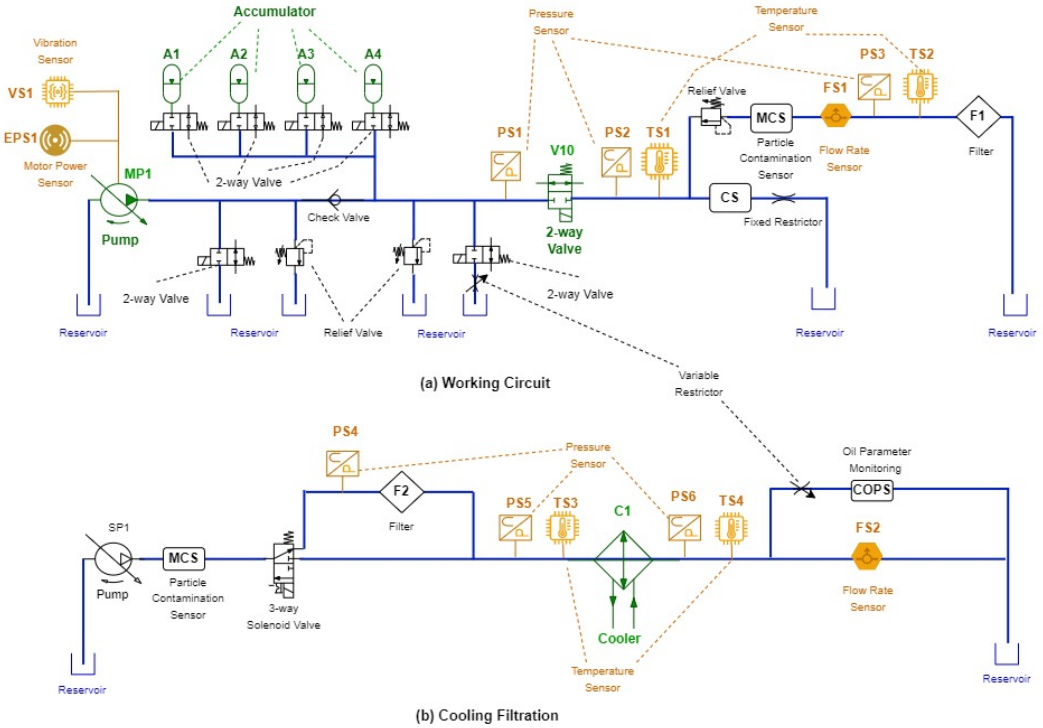


Fig. 1.1 Hydraulic system scheme: (a) primary working circuit and (b) secondary cooling-filtration circuit (adapted from [1]).

such as pressures, flow rates, and temperatures during sixty-second working cycles. Moreover, several types of faults may occur in the hydraulic components, which differ in fault types, severity, and duration. In Fig. 1.1 the green color represents the key elements of the HS (e.g., accumulators, pumps, valve, and cooler), while the orange ones represent the sensors, such as vibration, pressure, and temperature sensor. Table 1.2 shows the components and simulated fault conditions where the monitoring task in the given HS is conducted for four components: cooler $C1$, valve $V10$, main pump $MP1$, and accumulators $A1 - A4$ (see Fig. 1.1).

The final state of the HS is represented using a stable flag SF where the value 1 indicates a system failure type and the 0 value means that the system works in a healthy mode. Indeed, the HS runs over 2205 working cycles, and the corresponding samples are collected under different conditions. Some data samples were measured under stable conditions, whereas others were recorded when static conditions might

Table 1.1 Description of sensors deployed in the case study HS.

Identifier	Measured quantity	Unit	Frequency
PS1-6	Pressure	Pa	100 Hz
EP1	Motor Power	W	100 Hz
FS1-2	Flow Rate	Lit/min	10 Hz
TS1-4	Temperature	°C	1 Hz
VS1	Vibration	mm/s	1 Hz
CE	Cooling Efficiency	%	1 Hz
CP	Cooling Power	kW	1 Hz
SE	System Efficiency Factor	%	1 Hz

not have been reached yet. Data from both conditions are recorded in the predictive maintenance data set of the HS [13].

1.5.2 Pneumatic System

The PS dataset was collected from a single-component PS deployed in a real industrial process [14]. This system refers to the robotic work-cell that has been produced by an Italian automotive company leader in the European market.

In this PS, the rotary gripper is driven by two pneumatic actuators respectively called “A side" and “B side", whose pneumatic scheme is represented in Fig. 1.2.a. Each side consists of a linear pneumatic piston and a typical 4-way valve for provisioning the needed air flow-rate into the chambers of the related piston. The gripper is equipped with flow meters so that during each whole work-cycle both the maximum and mean flow rate of the air entering the piston chambers are retrieved and collected by an onboard acquisition system. Moreover, the overall working time of each 4-way valve (one for each piston side) is computed as the absolute value of the difference between the time when the actuation command is given to the valve and the time when the related piston rod reaches its final stroke. Therefore, for each work-cycle, six different monitoring features are collected and recorded in the pneumatic dataset, as summarized in Table 1.3.

The whole experiment is based on a data set of N=600 work-cycles, comprehending 100 nominal and 500 non-nominal operating conditions shuffled in a random

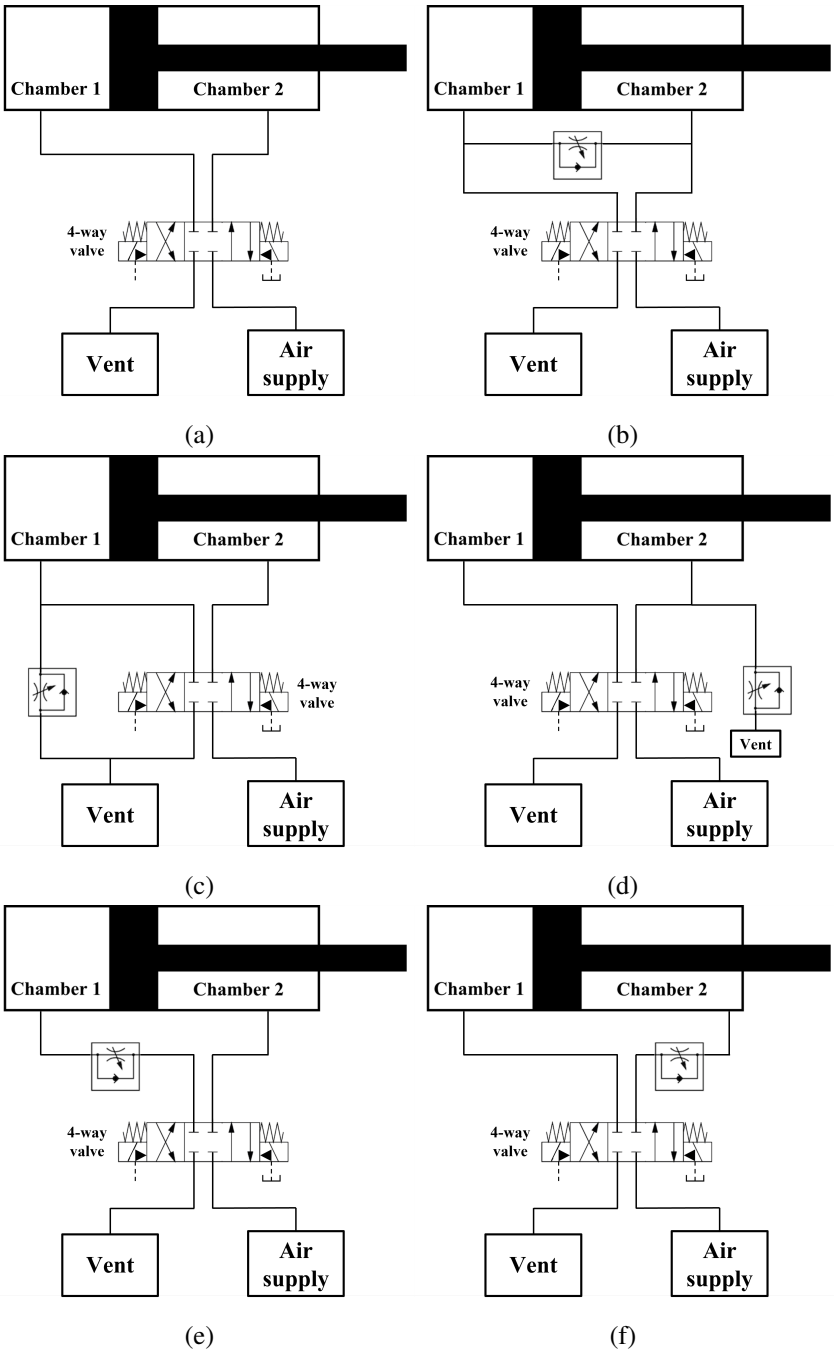


Fig. 1.2 Pneumatic schemes of the actuator used for driving the rotary gripper in the case of nominal working condition (a), internal air flow-rate leakage (b), external air flow-rate leakage for the A side (c) and the B side (d), internal occlusion for the A side (e) and the B side (f).

Table 1.2 Description of fault types for each HS component

Component	Value	Fault Severity	Samples
Cooler	3%	total failure	732
C1	20%	reduced efficiency	732
	100%	full efficiency	741
Valve	73%	total failure	360
V10	80%	severe lag	360
	90%	small lag	360
	100%	optimal behavior	1125
Pump	0	no leakage	1221
MP1	1	weak leakage	492
	2	severe leakage	492
Accumulators	90 bar	total failure	808
A1-A4	100 bar	severely reduced pressure	399
	115 bar	slightly reduced pressure	399
	130 bar	optimal pressure	599
HS stable flag	0	stable conditions	1449
SF	1	unstable conditions	756

Table 1.3 Data features collected during each work-cycle of PS dataset.

Feature name	Description	Unit
MaxFlowA	Maximum flow-rate on side A	l/min
MaxFlowB	Maximum flow-rate on side B	l/min
IntegrFlowA	Mean flow-rate on side A	l/min
IntegrFlowB	Mean flow-rate on side B	l/min
TimeA	Working time of the 4-way valve on side A	s
TimeB	Working time of the 4-way valve on side B	s

fashion. The faulty conditions regard the rotary gripper over the complete work-cycles of the robotic cell. Specifically, as described in Table 1.4, five different anomaly conditions are generated. It is worth pointing out that the robotic work-cell under study may also be affected by other less frequent types of faults in addition to the described ones. Nevertheless, only data related to the five considered faults have been provided by the automotive Italian company that produces the work-cell. As indicated in the Table 1.4, the external leakage of airflow and the internal occlusion are simulated for both pistons of the pneumatic actuator, while the internal leakage is the same for both sides. For the sake of implementing the above-described faulty scenarios, three different configurations of the pneumatic circuit are deployed:

Table 1.4 Description of the fault types addressed by the case study PS.

Fault type	Number of instances	System state
No fault	100	Nominal
Ext. leakage on side A	100	Non-nominal
Ext. leakage on side B	100	Non-nominal
Int. occlusion on side A	100	Non-nominal
Int. occlusion on side B	100	Non-nominal
Internal leakage	100	Non-nominal

1) Pneumatic circuit generating the internal air flow-rate leakage – An additional valve is inserted between both the outputs of the 4-way valve (Fig. 1.2.b).

2) Pneumatic circuit generating the external air flow-rate leakage – Two additional valves are inserted between the air venting source (i.e., flow-rate not in pressure) and chamber 1 of the piston for the A side (Fig. 1.2.c) and between the air supply source (i.e., the flow-rate in pressure) and chamber 2 of the piston for the B side (Fig. 1.2.d), respectively.

3) Pneumatic circuit generating the internal occlusion – Two additional valves are inserted between the pressure pipeline output of the 4-way valve (i.e., flow-rate not in pressure) and chamber 1 of the piston for the A side (Fig. 1.2.e) and between the venting pipeline output of the 4-way valve (i.e., flow-rate not in pressure) and chamber 2 of the piston for the B side (Fig. 1.2.f), respectively.

1.6 Outline of the Thesis

In chapter 2, I provide an extensive overview of diverse ML methods, with a comprehensive review of their related work in the domain of FDD. Moving forward, chapters 3 and 4 introduce two prominent supervised techniques, namely ETSC and MCMO Classification, illustrating their application in the HS case study with detailed results and analysis. Chapter 5 delves into SSL, focusing on label propagation-based SSL, and demonstrates its utilization in both HS and PS datasets for limited labeled data scenarios. Subsequently, chapter 6 unfolds the concept of adaptive clustering for real-time FDD, showcasing its practical implementation using streaming data from the PS for real-time fault detection, accompanied by results and profound analysis. Finally, chapter 7 concludes the thesis, summarizing key findings, contributions, and

prospects for future research, encapsulating the entirety of my journey into the FDD of industrial systems.

Chapter 2

Literature Review

In the context of manufacturing lines, FDD plays a critical role in optimizing performance and maintenance [15–17]. FDD encompasses various techniques and approaches that aim to identify and address failures in a timely manner. One prominent approach in FDD is learning-based FDD, which leverages the power of data analysis and ML methods to detect and diagnose failures effectively [18].

The ML model development process begins with data collection, which is followed by data preprocessing to clean, transform, and make the dataset suitable for learning. Then the dataset is typically divided into training, validation, and test sets. The appropriate ML algorithm is then chosen, and the model is trained on the training set. Hyperparameters of the trained model are tuned to optimize performance using the validation set. After fine-tuning the model, the performance is then evaluated on the test dataset using various metrics. Once satisfactory results are achieved, the model is deployed for real-world use, with ongoing monitoring and maintenance. Interpretation and visualization techniques can provide insights into the model's decision-making process. The specific details of each step can vary depending on the problem, dataset, and ML approach used.

We typically categorize the ML algorithms into four main types: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning involves training a model using labeled data, where the input features and their corresponding output labels are provided. Unsupervised learning deals with unlabeled data, where only the input features are available without any corresponding output labels. Semi-supervised learning lies in between supervised

and unsupervised learning. It leverages a combination of labeled and unlabeled data for training [19]. Reinforcement learning is an ML paradigm where an agent learns to make sequential decisions by interacting with an environment to maximize a cumulative reward.

2.1 Supervised Machine Learning Methods

Supervised learning, a subfield of ML, offers a powerful approach to train predictive models by utilizing labeled data. Supervised learning involves the use of algorithms that learn from labeled samples, where the input data (features) and corresponding output (labels) are provided. These algorithms are trained to recognize patterns in the data and make predictions based on these learned patterns [20–22]. In the realm of ML, the development of a supervised learning algorithm typically involves a structured approach consisting of three distinct phases: the training phase, the validation phase, and the testing phase. Accordingly, it is advisable to partition the dataset into three separate subsets, each allocated to one of these phases. Furthermore, it is essential to identify or choose appropriate performance evaluation metrics that will be used to assess and refine the supervised learning models throughout the training, validation, and testing processes. In simple terms, supervised learning means the tuning of model parameters using labeled data sets so that the tuned model parameters can work for larger and unseen data. Considering a scenario where I have n models represented as $y = f_{\mathbf{w}_1}(\mathbf{x}), y = f_{\mathbf{w}_2}(\mathbf{x}), \ldots, y = f_{\mathbf{w}_n}(\mathbf{x})$, and I select the best model $y = f_{\mathbf{w}_i}(\mathbf{x})$ through training and validation processes by using a labeled data set. The performance of the selected model is evaluated by testing it on another data set which is generally smaller than the training data set.[23].

2.1.1 A Comprehensive Guide to Supervised Learning

There are two main types of supervised learning models: classification models and regression models which offer valuable tools for FDD in manufacturing lines and production datasets. Classification models enable the identification and classification of failures into discrete categories, while regression models provide quantitative estimates of failure severity. The choice between these models depends on the

specific objectives and requirements of the problem at hand. Given the categorical nature of my dataset, I focus my attention on classification methods.

- Classification Methods

Classification models can be employed to classify instances as either normal or faulty based on the available input features. These models learn from labeled data that includes information about whether a failure has occurred or not. By analyzing the patterns and relationships between the input features and the associated failure labels, classification models can make predictions about the presence or absence of failures. The mathematical notation of the classification methods is detailed in the appendix A.1.

In this thesis, I have explored and utilized various classification methods as baselines for comparisons and evaluations. While there exist numerous classification algorithms, I specifically focus on introducing a selection of commonly used methods that have demonstrated their effectiveness in the field of FDD. These methods serve as reference models against which I compare the performance of my proposed approaches and assess their efficacy in optimizing the performance and maintenance of manufacturing lines. Notably, the classification algorithms employed in this study include mathematical learning models such as Logistic Regression (LR) and support vector machine (SVM), hierarchical models such as decision trees (DT) and random forests (RF), K-nearest neighbor (KNN), and deep learning, which have been extensively studied and proven to be effective in similar contexts.

The two mathematical learning models considered in this section are logistic regression and SVM techniques. The logistic regression model has been built upon the probability nature of the classes, and the SVM model has been built upon the separability nature of the classes. In the following subsections, these two mathematical models are discussed using fewer variables and simple examples. The explanation includes the optimization of an error factor and derivation of the models.

Logistic Regression LR is a widely used classification algorithm that models the relationship between the input features and the probability of belonging to a certain class. It is particularly suitable for binary classification problems but can be extended to handle multi-class classification through techniques like one-vs-rest

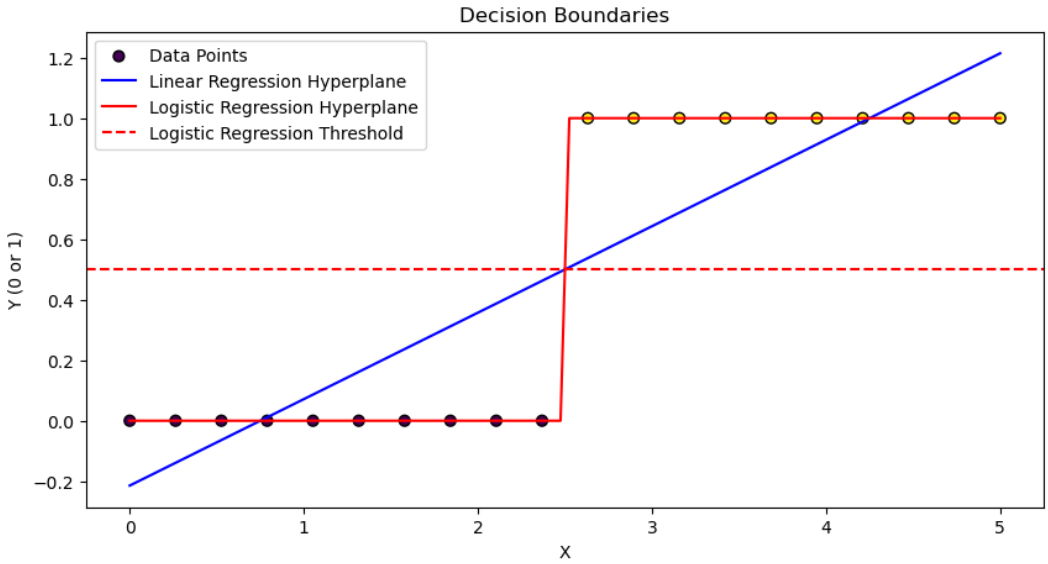


Fig. 2.1 Linear Regression (blue line) and Logistic Regression (red line) models on a simple dataset with 20 data points.

or multinomial logistic regression. In LR, the algorithm applies a logistic function (sigmoid function) to the linear combination of the input features. This transformation maps the output to a value between 0 and 1, representing the probability of belonging to the class labels. In figure (2.1) I created a simple dataset with 20 data points then I fitted both Linear Regression and Logistic Regression models to this data. The blue line represents the Linear Regression hyperplane, and the red line represents the Logistic Regression hyperplane. By setting an appropriate threshold, instances can be classified into different classes. LR is computationally efficient and interpretable, making it a popular choice for the FDD in manufacturing lines. It can handle both numerical and categorical input features and does not make strong assumptions about the distribution of the features. LR also allows for the analysis of feature importance through the examination of the coefficients associated with each feature. The mathematical notation of the optimization and objective function for LR is detailed in the appendix A.1.1.

Support Vector Machines SVM [24] is a powerful classification algorithm that finds an optimal hyperplane in a high-dimensional feature space to separate instances belonging to different classes. SVM aims to maximize the margin between

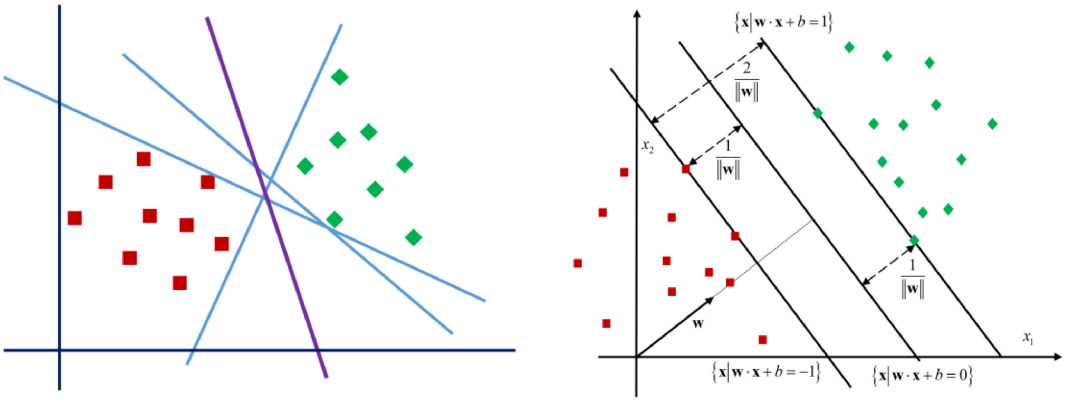


Fig. 2.2 (left) Representation of separating hyperplane for two-class data and (right) the optimal hyperplane to separate two-class data. The margin between support vectors (d) is equal to $\frac{2}{\|\mathbf{w}\|}$[2].

the hyperplane and the nearest instances, known as support vectors. SVM can handle linear and nonlinear classification tasks by using different kernel functions. It is effective in handling datasets with a large number of features and can handle outliers well. The optimization problem is the basis for the development of linear SVM models. It can also be used to make an optimal straight-line cut (in any direction) that divides the data domain into two for classifying two classes. Here I focus only on the two-class (or binary classification) SVM in detail while the SVM for multi-class classification can be created using combinations of several two-class SVMs.

- **Linear SVM Classification:** The two classes can clearly be separated easily with a straight line. In linear SVM (figure 2.2), the algorithm aims to find a hyperplane ($\mathbf{w}\cdot\mathbf{x}+b=0$) that best separates data into different classes. The optimal hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the nearest data points (support vectors $\mathbf{w}\cdot\mathbf{x}+b=1$ and $\mathbf{w}\cdot\mathbf{x}+b=-1$) from each class. Note that many possible linear classifiers can separate the data, but there is only one that maximizes the margin. This linear classifier is known as the optimal separating hyperplane (see figure 2.2).

- **Nonlinear SVM Classification:** Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable. One approach to handling nonlinear datasets is

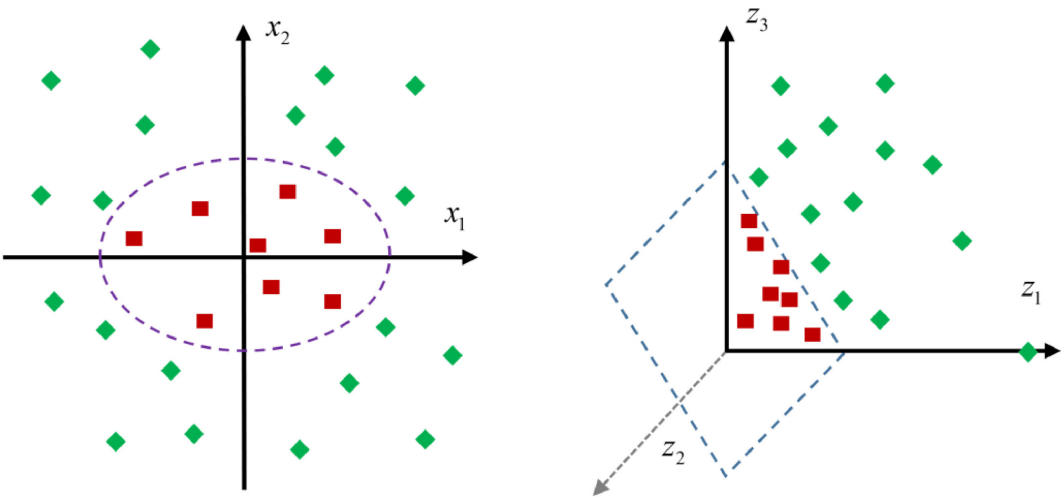


Fig. 2.3 Mapping of input space X into high-dimensional feature space Z [2].

to add more features, such as polynomial features. Polynomial and Radial Basis Functions (RBF) are the most common non-linear SVMs that extend the capabilities of SVM to handle data that is not linearly separable. This is done by transforming the data into a higher-dimensional feature space where it becomes separable by a hyperplane. In other words, the classification may be carried out either in a vector space or in a feature space, where the vector space is defined as the space that contains the original features, and the feature space is defined as the space that contains the transformed features using kernel functions. It can be observed from Figure (2.3) that the two-class data are not linearly separable in the original input space but are possible in a three-dimensional feature space.

The mathematical notation of the linear SVM is detailed in the appendix A.1.2. Therefore, the steps in the linear SVM are the mapping of the data domain into a response set and the dividing of the data domain. The steps in the nonlinear SVM are the mapping of the data domain to a feature space using a kernel function [25], the mapping of the feature space domain into the response set, and then the dividing of the data domain.

SVMs are effective in finding the optimal decision boundary, which results in good generalization to unseen data. They work well in high-dimensional spaces, making them suitable for various applications, including text classification and

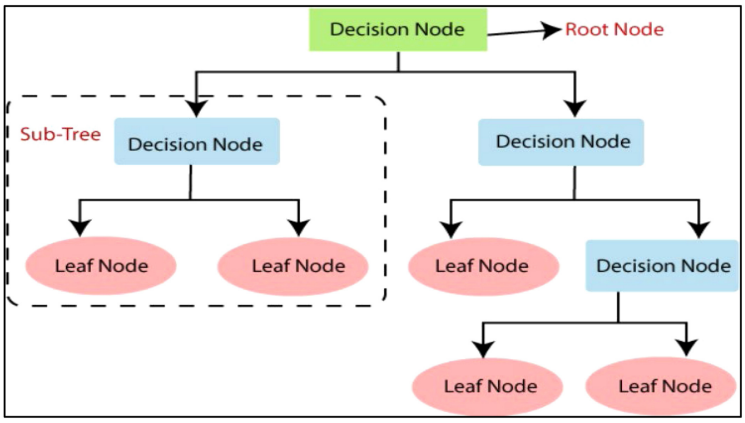


Fig. 2.4 Schematic diagram of a decision tree [3].

image recognition. SVMs can be sensitive to the choice of kernel function and its hyperparameters. Training SVMs on large datasets can be computationally intensive. Interpretability can be challenging, especially in non-linear SVMs.

Decision Trees Hierarchical models help to classify even isolated groups of points by connecting them with their parent classes using tree-like structures. These models are highly suitable for modern requirements, which include big data and distributed machine learning. It adopts both regression and classification strategies with a tree that may be built through a sequence of decisions. Hence, it is also called a decision tree (DT) [23].

DT [26, 3] classification is an ML algorithm that is based on a binary tree structure and is particularly useful for solving problems where you need to make decisions by following a sequence of rules based on input features. DT recursively splits the data based on different feature thresholds to create branches and leaf nodes representing different class labels. DTs are interpretable and can handle both numerical and categorical features. They are effective in capturing complex decision boundaries and can handle high-dimensional data.

As can be seen in figure (2.4), DT starts with a root node that represents the entire dataset. The algorithm selects a feature and a corresponding threshold to split the data into two subsets (sub-trees) in a way that optimally separates the target classes. The final outcome node (leaf node) is beyond which no further separation of trees is possible. The goal is to minimize impurity or maximize information gain with each split. Common impurity measures include Gini impurity and entropy. Information

gain as the name suggests calculates the amount of information that is provided by a feature regarding the class. The node is split and the tree is constructed on the basis of information gain values. The DT algorithm maximizes the information gain function besides splitting the node/attribute possessing the greatest amount of information, first. Information gain is mathematically denoted as:

$$\text{Information Gain} = \text{Impurity}(\text{parent node}) - \sum_{i=1}^{K} \frac{N_i}{N} \text{Impurity}(\text{child node}_i) \quad (2.1)$$

where N_i is the number of samples in child node i and N is the total number of samples in the parent node. To address bias toward attributes with many values, the gain ratio normalizes information gain by considering the number of values an attribute can take.

Entropy measures the level of disorder or impurity in a node. Mathematically, for a node with K classes, entropy is calculated as:

$$\text{Entropy } (\text{ node }) = -\sum_{i=1}^{K} p_i \log_2 (p_i) \quad (2.2)$$

where p_i represents the proportion of samples belonging to class i in the node. Information gain quantifies the reduction in impurity achieved by a split. It is calculated as the difference between the impurity of the parent node and the weighted average impurity of the child nodes.

The Gini impurity measures the likelihood of a randomly chosen data point being misclassified. Mathematically, for a node with K classes, Gini impurity is calculated as:

$$\text{Gini}(\text{ node }) = 1 - \sum_{i=1}^{K} (p_i)^2 \quad (2.3)$$

where p_i represents the proportion of samples belonging to class i in the node.

Once the initial split is made, the process is repeated recursively for each subset. The algorithm selects the best feature and threshold for each subset, creating child nodes and further splitting the data. The tree continues to grow until a stopping

criterion is met, such as a maximum tree depth, a minimum number of samples in a leaf node, or when no further improvement in impurity can be achieved. When a stopping criterion is reached, the terminal nodes of the tree are called leaf nodes or decision nodes. These nodes contain the class label that corresponds to the majority class of the samples in that leaf. To make a prediction, a data point is passed through the decision tree by evaluating the input features at each node and following the path down the tree based on the feature values. When the data point reaches a leaf node, the class label associated with that leaf is the predicted class.

Some of the most well-known DT classifiers and algorithms are ID3 (Iterative Dichotomiser 3), C4.5, and C5.0. ID3 is one of the earliest DT algorithms developed by Ross Quinlan [27]. It uses entropy and information gain as splitting criteria. C4.5 is an extension of ID3, also developed by Ross Quinlan [28]. It can handle both categorical and numerical attributes and uses gain ratio as a splitting criterion. C5.0 is an improved version of C4.5 which is significantly faster and more accurate than C4.5 [29]. It offers enhanced performance and support for more data types. Algorithm 9 in appendix A.1.3 outlines the high-level steps for constructing a decision tree using a given splitting criterion.

In simple terms, the DTs make several vertical and horizontal cuts on the data domain, and thus they are highly suitable for multiclass classification as well. DTs are interpretable, and their visual representation makes them easy to understand and explain. However, they can be prone to overfitting on noisy data. Techniques like pruning and ensemble methods like Random Forests and Gradient Boosting are often used to mitigate these issues and improve classification accuracy.

Random Forests Ensemble learning is an ML technique that harnesses the collective intelligence of multiple individual predictors, such as classifiers or regressors, to achieve more accurate and robust predictions than what can be obtained from the best individual predictor alone. In ensemble learning, a group of such predictors forms an ensemble, and the process of combining their predictions is referred to as aggregation. Ensemble methods are designed to leverage the diversity and complementary strengths of these individual predictors, resulting in enhanced predictive performance and increased reliability. The fundamental idea behind ensemble learning is that by combining the insights and perspectives of multiple models, it becomes

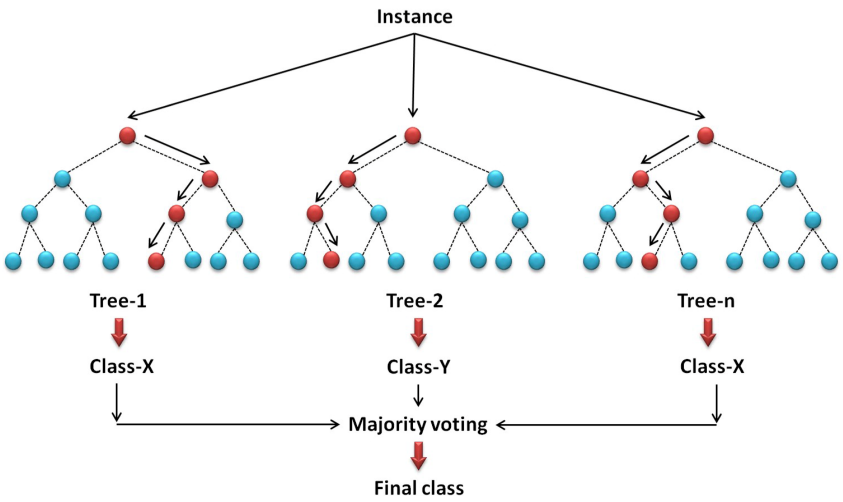


Fig. 2.5 Schematic diagram of a Random Forest.

possible to mitigate errors, reduce over-fitting, and improve the overall quality of predictions for a wide range of ML tasks.

Random Forest (RF) is an ensemble learning method that builds multiple DT during training and combines their predictions to improve accuracy and robustness. It is widely used for classification tasks and is based on the concept of bagging (Bootstrap Aggregating) and feature randomness. RF employs DTs as base learners. Each DT is trained on a bootstrap sample (randomly selected with replacement) from the original dataset. Bagging involves creating multiple bootstrap samples from the training data, each of which is used to train a separate DT. By training on different subsets of the data, RF reduces the risk of over-fitting and increases model robustness. RF introduces an additional layer of randomness by considering only a random subset of features (attributes) at each split in each DT. This helps to decorate the trees and ensures that each tree is trained on different subsets of features, reducing the chances of selecting the same dominant features in every tree. Figure 2.5 illustrates a simplified representation of a random forest consisting of three DT. In this ensemble method, the final class label for a given instance is determined through majority voting where each individual tree within the forest predicts a class label for the instance, and the most commonly predicted class label is chosen as the final output. Algorithm 15 in appendix A.1.4 represents a general overview of the RF learning methods. In practice, RF implementations may include additional optimizations, handling of different data types, and hyperparameter tuning.

K-Nearest Neighbors (KNN) KNN is a non-parametric classification algorithm that classifies instances based on their proximity to other instances in the feature space. In KNN, the K represents the number of nearest neighbors considered for classification. When a new instance needs to be classified, the algorithm calculates the distance between the new instance and all instances in the training data X. The K nearest neighbors are selected, and the majority class among these neighbors is assigned to the new instance. KNN is a simple and intuitive algorithm that can handle both binary and multi-class classification problems. It does not require any training process as it stores the entire training dataset (see algorithm 8 in appendix A.1.5).

The k-nearest neighbors are selected among instances of the training set where closeness is defined in terms of distance or similarity measures. Considering two instances $p(p_1, p_2, ..., p_d)$ and $q = (q_1, q_2, ..., q_d)$, some of the distance measures are as follows:

- **Manhattan Distance** (L_1 norm)

$$\mathrm{L}_1(p,q) = |p_1 - q_1| + |p_2 - q_2| + \ldots + |p_d - q_d|$$

  The sum of the absolute differences of the coordinates p and q.

- **Euclidean Distance** (L_2 norm)

$$\mathrm{L}_2(p,q) = \left((p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_d - q_d)^2\right)^{1/2}$$

  The length of the line segment connecting points p and q.

- **Supremum Distance** (L_∞ norm or L_{max} norm)

$$\mathrm{L}_\infty(p,q) = \max\{|p_1 - q_1|, |p_2 - q_2|, \ldots, |p_d - q_d|\}$$

  The maximum difference between any attributes of objects p and q.

- **Minkowski Distance** (Generalization of L_p-distance)

$$\mathrm{L}_p(p,q) = (|p_1 - q_1|^p + |p_2 - q_2|^p + \ldots + |p_d - q_d|^p)^{1/p}$$

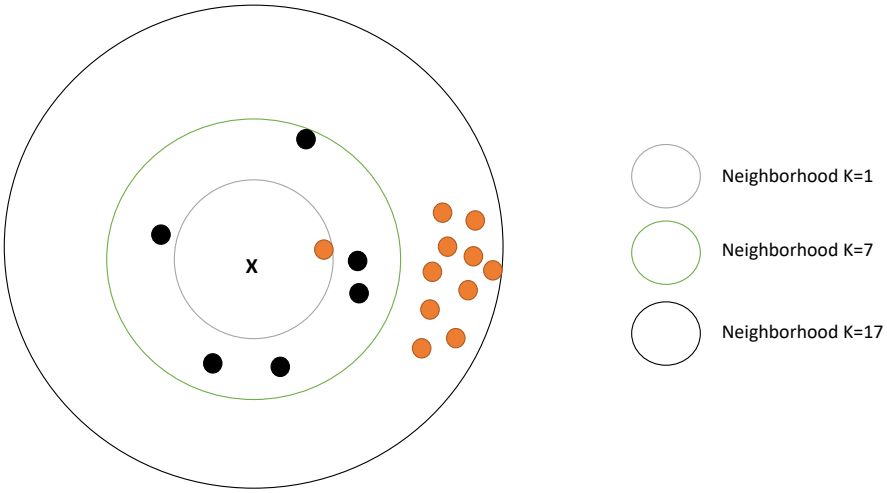


Fig. 2.6 The effect of K in KNN classification.

In Figure (2.6), we find the class of an unknown instance x based on its neighbors. When k equals 1, the class is determined by the closest neighbor. In the more general scenario with k greater than 1, we use *majority voting*. This means we consider the class labels of all neighbors, and they each have an equal contribution to the classification. But in *majority voting* the algorithm is very sensitive to the choice of K. Alternatively, *weighted voting* is another option, where each neighbor contributes to the classification based on its distance d from the unknown instance x (e.g., a possible weight factor: $w = 1/\operatorname{dist}(x,d)^2$)

KNN is robust to noise and can capture local patterns in the data. However, it can be sensitive to the choice of the value of K, and the performance may degrade with high-dimensional data or imbalanced datasets. Small values of K increase sensitivity to outliers, large values of K lead to the inclusion of many objects from other classes in the neighborhood, and optimal values of K tend to yield the highest classification accuracy.

The KNN algorithm is often referred to as a "lazy learner" or "lazy classifier" because it does not build an explicit model during the training phase. Instead, it memorizes the training data and makes predictions based on the nearest neighbors at the time of prediction. This makes it different from "eager learners" or "eager

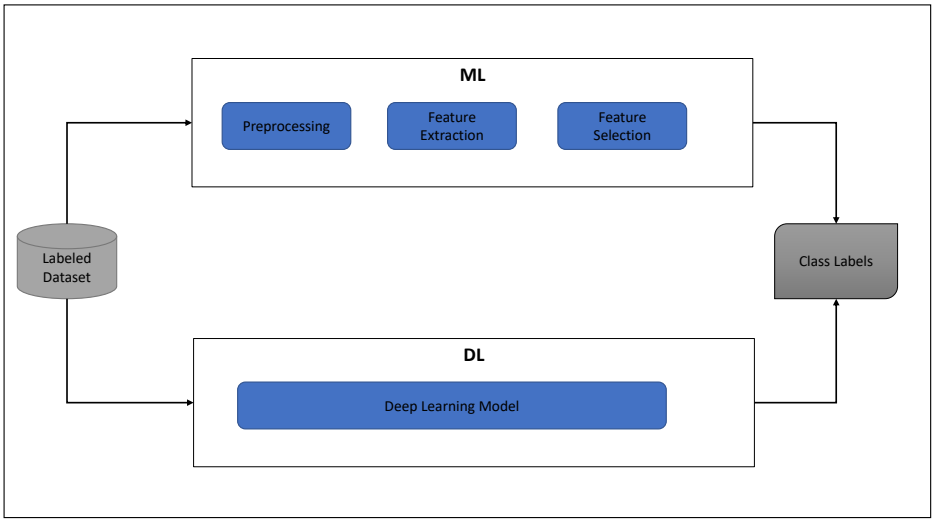


Fig. 2.7 The difference between deep learning and traditional machine learning (adapted from [4]).

classifiers" like decision trees or neural networks, which build a model based on the training data and use that model to make predictions. KNN's laziness allows it to adapt to changes in the data dynamically but may require more computational resources during prediction.

Deep Learning Classification Neural networks, specifically deep learning models, have gained significant popularity in recent years due to their ability to learn complex patterns and relationships in data. Neural networks consist of multiple layers of interconnected nodes (neurons) that process and transform the input data. They can handle both classification and regression tasks. The distinction between ML and DL is illustrated in Figure 2.7. In DL models, there is an automated process for preprocessing, feature extraction, and selection, enhancing their ability to learn and adapt effectively.

Three common types of deep learning classification models are:

- **Feedforward Neural Networks (FFNs):** also known as Artificial Neural Networks (ANN) and MultiLayer Perceptrons (MLPs), consists of layers of interconnected nodes, including an input layer, hidden layers, and an output

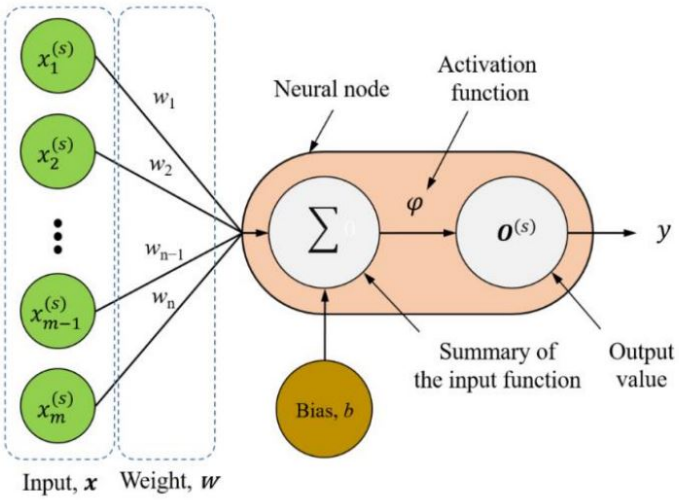


Fig. 2.8 Single-neuron perceptron model [5].

layer. Information flows in one direction, from input to output. It is a fully -connected network, meaning each neuron in one layer is connected to all neurons in the subsequent layer. In an FFN, the input layer first receives and normalizes the input data. The hidden layers, which can vary in quantity, analyze these input signals. The output layer then makes decisions or predictions based on the processed data. Figure 2.8 illustrates a model of a single-neuron perceptron, where an activation function $y = \varphi(xw+b)$ is applied to map the output value y from the summation function $((xw+b))$ where x, w, b, and y represent the input vector, weighting vector, bias, and output value, respectively [30]. Then Figure 2.9 illustrates the structure of the MLP (ANN or FFN) model. The details regarding the computation of output layers in the MLP can be found in Appendix A.1.5.

- **Recurrent Neural Networks (RNNs):** The RNN category includes gated recurrent units (GRUs) and long short-term memory (LSTM) approaches to handle sequential data. They contain loops that allow information to persist over time, making them suitable for tasks where previous context matters. A typical unfolded RNN diagram for a particular input sequence is shown in Figure 2.10.

- **Convolutional Neural Networks (CNNs):** CNNs are tailored for grid-like data, particularly images. They consist of convolutional layers for feature

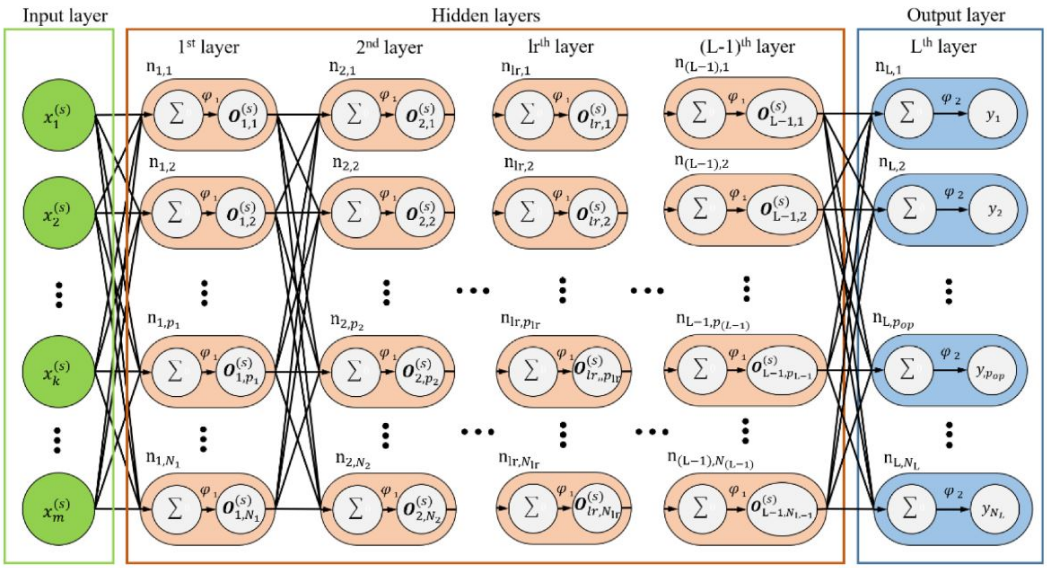


Fig. 2.9 Structure of the multilayer perceptron (MLP)[5].

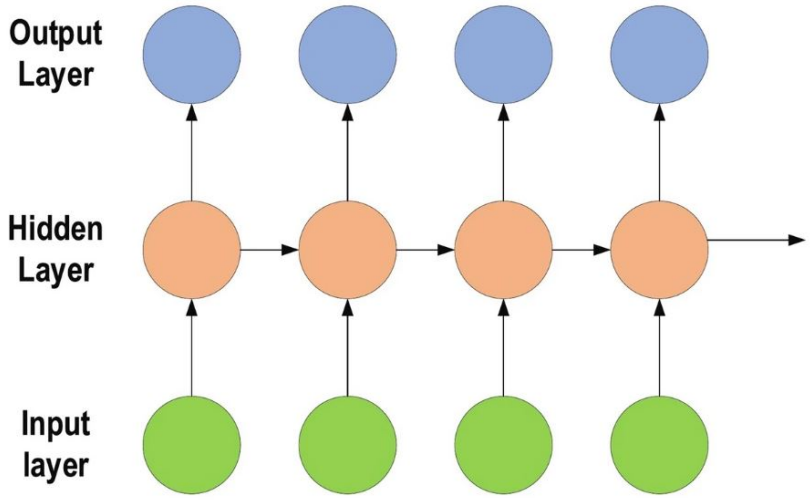


Fig. 2.10 Typical unfolded RNN diagram [4].

extraction and pooling layers for down-sampling. An example of CNN architecture for image classification is illustrated in Figure 2.11.

The selection of the appropriate deep learning classification model depends on the nature of the data and the specific requirements of the task. It is common to experiment with different architectures and techniques to achieve optimal results. In

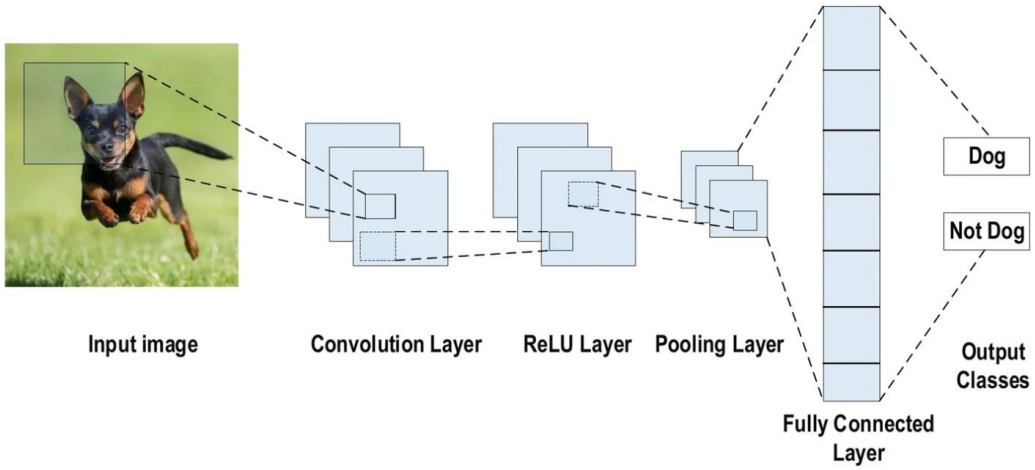


Fig. 2.11 An example of CNN architecture for image classifcation[4].

complex applications, hybrid models may be used, combining elements of FNNs, RNNs, and CNNs to leverage their respective strengths.

- Regression Methods

Regression models are used for predicting continuous outcomes, and they are well-suited for systems that yield such continuous responses [23]. We can represent the continuous response variable Y of a system by establishing a linear relationship between X and the domain variable x. Let's consider a straightforward two-dimensional parametric model for these variables, as outlined below:

$$Y = aX \tag{2.4}$$

If the actual responses are y, and the domain values are x, then we can define a nonlinear error factor as follows:

$$E = (y - ax)^2 \tag{2.5}$$

The continuous property of the variables x and y allows the application of derivatives for minimization. This holds true even when the model is high-dimensional, represented as a vector model as follows:

$$E = (y - Ax)^2 \tag{2.6}$$

When we have both a dataset and its associated response values, we can estimate the parameter (matrix) A by minimizing errors, and thus we can have the model defined in Equation (2.4) as a predictive model.

In the context of failure detection and diagnosis in manufacturing lines, regression models can be utilized to estimate the severity or magnitude of a failure based on the available input features. This can provide valuable information for maintenance planning and resource allocation. By learning from labeled data that includes information about the severity of failures, regression models can capture the underlying patterns and relationships between the input features and the associated failure severity values. Regression models commonly used in this domain include Linear Regression, Decision Tree Regression, Random Forest Regression, Gradient Boosting Regression, Support Vector Regression (SVR), and Deep Learning Regression.

- **Linear Regression:** Linear regression is a fundamental regression algorithm that models the relationship between the input features and the target variable using a linear equation. It aims to minimize the difference between the predicted and actual target values by adjusting the coefficients. Linear regression is interpretable and computationally efficient. However, it assumes a linear relationship between the features and the target variable, which may not always hold in complex manufacturing line scenarios.

- **Decision Tree Regression:** DT regression is a regression algorithm that utilizes a tree-like model to make predictions based on the input features. Similar to classification decision trees, DT regression recursively splits the data based on different feature thresholds to create branches and leaf nodes representing different predicted values. The predicted value for a new instance is the average of the target values in the leaf node that the instance belongs to. DT regression can handle both numerical and categorical features and can capture nonlinear relationships in the data.

- **Random Forest (RF) Regression:** RF regression is an ensemble learning method that combines multiple decision trees for regression tasks. Each DT

in the random forest is trained on a different subset of the data and features. The predicted values from individual trees are aggregated to produce the final prediction. RF regression provides improved accuracy and robustness by reducing overfitting and capturing diverse patterns in the data. It can handle high-dimensional data, handle missing values, and accommodate nonlinear relationships.

- **Gradient Boosting Regression:** Gradient boosting regression is another ensemble learning technique that combines weak regression models into a strong predictive model. It sequentially trains multiple regression models, where each subsequent model corrects the errors made by the previous models. The final prediction is obtained by summing the predictions from all the models. Gradient boosting regression is known for its ability to handle complex relationships and effectively model nonlinear patterns in the data.

- **Support Vector Regression (SVR):** SVR extends the principles of SVM to regression tasks. SVR aims to find a regression function that fits the data while minimizing deviations from a margin around the predicted values. It can handle linear and nonlinear regression problems by using different kernel functions. SVR is effective in handling datasets with a large number of features and outliers. It provides robustness against noise and can handle high-dimensional data.

- **Deep learning Regression:** Deep learning regression models are designed to predict continuous numerical values or real numbers, making them suitable for tasks where the output is a quantity rather than a category or class. These models have gained popularity for their ability to capture complex relationships in data, particularly in domains like image processing, natural language understanding, and time series forecasting.

In the previous section, we discussed conventional supervised learning techniques designed for scenarios where the entire dataset is available from the outset. However, practical industrial applications often involve data streaming in real time. To address this, adaptive or incremental learning methods are employed to effectively process streaming data. This is particularly relevant when dealing with classification and regression tasks on data that continuously arrives over time. In essence, stream classification (e.g. Hoeffding Trees) and regression (e.g. Online Linear Regression)

models play a crucial role in making real-time decisions, offering insights into stream data, and responding quickly to changing patterns and trends in dynamic, data-rich environments.

2.1.2 Related Works in Supervised Machine Learning for Failure Detection and Diagnosis

Industrial systems play a pivotal role in various sectors, such as manufacturing, metallurgy, and energy, contributing to their efficient and reliable operations. Hydraulic systems (HSs), as a specific type of industrial system, are widely employed and known for their ability to generate and control power through the use of pressurized fluids. Due to severe environments and unpredictable operating conditions, HSs are prone to failure. Furthermore, as the scope and automation levels of HS expand, their operational stability and reliability must be improved as well. By properly recognizing and addressing possible faults, damages may be reduced, maintenance costs can be minimized, and productivity can be increased [1]. Consequently, prognostic health management is a growing field, crucial to implement predictive maintenance in complex manufacturing systems. It is based on a number of condition-based monitoring (CBM) techniques, including vibration analysis, acoustic emissions analysis, and oil analysis, all of which can provide useful information on the health of monitored equipment in real-time [31].

As a result, CBM of HSs is becoming increasingly important in order to improve industrial applications and bring several benefits such as the reduction of machinery downtime and maintenance costs [32]. However, due to the frequent and costly shutdown problems, it is not convenient to predict the future of machinery damage using manual CBM systems. Conversely, various types of CBM indicators can be automatically derived from running machines to suitably identify their health in real-time. The basic idea commonly exploits the degradation trends that change over time due to shifting operational conditions: it is thus ideal to explore the correlation between changing operational conditions and degradation patterns over time [33]. Since the majority of today's industrial systems, as well as HSs, make use of sensors to collect measured data, healthy indicators (HIs) can be straightforwardly constructed from the measured data to characterize the healthy condition of devices

using automated methods based on signal processing, artificial intelligence, and other analytics approaches.

In the related literature, several automated techniques have been proposed to address the CBM problem in HSs. For instance, [32] analyzes a complex HS to find the correlation between features extracted from raw sensor data and fault types. Using a multivariate statistic, they identify the most significant features specific to a fault case where the fault characteristics of experimental data are known. Authors of [34] propose an intelligent system to accurately characterize the health of the same HS using streams of real-time sensors' data. Authors of [35–38] apply different fault detection methods based on conventional machine learning models to detect failures in HSs. Considering the nonlinear characteristics of the signals measured by multiple sensors in the HS system, a novel fault diagnosis method of the HS based on multi-source information fusion and fractal dimension is proposed by [39], where the fault pattern recognition of the system is realized by using the k-nearest neighbor technique and fuzzy clustering approach. The sampling rates of the sensors' data collected from the HS are always varied, and the coupling interaction between the components makes consistent data collection challenging. Authors of [1] and [40] apply various deep learning models to the HS sensors' data to overcome such a problem. The performance of Deep Learning (DL) FDD relies on mathematical plant models where the development of DL networks helps to extract high-level and abstract features from the data to enhance fault classification and regression results when effective feature representations are extracted [41].

In all the above-cited works, the proposed CBM systems are able to accurately monitor the working condition of the industrial devices, without being capable of detecting in advance the failure. In the related literature of HSs, the decision-making is postponed until the entire information has been gathered. Indeed, all the previous works do not predict the time of failure during a working cycle and do not prevent the system from working in the presence of possible severe faults. In fact, the time of a potential failure in a component, which may be identified via automated processing of available sensors' signals, is one of the most significant HIs in HSs. When new measurements arrive sequentially, it is required to gradually monitor the health status of an asset in these kinds of applications. Since the measures are not always regular and arrive over time, it is indeed critical to make a decision as soon as possible to optimize the performance of the system ([12]). As the HS runs over cycling operations, several relevant parameters may be monitored to observe any progressive

change leading to some potential failure. However, at this time, only a slight drop in running performance occurs, and it is not possible to prevent these faults through traditional fault diagnosis techniques. As a result, it is crucially needed to raise an early warning of fault according to the changing trend of working signals in HSs.

As mentioned before, industrial systems are complex and failure-prone systems that require constant monitoring and maintenance to prevent downtime and ensure safety. ML and Failure Analysis (FA) are two powerful tools that can be used to diagnose system failures and establish the root cause of failures. On the one hand, ML techniques can learn from historical data to diagnose the likelihood of future failures. For instance, several automated techniques have been proposed in the literature to solve fault diagnosis issues in industrial assets ([42, 43]), particularly HSs. These techniques are capable of dealing with various challenges in fault diagnosis, such as the non-linear nature of sensor signals, varied sampling rates, and coupling interactions between components. On the other hand, FA is a systematic approach to investigating the root cause(s) of a failure or malfunction in a system or component ([44]). The process of FA typically involves a combination of data analysis, physical inspection, and testing to identify and isolate the cause(s) of the failure. Fault tree analysis (FTA) is one of the most common techniques within the broader field of FA that can identify the root causes of failures and help engineers develop effective mitigation strategies as one of the reliability analysis tools. Several studies propose different methods for FA and maintenance planning of industrial systems: for instance, [45] propose a methodology for maintenance planning based on system-level prognostics and FA. Authors of [46] use FA to analyze failure behavior and component contributions on a coal mine's dragline for maintenance planning and cost reduction. Authors of [47] present a case study showing how their approach identifies events causing failures during the warranty period in a computer numerical control (CNC) turning center. Authors of [48] introduce a method to determine the likelihood of uncertain events by combining expert opinions with FTA for an HS case study. Authors of [49] analyze faults in complex systems to capture hierarchical causality between root causes and faults using an actuator system dataset.

While FTA is a powerful tool for analyzing the causes of system failures, it has limitations when it comes to dynamic systems. In such cases, statistical FA ([50–52]) may be more appropriate, as they can provide a more accurate picture of the industrial system behavior contributing to failure.

The previously reported literature review shows that existing works apply either ML or FA to diagnose and analyze the system failures, based on the dataset availability and system knowledge. In particular, ML techniques are typically employed for handling large and complex industrial datasets, whereas FA is employed for analyzing the failure behavior of individual components or systems. Moreover, FA is typically based on expert knowledge and requires a thorough understanding of the system under consideration, while ML algorithms can learn patterns and behaviors from large datasets without prior knowledge of the system.

2.2 Semi-Supervised Machine Learning Methods

Semi-supervised learning (SSL) is a type of ML approach that lies between supervised and unsupervised learning. In this paradigm, a combination of labeled and unlabeled data is used for training the model. The goal of SSL is to leverage the limited labeled data along with the abundant unlabeled data to improve the model's performance and generalization.

2.2.1 A Comprehensive Guide to Semi-Supervised Learning

SSL methods can be categorized into several approaches based on their underlying principles and techniques.

Self-training Methods

This is a simple and widely used SSL technique. It starts with a small labeled dataset and iteratively labels unlabeled data points using a trained model's predictions. The labeled dataset grows over time as can be seen in figure 2.12.

This approach is compatible with any supervised learning algorithm and typically performs well in the absence of outliers. Figure 2.13 illustrates this approach when the KNN is taken as the base classifier into consideration.

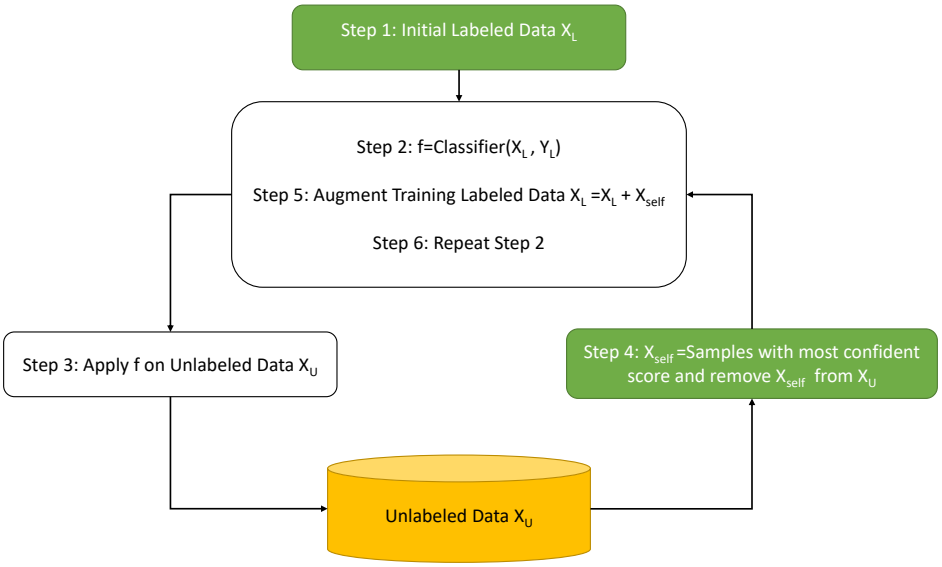


Fig. 2.12 The process of self-training SSL (adapted from [6]).

Co-training Methods

Co-training methods work with multiple views or features of the data. They train multiple classifiers, each on a different view, and exchange predictions on unlabeled data to improve classification performance. Similar to co-training, the multi-view SSL approach leverages multiple views of the data to improve the classification of unlabeled data points.

In co-training, the algorithm trains two or more classifiers, each using a different set of features or perspectives (often referred to as "views") of the data. However, there is a special twist: each classifier uses the predictions of the others on the unlabeled data to enhance its own learning process (see figure 2.14).

Graph-Based Methods

Graph-based Semi-Supervised Learning (GSSL) methods use the structure of the data, typically represented as a graph, to propagate labels from labeled to unlabeled data points. GSSL is a class of SSL techniques that holds great promise. GSSL capitalizes on the inherent structure of data through the use of graphs, aligning with the fundamental manifold assumption in SSL. In essence, GSSL methodologies

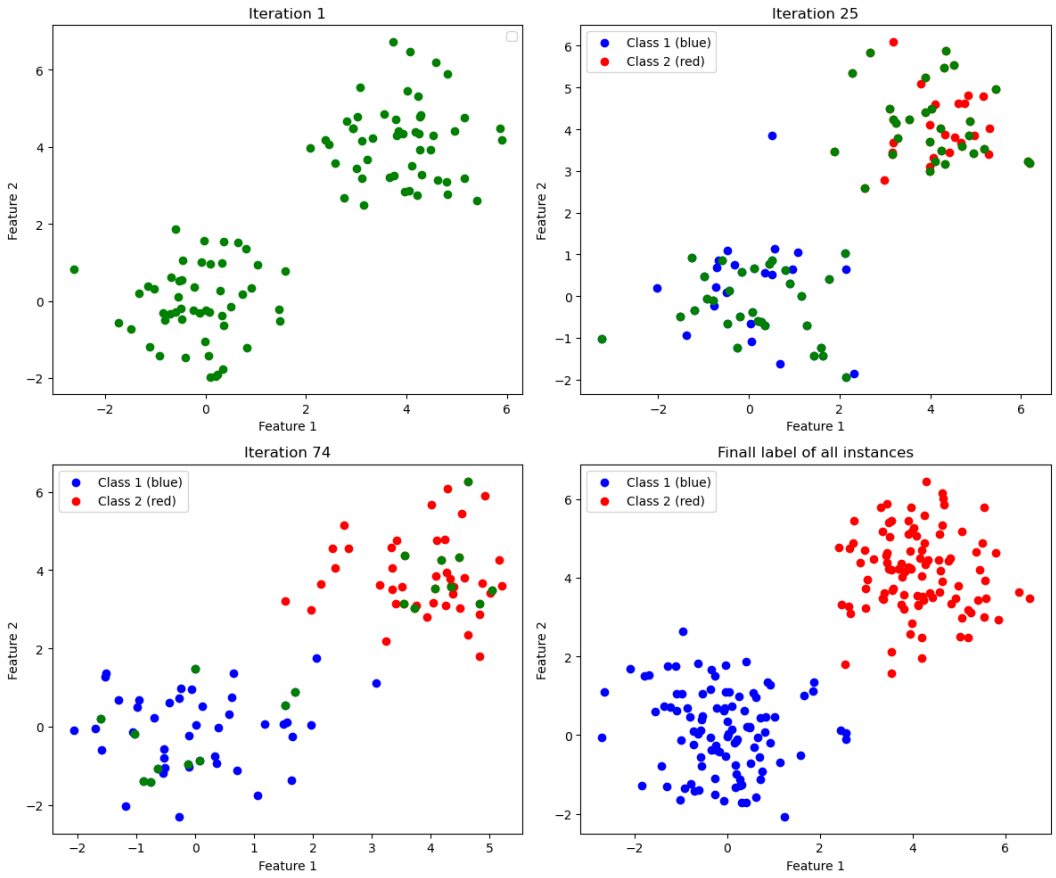


Fig. 2.13 self-training SSL based on KNN base classifier [6].

commence by creating a graph representation, where every sample is a node, and the connections, typically characterized by weighted edges, reflect the similarity between pairs of nodes. This approach to graph construction implies that nodes linked by high-weight edges tend to share the same labels, aligning with the manifold assumption that samples residing in close proximity within a lower-dimensional manifold should exhibit similar labels [7]. The primary two-step process in GSSL involves, first and foremost, constructing an appropriate graph structure along which the provided labels can be subsequently propagated. This fundamental procedure is illustrated in Figure 2.15.

GSSL algorithms can be divided into two categories namely transductive and inductive based on whether to predict data samples' labels out of training data. While the majority of GSSL methods adopt a transductive approach, there do exist

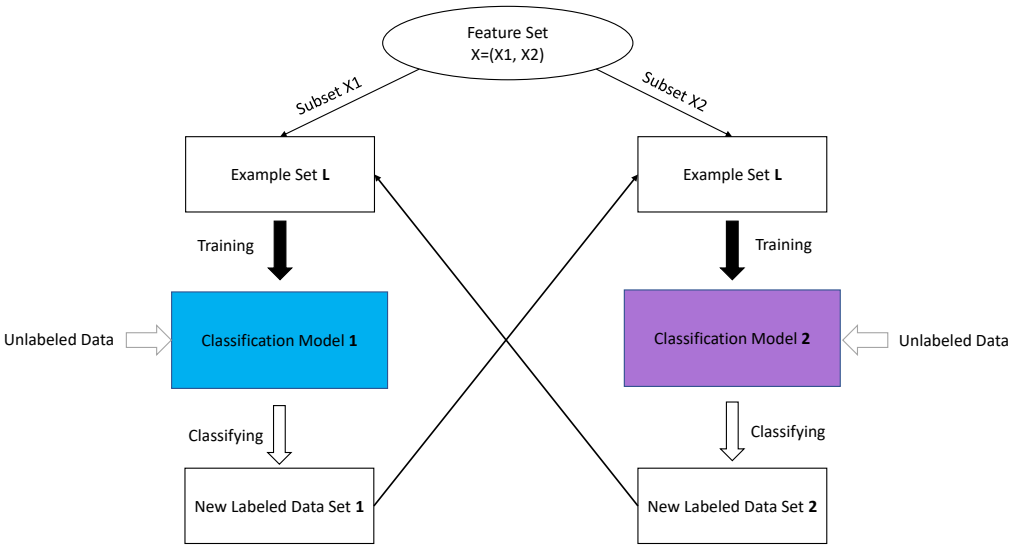


Fig. 2.14 Co-training approach.

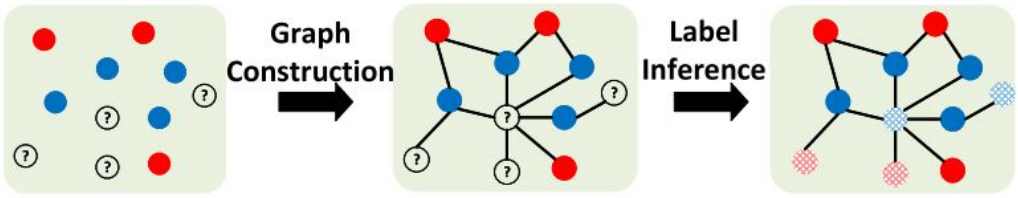


Fig. 2.15 Two-step procedure of GSSL. Here, red circles and blue circles denote the labeled positive and negative nodes, respectively. Circles with question marks represent the unlabeled nodes and the color in the shaded circles indicate their corresponding predicted label [7].

several inductive GSSL methods. Typically, in real-world scenarios, transductive SSL techniques tend to outperform their inductive counterparts when it comes to prediction accuracy. However, it is important to note that transductive methods often come with a trade-off in terms of computational cost, particularly in situations involving large-scale incremental learning. Figure 2.16 illustrates the distinction between these two approaches within the realm of GSSL."

In the context of FDD in manufacturing lines, SSL can be applied to enhance the accuracy and efficiency of the detection and diagnostic processes. By utilizing the

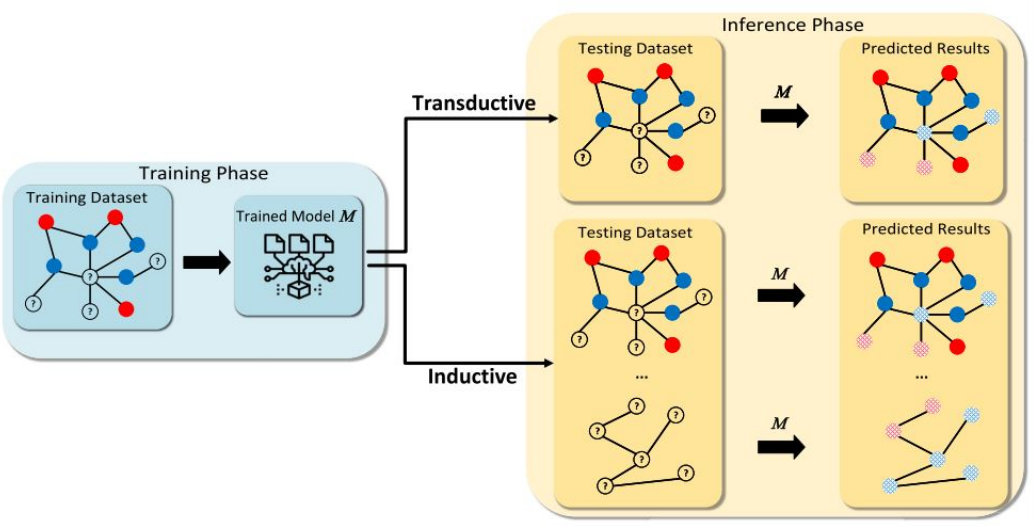


Fig. 2.16 Comparison between transductive and inductive settings in GSSL. The significant difference between them lies in the inference stage. For the transductive setting, only the labels of unlabeled nodes on the same graph in the training dataset need to be inferred. For an inductive setting, however, the trained model can also predict the label of unseen nodes on new graphs that do not exist in the training set. Here, red circles and blue circles denote the labeled positive and negative nodes, respectively. Circles with question marks represent the unlabeled nodes, and the color in the shaded circles indicates their corresponding predicted label [7].

available labeled data, which typically consists of instances with known failure states and the large amount of unlabeled data collected from normal operation conditions, SSL algorithms can effectively learn the underlying patterns and identify anomalies.

2.2.2 Related Works in Semi-Supervised Machine Learning for Failure Detection and Diagnosis

The use of SSL in FDD allows for the exploration of unlabeled data, which can reveal valuable insights into the normal behavior and operation of the manufacturing lines. By learning from both labeled and unlabeled data, the model can capture the underlying structure of the data distribution and make accurate predictions on unseen instances.

By incorporating SSL into the FDD process, manufacturing lines can benefit from enhanced accuracy and efficiency in detecting failures and diagnosing their

causes. This can contribute to reduced downtime, optimized maintenance strategies, and improved overall performance and productivity. However, it is important to note that the success of SSL relies heavily on the availability of reliable labeled data and the quality of the unlabeled data.

In the previous works, the proposed supervised models are able to accurately monitor the working condition of the industrial devices, only in the case the entirely labeled data is available. Such an assumption is indeed a big challenge in industrial applications.

However, as the mechanical system runs over cycling operations, several relevant parameters may be monitored to observe any progressive change leading to some potential failures. In many tasks, there is a paucity of labeled data. The labels may be difficult to obtain because they require human annotators, special devices, or expensive and slow experiments. In real-world scenarios, faults are usually fixed within a short period of time while a practical FDD method is demanded to detect and diagnose various faults with the smallest possible number of faulty samples available. In this context, where label scarcity is the common problem, inductive and transductive SSL has a significant practical value. On the one hand, inductive SSL assumes that the unlabeled data is generated from the same distribution as the labeled data and tries to learn a general model that can be applied to new unseen examples. Transductive SSL, on the other hand, makes use of the specific information contained in the unlabeled data that is related to the specific examples in the labeled set. In other words, transductive SSL makes predictions for the test data based on both labeled and unlabeled data rather than trying to generalize to new examples.

For instance, [53] develops an inductive semi-supervised FDD method with fuzzy rules. The developed approach was applied to the FDD of an industrial actuator. All failure modes can be well identified without any given prior knowledge. [54] combines the Gaussian mixture model and support vector machine (SVM) to form an inductive SSL model for fault diagnosis of chemical processes. Experimental results show that the SSL enhanced the traditional classification ability and is suitably applicable to the complex industrial environment. As the amount of labeled data remains fixed in inductive SSL, it may not be sufficient to capture the full complexity of the problem. In fact, transductive SSL typically outperforms inductive SSL since it can use unlabeled data when training the model [55]. [56] used SVM as a base classifier for their proposed transductive SSL framework. The proposed framework

iteratively adds confidently labeled testing samples to enrich the training pool for early fault detection, achieving comparable performance to classic supervised FDD methods with a small amount of faulty training data.

[57] proposes a transductive SSL approach for fault detection in Additive Manufacturing products, leveraging both labeled and unlabeled data to make predictions about component quality without individual certification. [58] proposes a new transductive semi-supervised deep generative approach that incorporates CNNs to model the complex relationship between high-dimensional process data and process status, considering temporal and intervariable correlations. In [59], a three-stage transductive SSL approach is proposed for intelligent bearing fault diagnosis, which outperforms existing methods when limited labeled data is available. [60] proposes LSTM-LAE, a novel transductive method for fault diagnosis that combines LSTM and LAE techniques to extract temporal features and improve performance by leveraging unlabeled data. [61] discusses graph-based SSL, a useful subset of transductive SSL that labels partially observed datasets using information from both labeled and unlabeled data points in a network, which can be especially helpful in situations where labeled data is scarce, time-consuming, or expensive.

Graph-based SSL is a subset of transductive SSL in which the aim is to label a partially observed data set by using information from both labeled and unlabeled data points in a network [61]. Similar to the other SSL methods, it can be particularly useful in situations where labeled data is scarce, or where labeling the data is time-consuming or expensive. However, graph-based SSL has some advantages over other transductive SSL methods. For example, graph-based SSL can naturally handle high-dimensional data, capture complex dependencies between features, and incorporate prior knowledge in the form of graphs or networks. Authors in [62] propose an SSL model for fault detection and classification using a combination of expert knowledge and unlabeled data based on graph regularization and convex programming optimization function, which helps in accurately detecting and classifying faults in the monitoring data.

Methods that rely on expert knowledge are limited by the amount and quality of that knowledge, and may not be able to capture all of the relevant information in the data. Graph-based SSL with label propagation (LP) relies not only on the limited and potentially biased knowledge of human experts, thus allowing for a more comprehensive and effective SSL approach in many applications.

Moreover, using LP in graph-based SSL can be a successful method for eliminating redundant labels. Graph-based SSL with LP can help to improve the accuracy and performance of algorithms in situations where there are irrelevant or noisy data. By making use of both labeled and unlabeled data, this technique can lead to a more precise and refined dataset. Despite being effective in a variety of scenarios, research on the application of graph-based SSL to FDD industrial systems is still lagging behind. In the context of FDD, the use of SSL for maintenance planning can be particularly beneficial, as it can help to enrich the dataset with additional data and consequently can be an effective approach for predicting corrective and preventive maintenance actions for the reliability and availability of industrial systems.

2.3 Unsupervised Machine Learning Methods

Unsupervised learning is an ML approach where the algorithm learns from unlabeled data without explicit output labels. The goal of unsupervised learning is to discover hidden patterns, structures, or relationships within the data.

2.3.1 A Comprehensive Guide to Unsupervised Learning

There are several types of commonly used unsupervised learning methods:

Clustering

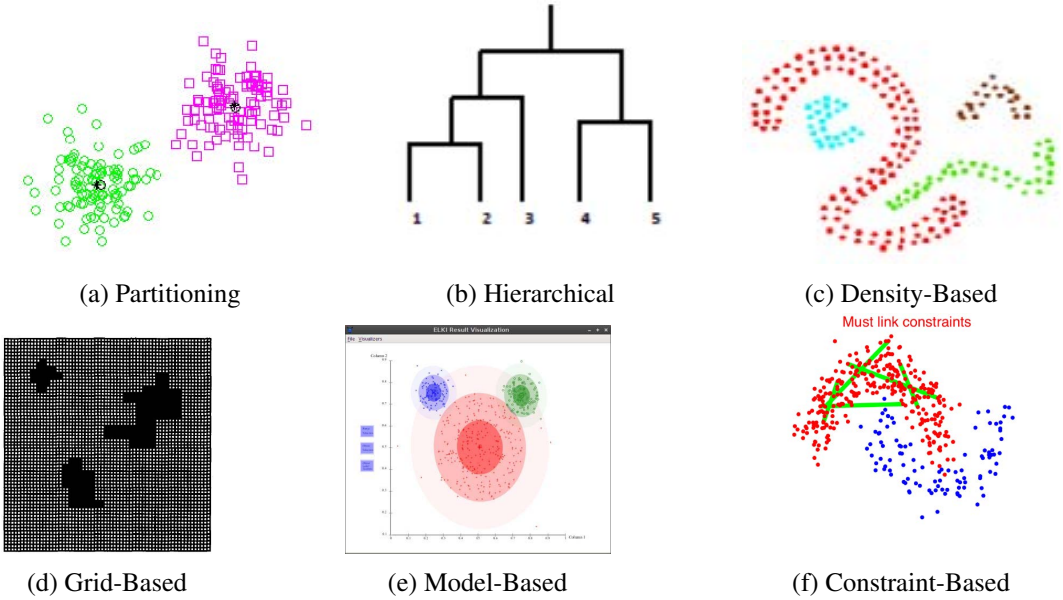
A clustering [63] approach, in the context of ML and data analysis, refers to the use of various methods to group similar data points together into clusters based on shared characteristics or similarity. These approaches are categorized into several major types (see figure 2.17):

- **Partitioning Approaches:** Partitioning methods aim to divide the dataset into distinct partitions or clusters. Examples include k-means (algorithm 10 in appendix A.2.1) and k-medoids. These algorithms iteratively refine cluster assignments to minimize an objective function, often based on distances between data points.

- **Hierarchical Approaches:** Hierarchical clustering methods create a tree-like structure of nested clusters. Agglomerative algorithms start with individual data points as clusters and merge them progressively, while divisive methods begin with a single cluster and split it into smaller subclusters.
- **Density-Based Approaches:** Density-based methods identify clusters as regions of high data point density separated by areas of lower density. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a well-known example that groups points together if they are sufficiently dense and have nearby neighbors.
- **Grid-Based Approaches:** Grid-based clustering methods divide the data space into a grid of cells and assign data points to these cells. Clusters are then formed by grouping adjacent cells with a sufficient number of points. STING (Statistical Information Grid) is an example of a grid-based approach.
- **Model-Based Approaches:** Model-based clustering assumes that the data is generated by a statistical model. These methods search for the best-fitting model parameters and use them to determine cluster assignments. The Gaussian Mixture Model (GMM) is a classic model-based clustering technique.
- **Constraint-Based Approaches:** Constraint-based clustering allows the incorporation of prior knowledge or constraints into the clustering process. Constraints can be used to guide the clustering algorithm by specifying which data points should or should not belong to the same cluster.

These categorizations provide a framework for understanding the diverse range of clustering techniques available, each with its own strengths and suitability for different types of data and clustering tasks. Researchers and practitioners choose clustering methods based on the specific characteristics of their data and the objectives of their analysis.

In the preceding section, we delved into traditional clustering methods, which assume that we have all the data available upfront. However, real-world industrial scenarios often involve data that arrives gradually over time. To effectively handle such stream data, we turn to adaptive, incremental, or stream clustering methods. These techniques are specifically designed to accommodate the dynamic nature of incoming data, enabling us to gain valuable insights and patterns as information



(a) Partitioning (b) Hierarchical (c) Density-Based

(d) Grid-Based (e) Model-Based (f) Constraint-Based

Fig. 2.17 Clustering Methods

unfolds in real-time. Following the traditional taxonomy of clustering algorithms, stream clustering methods can be categorized into four main categories [64]:

- Partitioning methods: Clustream, SWClustering and StreamKM++.
- Density-based methods: DenStream, rDenStream, C-DenStream, SDStream, HDDStream, MuDi-Stream, and HDenStream
- Grid-based methods: D-Stream, DDStream, MR-Stream, DENGRIS and PKS-Stream.
- Model-based methods: SWEM

Anomaly Detection

Anomaly detection [65, 66] algorithms aim to identify instances that deviate significantly from the normal patterns or behaviors observed in the data. These algorithms can detect rare events or outliers that may indicate the presence of failures. For instance, in Figure 2.19, Points o_1 and o_2, as well as points in the region O_3 lie outside the bounds of the normal regions and are, thus, point anomalies since they deviate from normal data points [67].

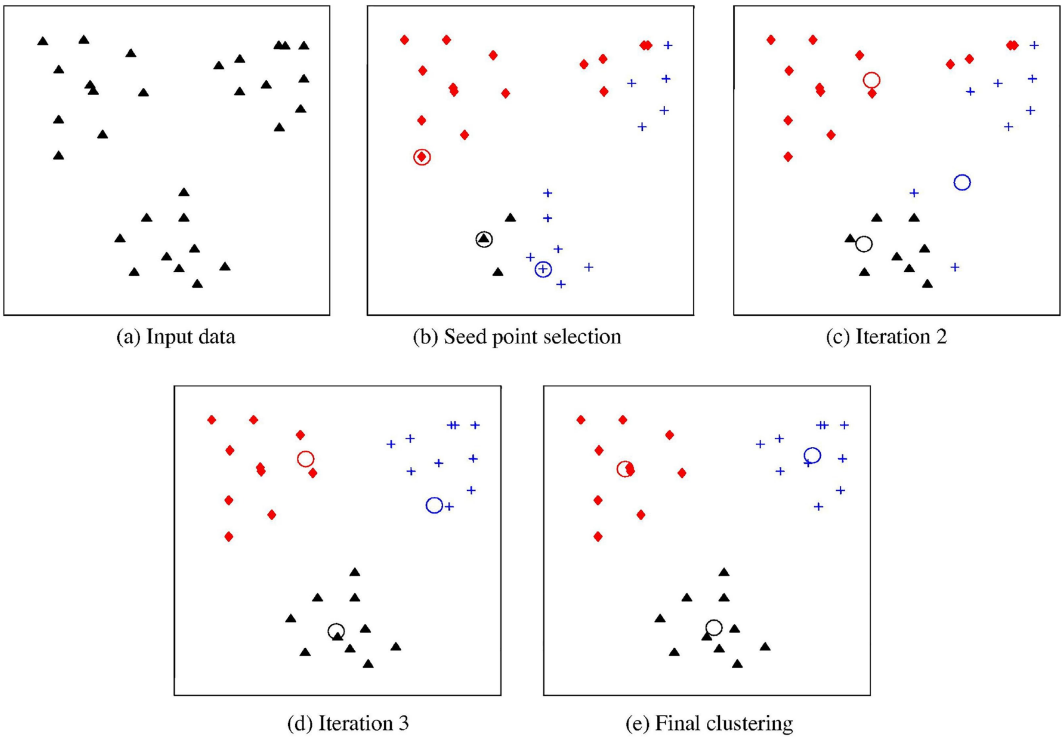


Fig. 2.18 Illustration of K-means algorithm. (a) Two-dimensional input data with three clusters; (b) three seed points selected as cluster centers and initial assignment of the data points to clusters; (c) and (d) intermediate iterations updating cluster labels and their centers; (e) final clustering obtained by K-means algorithm at convergence [8]

Anomaly Detection encompasses various techniques to identify unusual patterns or outliers within datasets. Common methods include statistical approaches like Z-Score and Median Absolute Deviation (MAD), which assess data points' deviations from statistical norms; density-based methods like DBSCAN and Local Outlier Factor (LOF), which pinpoint anomalies in low-density regions; distance-based techniques like K-Nearest Neighbors and One-Class SVM, which gauge data points' distances from their neighbors; clustering methods like K-Means and Hierarchical Clustering, which flag outliers as data points that do not fit into clusters; ensemble methods like Isolation Forest, which isolate anomalies by random feature splits; autoencoders that use neural networks to detect anomalies based on reconstruction errors; histogram-based approaches dividing feature space into cells; and specialized techniques for time-series data, such as AutoRegressive Integrated Moving Average (ARIMA) and Prophet, which predict values and identify anomalies by comparing

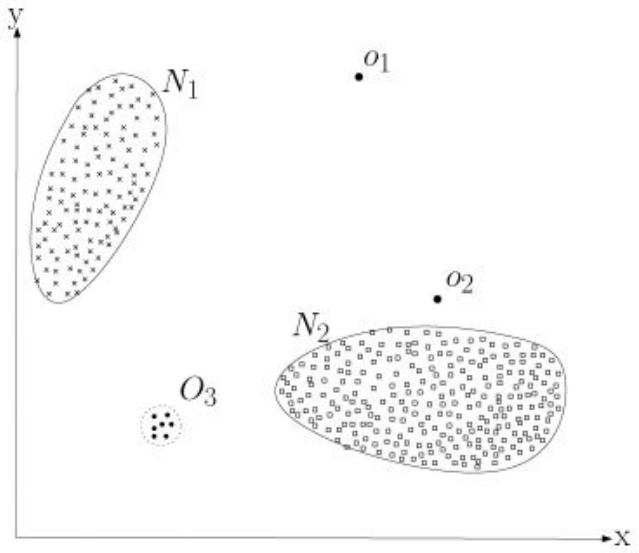


Fig. 2.19 A simple example of anomalies in a two-dimensional data set [8]

actual and predicted values. The choice of method depends on data characteristics and the nature of anomalies, with domain expertise guiding the selection process.

Dimensionality Reduction

As the dimensionality of the dataset or the number of features grows, the data volume needed for achieving a statistically meaningful outcome also rises exponentially. This presents challenges like overfitting, prolonged computation times, and diminished precision in machine learning models, commonly referred to as the issues stemming from the curse of dimensionality when dealing with high-dimensional data. Dimensionality reduction techniques reduce the dimensionality of the data by extracting a lower-dimensional representation while preserving important information. This can help visualize the data, identify relevant features, and simplify subsequent analysis and interpretation. To address the curse of dimensionality, feature engineering techniques are used which include feature selection and feature extraction:

- **Feature Selection:** In this method, you choose a subset of the original features and discard the rest. The selection can be based on statistical techniques, domain knowledge, or other criteria. Feature selection methods include:

Filter Methods: These methods use statistical metrics like correlation or mutual information to select the most relevant features.

Wrapper Methods: These methods use a machine learning model's performance to select features. They involve training and evaluating the model with different feature subsets.

Embedded Methods: Feature selection is integrated into the learning algorithm itself. Techniques like L1 regularization for linear models automatically select relevant features.

- **Feature Extraction:** Feature extraction transforms the original features into a new set of features. The idea is to project the data into a lower-dimensional space while preserving as much relevant information as possible. Popular techniques for feature extraction include:

  Principal Component Analysis (PCA): PCA is a linear transformation method that finds orthogonal axes (principal components) where the variance of the data is maximized. It is excellent for reducing dimensions in a decorrelated manner by finding a new set of variables, smaller than the original set of variables, retaining most of the sample's information, and useful for the regression and classification of data.

  Indeed, PCA is designed to capture the highest variance present in the data. The principal components, which are linear combinations of original variables in the dataset, are organized by importance in decreasing order. The cumulative variance encompassed by all principal components equals the total variance found in the initial dataset. As depicted in Figure 2.20, the first principal component (PCA_1) captures the most substantial data variation. Subsequently, the second principal component (PCA_2) seizes the maximum variance that is orthogonal to the first principal component, and so forth [9].

  Linear Discriminant Analysis (LDA): LDA is often used for supervised dimensionality reduction and is especially useful in classification tasks [68]. It aims to maximize the separability between classes. This is achieved by identifying a collection of linear discriminants that optimize the ratio of variance

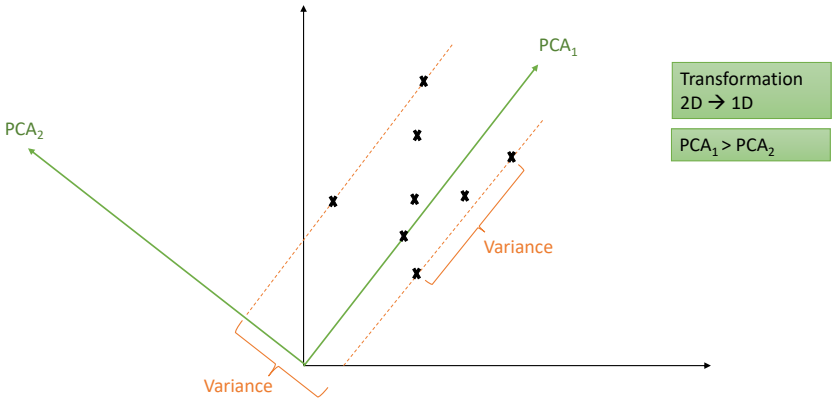


Fig. 2.20 Principal Component Analysis (PCA) (adapted from [9]).

between classes to variance within classes. In simpler terms, it determines the feature space directions that most effectively distinguish various data classes from one another.

Consider two distinct sets of data points representing different classes that I aim to classify. As depicted in the provided 2D plot in figure 2.21, when these data points are visualized in a two-dimensional space, there is no single straight line that can entirely separate the two data point classes. In such scenarios, LDA comes into play. LDA transforms the 2D graph into a 1D representation, effectively enhancing the separability between the two classes.

In figure 2.21, it can be seen that a new axis (in red) is generated and plotted in the 2D graph such that it maximizes the distance between the means of the two classes and minimizes the variation within each class. In simple terms, this newly generated axis increases the separation between the data points of the two classes. After generating this new axis using the above-mentioned criteria, all the data points of the classes are plotted on this new axis and are shown in figure 2.22 [10].

The choice between feature selection and feature extraction depends on the specific problem, the nature of the data, and the computational resources avail-

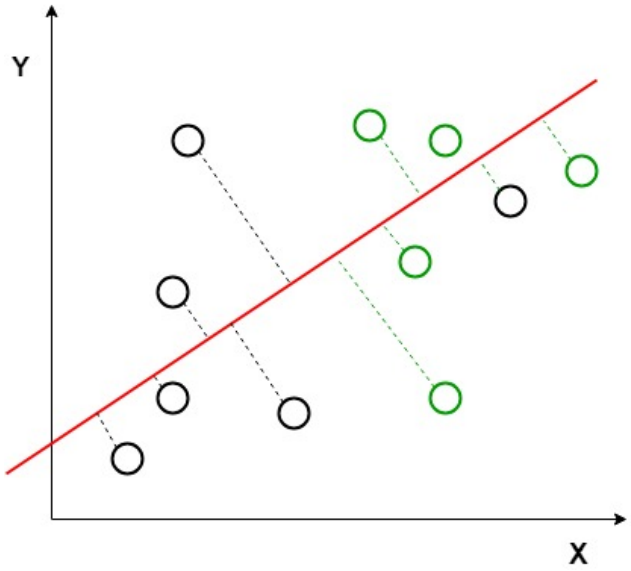


Fig. 2.21 Two sets of data points belonging to two different classes that need to be classified [10].

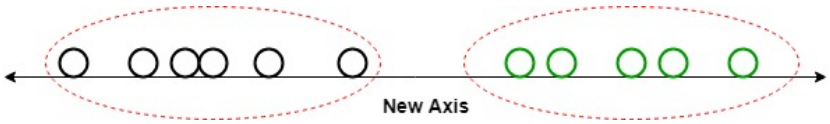


Fig. 2.22 Linear Discriminant Analysis (LDA) is used which reduces the 2D graph into a 1D graph to maximize the separability between the two classes [10].

able. Dimensionality reduction can help in cases where a dataset has too many features, some of which may be redundant or irrelevant, leading to improved model performance, reduced overfitting, and faster training times.

Association Rule Mining

Association rule mining techniques discover relationships and associations between different features or variables in the data. These associations can provide insights into the dependencies and interactions between variables, which may be indicative of failure-related patterns. The Apriori algorithm is a popular method for association rule mining. The Apriori algorithm operates on a database of transactions, where each transaction is a set of items. It is widely used for market basket analysis, which identifies item sets that are frequently purchased together. The algorithm employs

a bottom-up approach to discover frequent item sets and generate association rules (see Appendix A.2.2).

2.3.2 Related Works in Unsupervised Machine Learning for Failure Detection and Diagnosis

In the context of FDD in manufacturing lines, unsupervised learning methods can be applied to explore the data and identify anomalies or abnormal behavior without prior knowledge of specific failure instances. Unsupervised learning techniques can provide valuable insights into the normal operating conditions and help in detecting deviations from the expected behavior.

Unsupervised learning methods can be applied to analyze the historical data collected from manufacturing lines and identify abnormal behavior or patterns that may signal the presence of failures. By detecting anomalies or identifying distinct clusters within the data, unsupervised learning techniques enable proactive maintenance strategies, allowing for timely interventions to prevent further damage or downtime. It is worth noting that unsupervised learning alone may not provide explicit information about the specific types or causes of failures. However, it can complement other FDD approaches and assist in exploratory data analysis, feature selection, and gaining a deeper understanding of the data distribution, which can guide subsequent supervised or semi-supervised learning methods in failure diagnosis and decision-making processes.

Recently, as opposed to model-based techniques [69], data-driven methods have been widely used in industrial systems to detect possible faults under the realistic hypothesis that monitoring data is available [70–72]. For instance, Bayesian networks and the principal component analysis are respectively used in [73] and [74] as the core component of the fault diagnosis approach. However, within the ML context, unsupervised learning (e.g., *clustering*) is one of the most common techniques to interpret and transform raw data into information that can potentially create value in CBM applications. For instance, a clustering analysis successfully supports the identification of the fault severity level in the case of multi-degradation systems where distinguishing among different faults is difficult. Hence, several clustering-based automated techniques are proposed in the related literature to deal with CBM.

In [75] the normal operation of a selective laser melting machine – defined as a manufacturing process that uses a metal powder bed and thermal energy supplied by a computer-controlled and focused laser beam to build a work-piece – are compared against three faulty conditions using clustering methods in order to implement a CBM system and enable machine tools for predictive maintenance solutions. In standard clustering algorithms (e.g., K-means) the goal is to divide the original data set into non-intersecting groups such that the distances between the instances of the same group (i.e., intra-clusters) are smaller than the distances with respect to instances of other groups (i.e., inter-clusters). An interesting modification of the traditional K-means is represented by the Constrained K-means, where the learning process is not totally unsupervised. In addition to the original data set, some additional information about the desired clusters are provided: for example, information can be related, but not limited, to must-link and cannot-link constraints.

Nevertheless, all the traditional clustering methods (such as K-means and Constrained K-means) require the availability of the whole data set to build and run the ML models. In the current industrial scenarios, where sensors and Internet of Things devices generate dynamical process data at high speed, producing actionable insights at the right time is challenging [76]. More specifically, when real-time monitoring data arrive over time, conventional techniques used for static data sets are not suitable for online fault detection based on data stream (DS) [64]. The goal of DS clustering is to maintain a continuously consistent good clustering of the observed sequence, using a limited amount of memory, time, and information about data. Performing the clustering analysis over DS requires additional challenges, including the need for quick responses and the single-pass constraint over the raw data. Several stream clustering approaches have been proposed in the literature to address the DS clustering problem: they can be mainly categorized into *partitioning-based* (e.g., Leader, Stream K-means, and CluStream), *density-based* (e.g., DenStream), and *grid-based* e.g., DStream) [64]. In particular, the partitioning-based DS clustering techniques are divided, in turn, into adaptive methods (e.g., Leader and Stream K-Means) and Online-Offline clustering methods (e.g., CluStream) [64]. For instance, in [77] a fault-detection system based on data stream prediction is proposed as a DS management system, including different data prediction methods both for short-term and long-term prediction: the achieved results show that the methods using more historical data perform better in long-term prediction, while the methods using the most recent data perform better in short-term prediction. Another peculiarity of

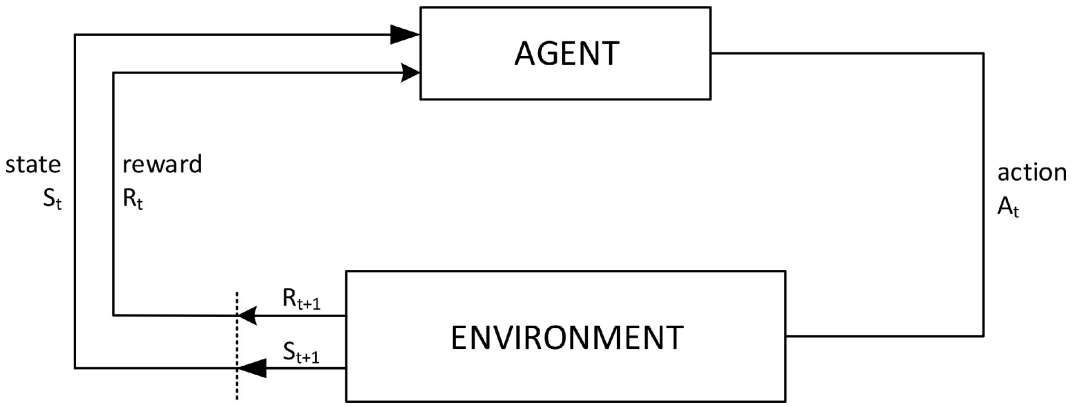


Fig. 2.23 General Reinforcement Learning Structure. [11]

DS mining is the so-called *concept drift*, which refers to the unforeseen change in statistical properties of the instances coming from the stream of data over time [78]. Especially when the goal of the clustering activity is to determine a fault detection strategy, understanding how and when the stream of data changes over time is crucial. To this aim, in [79] authors indicate four different concept drift modes: sudden, gradual, incremental, and recurring. A method to detect concept drift in data streams as a potential indication for faulty system behaviors is proposed in [80]. Although the results show the effectiveness of the approach, it is worthwhile noting that the concept drift does not always represent a meaningful measure of the degradation of the process under analysis.

2.4 Reinforcement Learning Methods

Reinforcement Learning (RL) is a type of ML paradigm where an agent learns to make a sequence of decisions by interacting with an environment. The agent aims to maximize a cumulative reward signal over time through a trial-and-error process. RL is commonly used in scenarios where the agent has to make a series of actions to achieve a goal, and it does not have access to a labeled dataset but can learn from its own experiences (see figure 2.23).

In figure 2.23 the interaction between the agent and the environment is usually defined by the formal framework of Markov Decision Processes (MDP).

2.4.1 Brief Overview on Reinforcement Learning Methods

There are several types or categories of RL, including:

Model-Free vs. Model-Based RL

In model-free RL, the agent learns a policy that maps states to actions directly without building an explicit model of the environment. Common algorithms include Q-learning, State-Action-Reward-State-Action (SARSA), and various policy gradient methods.

In model-based RL, the agent constructs an internal model of the environment to simulate possible outcomes of actions. This model is used for planning and decision-making. Model-based RL can be more sample-efficient but requires learning an accurate model of the environment.

Value-Based vs. Policy-Based RL

In value-based RL, the agent learns a value function that estimates the expected cumulative reward of being in a certain state and taking a certain action. Q-learning and Deep Q-networks (DQN) are examples of value-based methods.

In policy-based RL, the agent directly learns a policy, which is a mapping from states to actions, without explicitly computing value functions. Policy gradient methods like REINFORCE are used in policy-based RL.

On-Policy vs. Off-Policy RL

On-policy methods update the policy based on the experiences generated by the current policy. This can lead to slower learning but is more stable.

Off-policy methods allow the agent to learn from experiences generated by a different policy than the one being updated. This often leads to more efficient learning but can be less stable.

Exploration Strategies: Exploration vs. Exploitation

RL agents face the exploration-exploitation dilemma, where they must balance exploring unknown actions to discover better strategies and exploiting known actions to maximize rewards. Various exploration strategies, such as Epsilon-Greedy, Upper Confidence Bound (UCB), and Thompson sampling, are used to address this dilemma.

Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines RL with deep neural networks, allowing agents to handle high-dimensional state and action spaces. DRL methods have achieved remarkable success in applications like game playing, robotics, and autonomous vehicles.

These categories and types provide a framework for understanding the diverse range of RL methods and approaches used to solve complex decision-making problems in various domains. The choice of RL algorithm depends on the specific problem, the available data, and the trade-offs between exploration and exploitation.

2.4.2 A Brief Assessment of Reinforcement Learning for Failure Detection and Diagnosis

In the realm of RL for failure detection and diagnosis, existing research has focused on developing intelligent systems that can autonomously learn and adapt to complex industrial environments. Compared to supervised or unsupervised learning, RL is still not as frequently used in FDD, reliability, and safety applications [42]. However, it presents several potentials for making crucial contributions to these fields. The related works delve into the application of RL algorithms to enhance the reliability and performance of FDD processes in industrial systems. They explore the utilization of RL techniques to optimize maintenance strategies, reduce downtime, and improve overall system resilience. By leveraging RL, these studies aim to pave the way for more efficient and proactive FDD solutions in industrial contexts. RL has been applied, for instance, to condition-based maintenance planning for multi-component systems with conflicting risks [81]. With the complexity of engineering systems

rising, optimization of CBM with Markov Decision Processes (MDP) and other traditional decision-making techniques becomes difficult, if not computationally infeasible. To find effective inspection and maintenance procedures for large-scale infrastructure systems, authors in [82] used Deep Reinforcement Learning.

2.5 Conclusion of Machine Learning Models for Failure Detection and Diagnosis

In summary, this chapter provides an overview of ML models for FDD with a focus on four main categories of ML techniques: supervised, unsupervised, semi-supervised, and reinforcement learning. Each category is introduced and explained in detail, highlighting subsets within each. The chapter delves into the mathematical optimization aspects and provides comprehensive descriptions of key algorithms associated with each category. Furthermore, it offers insights into related work in FDD, presenting a thorough literature review for each method as applied to industrial systems. This comprehensive exploration sets the stage for a deeper understanding of the role of main ML techniques in FDD and their potential applications in real-world scenarios.

Within this chapter, I have discussed the four primary categories of ML models central to the realm of FDD. However, it is important to note that the landscape of ML is vast and continuously evolving. Alongside the models explored here, there exist other techniques, including Physics-Informed ML, Transfer Learning, Automated (AutoML), and more, which have found utility in FDD and industry 4.0 applications[83]. Although these approaches bear significant relevance, my emphasis in this discussion is placed squarely on the core ML models, and I will not delve into the intricacies of these other methodologies.

Chapter 3

Supervised Methods: Data-Driven Fault Diagnosis in a Complex Hydraulic System based on Early Classification

3.1 Introduction to Early Classification Algorithms and Contribution

In the previous works, the proposed CBM systems are able to accurately monitor the working condition of the industrial devices. However, they have not exhibited the capability to proactively predict equipment failures beforehand. In the related literature of HSs, the decision-making is postponed until the entire information has been gathered. Notably, prior research endeavors have not included the ability to anticipate the timing of system failure within a working cycle, nor to proactively avert system operation in the face of potentially critical faults. In fact, the time of a potential failure in a component, which may be identified via automated processing of available sensors' signals, represents one of the most crucial Health Indicators (HIs) in the realm of HSs. In scenarios where new measurements arrive sequentially, there is a need for progressive monitoring of an asset's health status. Given the irregularity of these measurements and their asynchronous nature, it becomes paramount to expedite decision-making to enhance system performance([12]). As the HS runs

over cycling operations, several relevant parameters may be monitored to observe any progressive change leading to some potential failure. However, at this time, only a slight drop in running performance occurs, and it is not possible to prevent these faults through traditional fault diagnosis techniques. As a result, it is crucially needed to raise an early warning of fault according to the changing trend of working signals in HSs.

The main contribution of this work is early fault detection to avert equipment breakdown and enhance the efficiency of industrial systems. To achieve this goal, I utilize an early time-series classification (ETSC) algorithm to solve the problem of a complex HS with several interconnected components. As it can be seen in section 1.5.1, the sequential data, generated by different types of sensors, are processed and an early classification model is applied for each component of the HS to detect in advance potential failures. By taking both accuracy and earliness into account, this adaptive approach formalizes the early classification of time series as a sequential decision-making task and forecasts the healthy state of incoming and incomplete sensors' data before the observation of the entire time series. In other words, at each time step, it computes the optimal expected time for decision-making in advance. As a result, the early classification model successfully achieves a trade-off between the performance and the earliness criterion. This early predicted failure time helps to prevent the HS from breaking down and reduces the cost of maintenance. Finally, since the failure time is known in advance, the remaining useful life of the components may be anticipated by dividing the asset's life cycle into healthy and unhealthy stages. The experimental results on a realistic HS dataset from the related literature show that the ETSC method can effectively identify different fault types with a higher accuracy and earlier prediction time compared to baseline methodologies.

The rest of this chapter is organized as follows. Section 3.2 describes the ETSC method aimed at classifying the eventual fault types of each new working cycle of the HS. In Section 3.4, the proposed methodology is applied to the early classification of the state of the main HS components, and the achieved results are comprehensively discussed and compared with those obtained by baseline methods. Lastly, concluding remarks and outlooks to future works are summarized in Section 3.5.

Algorithm 1: Early Time-Series Classification Algorithm

Inputs: $\mathbf{x}_t = \langle x_1, x_2, ..., x_t \rangle$, $\left\{h_t^k\right\}_{t\in\{1,...,T\},k\in\{1,...,K\}}$

$\mathbf{x}_t \leftarrow \langle x_1 \rangle$
$t \leftarrow 1$ **while** $(\neg \textsc{Trigger}\left(\mathbf{x}_t, \left\{h_t^k\right\}_{k\in\{1,...,K\}}\right)) \vee (t < T)$ **do**
    $t \leftarrow t+1$
$\mathbf{x}_t \leftarrow \langle \mathbf{x}_{t-1}, x_t \rangle$
$k = \underset{k\in\{1,...,K\}}{\operatorname{argmax}} P(\mathfrak{c}_k \mid \mathbf{x}_t)$, $\hat{y} \leftarrow h_t^k(\mathbf{x}_t)$
$t^\star \leftarrow t$

Outputs: $\hat{y}$, $t^\star$

procedure $\textsc{Trigger}(\mathrm{x}_t, \{\mathrm{h}_t^k\}_{k\in\{1,...,K\}})$
    $\mathscr{T} \leftarrow False$ **for** $\tau \in \{0, \ldots, T-t\}$ **do**
        Compute $f_\tau(\mathbf{x}_t)$ through (3.3)
    $\tau^* = \underset{\tau\in\{0,...,T-t\}}{\operatorname{argmin}} f_\tau(\mathbf{x}_t)$ **if** $(\tau^* = 0)$ **then**
        $\mathscr{T} \leftarrow True$
    return $\mathscr{T}$
end procedure

3.2 The Early Time Series Classification Methodology

In this section, I outline an early time-series classification algorithm based on the framework introduced in [12]. This algorithm is designed to forecast the category of incoming time-series data while observing a minimal number of initial measurements.

Let us consider a system with a cyclic operation, whose generic component working-cycle is represented by a T-dimensional time series of real-valued measurements $\mathbf{x}_T = \langle x_1, x_2, \ldots, x_T \rangle$ that can be classified by a label y in accordance with a finite set $\mathscr{M}$ of $M = |\mathscr{M}|$ classes. The goal of the ETSC is to optimally classify a new and still incomplete time-series $\mathbf{x}_{t^\star} = \langle x_1, x_2, \ldots, x_{t^\star} \rangle$ at $t^\star \in \{1, \ldots, T\}$ before the full observations become available.

To this aim, let us consider a set $\mathscr{S}$ of m sequences for the training of the early classification model. Each sequence is a couple $\left(\mathbf{x}_T^i, y_i\right) \in \mathbb{R}^T \times \mathscr{M}$, where $\mathbf{x}_T^i = \left\langle x_1^i, x_2^i, \ldots, x_T^i \right\rangle$ $(i = 1, \ldots, m)$ is the i-th time-series and y_i represents the associated label. The training set $\mathscr{S}$ is used to determine a series of classifiers h_t (with $t \in \{1, \ldots, T\}$): each classifier h_t estimates the class of the t-dimensional sub-series $\mathbf{x}_t = \langle x_1, x_2, \ldots, x_t \rangle$. The early classification model is based on the estimation of the earliest time $t^\star$ at which the prediction $h_{t^\star}(\mathbf{x}_{t^\star})$ is equivalent to $h_T(\mathbf{x}_T)$, i.e., the classifier based on the observation of the entire time-series $\mathbf{x}_T$.

As a first pre-condition, all the full-length training time-series in $\mathscr{S}$ must be partitioned into K clusters $\{\mathfrak{c}_1, \mathfrak{c}_2, \ldots, \mathfrak{c}_K\}$. Clusters should have similar time series and be distinct from one another, such that an incoming incomplete sequence is generally assigned to one of them. The number of clusters for each target class corresponds to the maximum silhouette factor, which is a metric used to assess the performance of a clustering technique. The Euclidean distance is used to calculate the distance between an incomplete incoming time series and a complete one. Then, the membership probabilities of the sub-series to each cluster are computed. In particular, the membership probability of sub-series $\mathbf{x}_t$ to cluster $\mathfrak{c}_k$ is denoted as $P(\mathfrak{c}_k \mid \mathbf{x}_t)$, and is computed as follows:

$$P(\mathfrak{c}_k \mid \mathbf{x}_t) = \frac{s_k}{\sum_i^K s_i} \tag{3.1}$$

where s_k is a sigmoid function defined as:

$$s_k = \frac{1}{1 + \exp^{-\lambda \Delta_k}}. \tag{3.2}$$

Note that in (3.2) λ is a constant value and Δ_k is the difference between the average of all distances of the clusters to $\mathbf{x}_t$, and the distance of the cluster $\mathfrak{c}_k$ and $\mathbf{x}_t$.

As a second pre-condition, for each time-point $t \in \{1, \ldots, T\}$ the confusion matrix for each cluster and each classifier h_t is computed based on the frequencies observed in the training data, and is denoted as $P_t(\hat{y} \mid y, \mathfrak{c}_k)$, where $\hat{y}$ and y are the predicted and the true class, respectively.

The proposed methodology for the early classification is formally described in Algorithm 1. The core part of the algorithm lies in the procedure TRIGGER reported at the bottom part of Algorithm 1 (lines 9-19). Such a procedure implements a

function that looks for a future time step at which a trustworthy classification can be made.

Given x_t at time t, $T-t$ measurements are still missing on the incoming series to carry out the complete cycle. To find the optimal future step $\tau \in \{0,\dots,T-t\}$ when the incoming time-series will be correctly classified, an expected cost function is calculated in the TRIGGER function as follows:

$$f_\tau(\mathbf{x}_t) = \sum_{\mathfrak{c}_k} P(\mathfrak{c}_k \mid \mathbf{x}_t) \sum_{y \in \mathscr{M}} \sum_{\hat{y} \in \mathscr{M}} P_{t+\tau}(\hat{y} \mid y, \mathfrak{c}_k) C_t(\hat{y} \mid y) + C(t+\tau). \tag{3.3}$$

Equation 3.3 represents the cost function associated with the prediction decision problem and is composed of two terms: misclassification and time cost. The term $C(t+\tau)$ in (3.3) is a non-decreasing function, indicating the cost of delaying the classification at future time $t+\tau$. Conversely, the first term in (3.3) is based on the misclassification function $C_t(\hat{y} \mid y) : \mathscr{M} \times \mathscr{M} \longrightarrow \mathbb{R}$ –that provides the cost at time t of predicting $\hat{y}$ when the true class is y– and the conditional probabilities $P(\mathfrak{c}_k \mid \mathbf{x}_t)$ and $P_{t+\tau}(\hat{y} \mid y, \mathfrak{c}_k)$.

Equation (3.3) allows us to determine the optimal time step τ^* at which the expected cost is minimum. When a new measurement is available, the computation of the expected cost is re-evaluated (lines 3-4): only and only when the current time corresponds to the optimal prediction time, the classification decision procedure stops and the classifier h_t^k is selected to predict the class of the input sequence x_t (lines 7-8). Indeed h_t^k is the base classifier that is learned over t time steps using the k-th cluster of the complete time series in the training phase.

3.3 Illustrative Numerical Example

In this section, I provide a step-by-step numerical example to illustrate the workings of the ETSC (Algorithm 1).

In Figure 3.1, confusion matrices are computed for each time step and cluster. While in Figure 3.2, the curves visually demonstrate the tradeoff between the gain in expected precision and the cost of delaying decisions, with the minimum expected at time τ^*. The introduction of new measurements has the potential to alter both the

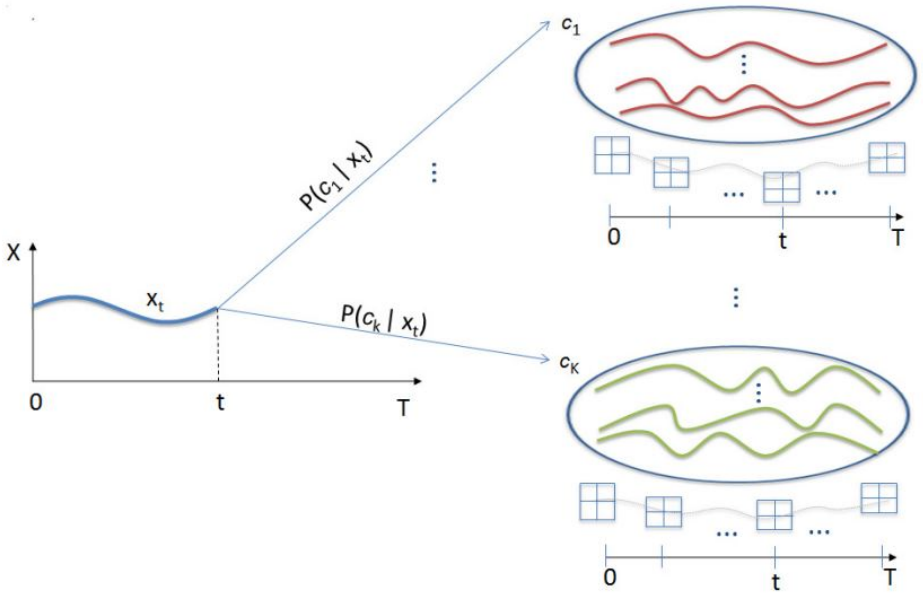


Fig. 3.1 Comparison of an incoming sequence with clusters from the training set [12].

expected cost curve and the estimated τ. This numerical example demonstrates the step-by-step execution of the ETSC for the provided inputs in Section 3.4.3.

3.4 Case Study: Complex Hydraulic System

In this section, the proposed ETSC methodology is applied to a real-world complex HS, as detailed in section 1.5.1. Applying this early classification model allows us to not only detect possible faults but also estimate as soon as possible the time of these faults by meeting the earliness criterion.

3.4.1 Data Pre-processing and Sensor Selection

Before conducting any data-driven process, data pre-processing is required. Undoubtedly, the quality of the training dataset plays a crucial role in determining the accuracy of classification methods. Using all the attributes in a large data set would lead in an over-fitted model ([84]). Conversely, the main characteristic of each fault is strongly influenced only by some correlated sensor signals. In this case study, the most significant sensor signals for CBM of each component are selected based on

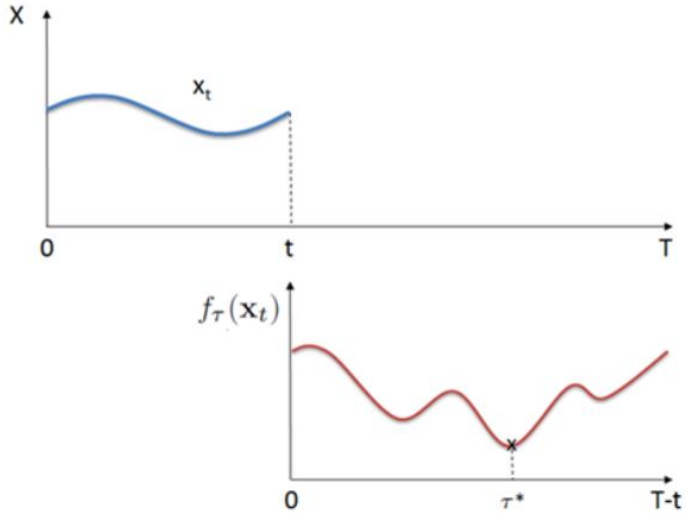


Fig. 3.2 The first curve depicts an incoming time series x_t, while the second curve illustrates the expected cost $f_\tau(\mathbf{x}_t)$ given x_t, $\forall \tau \in \{0,\ldots,T-t\}$ [12].

Table 3.1 The most correlated sensor for each component of the HS.

Component	**The most correlated sensor**
Valve	PS1
Cooler	CE
Pump	SE
Accumulator	FS1

the correlation between the features and the fault degrees ([32, 35]). The pressure ($PS1-PS6$), flow rate ($FS1-FS2$), and temperature ($TS1-TS4$) sensor signals have been plotted in figure 3.3, 3.4, and 3.5 respectively. Furthermore, Figure 3.6 illustrates motor power (EPS1) and vibration (VS1), while Figure 3.7 exhibits signals for Cooling Efficiency (CE), Cooling Power (CP), and System Efficiency Facto (SE). The most correlated time-domain sensors to each fault type are shown in Table 3.1 based on the Spearman ranking correlation analysis.

As usually done in supervised learning tasks, the series of observations including both features and corresponding labels is partitioned into two distinct subsets: the training set and the testing set. The model learns on the training data using both features and labels, while its performance is measured on the testing set when only features are employed to predict the labels. In this case study, the 2205 working

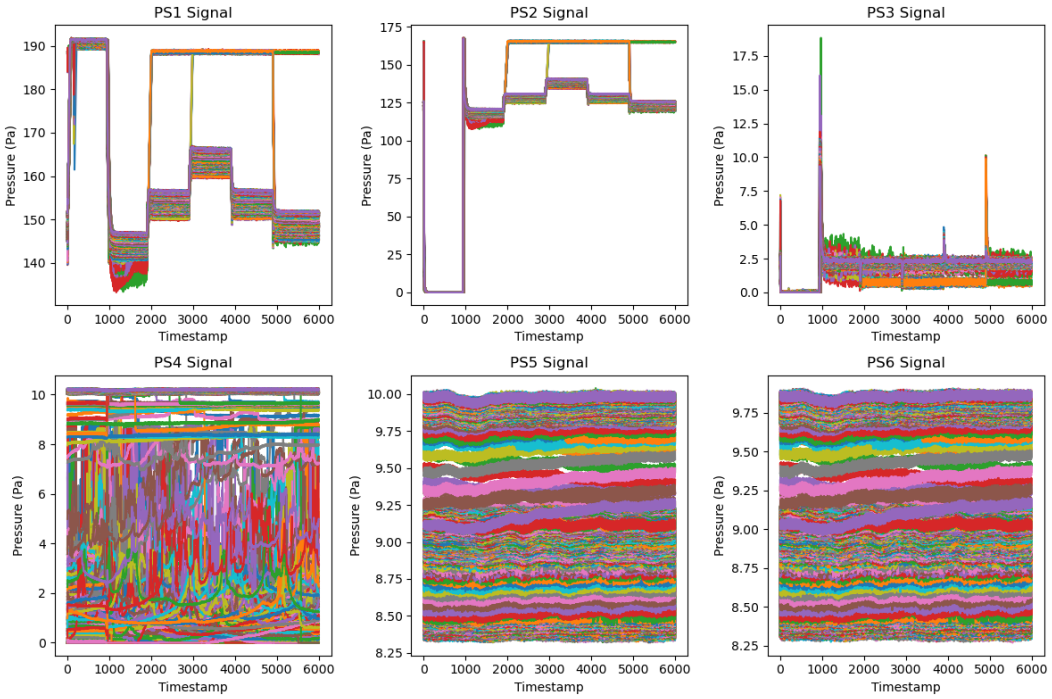


Fig. 3.3 PS1-PS6 pressure signals of the HS Dataset

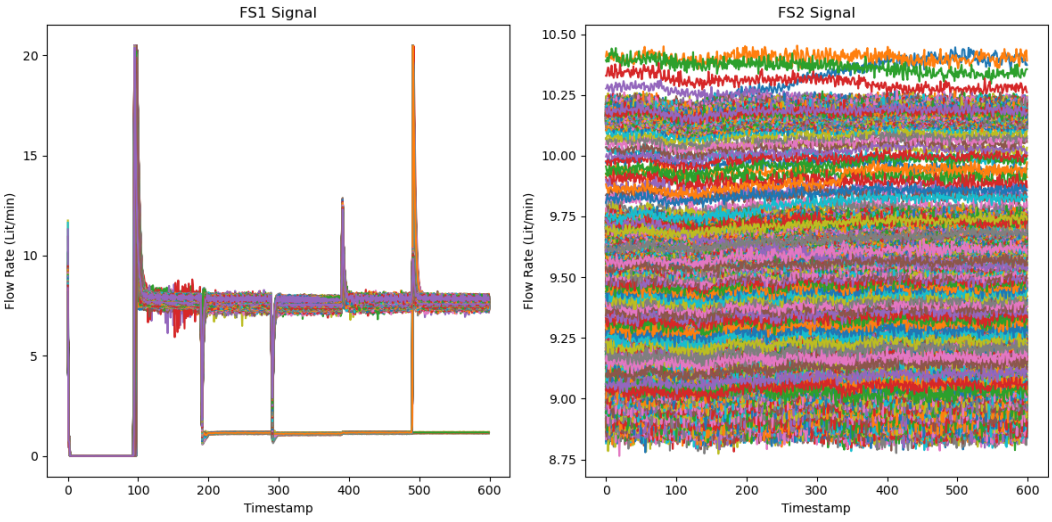


Fig. 3.4 FS1-FS2 flow rate signals of the HS Dataset

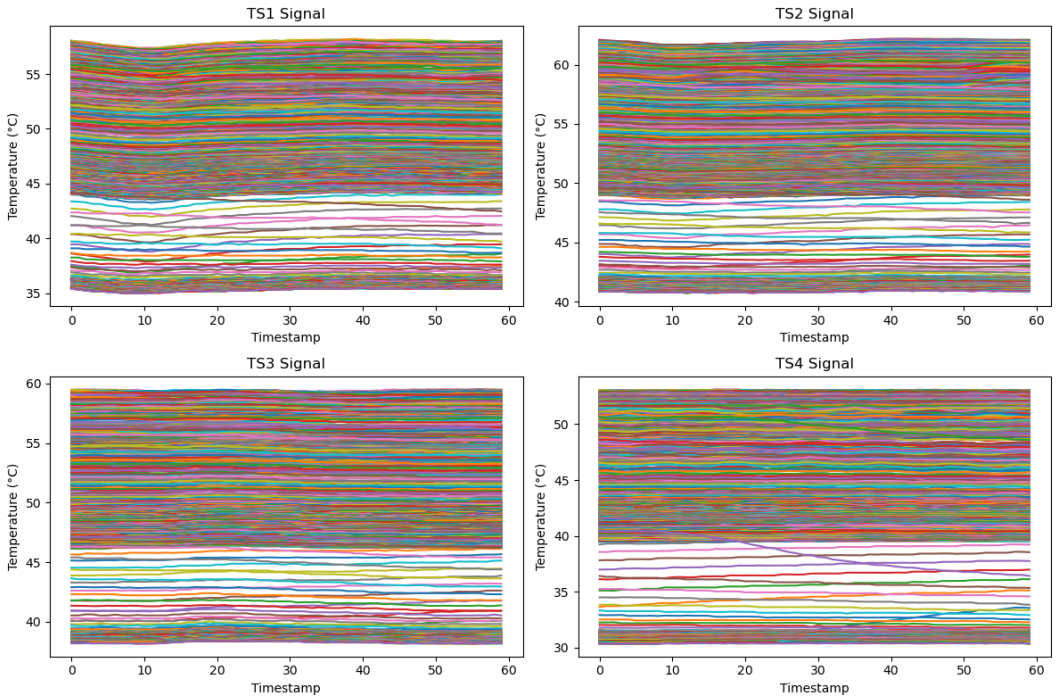


Fig. 3.5 TS1-TS4 Temperature signals of the HS Dataset

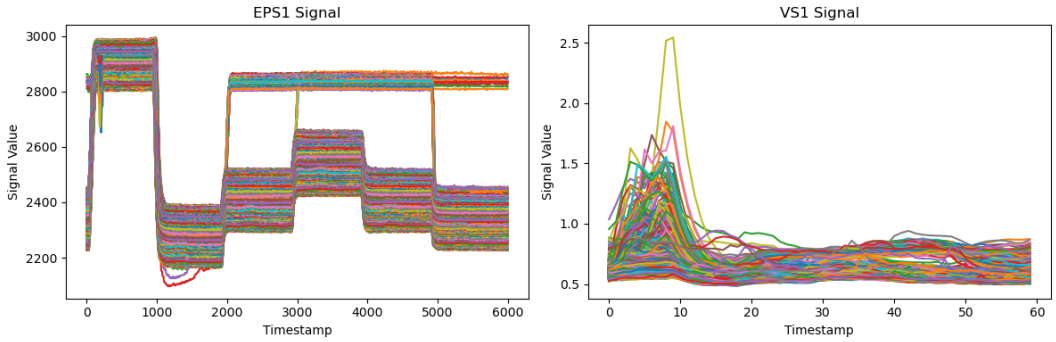


Fig. 3.6 EPS1 and VS1 signals of the HS Dataset

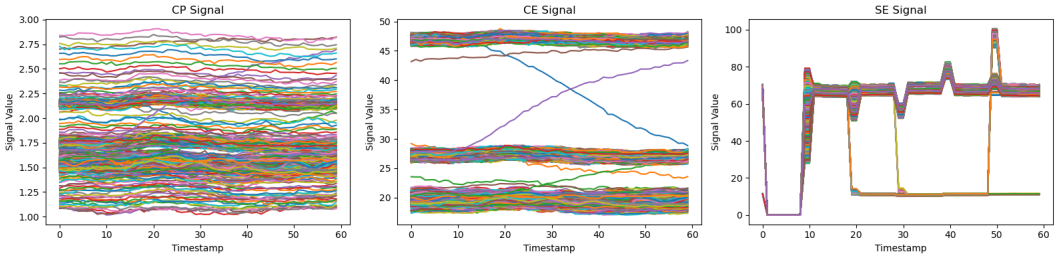


Fig. 3.7 CP, CE and SE signals of the HS Dataset

cycles of the selected sensors are splitted into training (75%) and testing (25%) subsets. Certainly, a portion of the training data can be utilized for model validation as well.

3.4.2 Baseline Methods and Performance Indicators

The following reference methods are introduced to evaluate the performance of the proposed techniques. One of the most used classification methods is Linear discriminant analysis (LDA), which divides each sample of various sensors into numerous intervals and performs feature extraction using median, variance, skewness, and kurtosis. Then the Pearson correlation coefficient is used to analyze the correlation between sensors' data features and faults. In addition, common classification methods such as artificial neural networks (ANNs) and support vector machine (SVM) with linear and radial basis function (RBF) kernels are also used as alternative state-of-the-art techniques.

To compare the results of the ETSC method with those of the baseline models, I employ two key metrics: accuracy and earliness. The accuracy of a classification task is calculated based on the so-called confusion matrix, also known as an error matrix. By definition, a confusion matrix CM is a square matrix whose dimension corresponds to the number M of classes and whose element $\mathrm{CM}(i,j)$ is equal to the number of observations known to be in class i (actual class) and predicted to be in class j (predicted class). Then the accuracy metric indicates how close the predicted labels are to the actual labels, and is defined as follows:

$$\text{Accuracy} = 100\,\frac{\sum_{i=1}^{M}\mathrm{CM}(i,i)}{\sum_{i=1}^{M}\sum_{j=1}^{M}\mathrm{CM}(i,j)} \tag{3.4}$$

Table 3.2 Tuned values of main hyper-parameters for each HS component.

Component	Cost time ($C(\cdot)$)	Sigmoid parameter (λ)
Valve	0.001	1000
Cooler	0.01	1000
Pump	0.01	1000
Accumulator	0.001	1000

where $\sum_{i=1}^{M} \mathrm{CM}(i,i)$ indicates the number of sample that have been truly labeled using the classification algorithm (i.e., the elements on the diagonal of matrix CM), while $\sum_{i=1}^{M}\sum_{j=1}^{M} \mathrm{CM}(i,j)$ represents the total number of samples in the given data set. If the entire set of predicted labels for a data set strictly match the true set of labels, then the accuracy achieves the highest value, i.e., 100%.

Second, the earliness metric is considered to provide the time of fault diagnosis during the procedure of early classification. If the early classifier uses $t^{\star}$ data points of a testing time series during classification, then earliness is defined as follows:

$$\text{Earliness} = 100\,\frac{T - t^{\star}}{T} \tag{3.5}$$

where T is the length of complete time series ([31]). If the entire set of measurements is used to perform the classification, then earliness achieves the lowest value, i.e., 0%.

3.4.3 Results and Discussion

For every hydraulic component, I implement and fine-tune the ETSC model in Python. Subsequently, I apply this model individually to the chosen sensor data. To assess how each hyperparameter influences the decision-making process, I systematically vary their values within predefined ranges The main hyper-parameters of the early classification model and the best values after the tuning process for each component are listed in Table 3.3.

In particular, the most important parameter is the cost time parameter, indicated by $C(\cdot)$ in (3.3). In particular, this parameter is varied in the range:

$$\{0.1, 0.01, 0.001, 0.0001, 0.00001\}$$

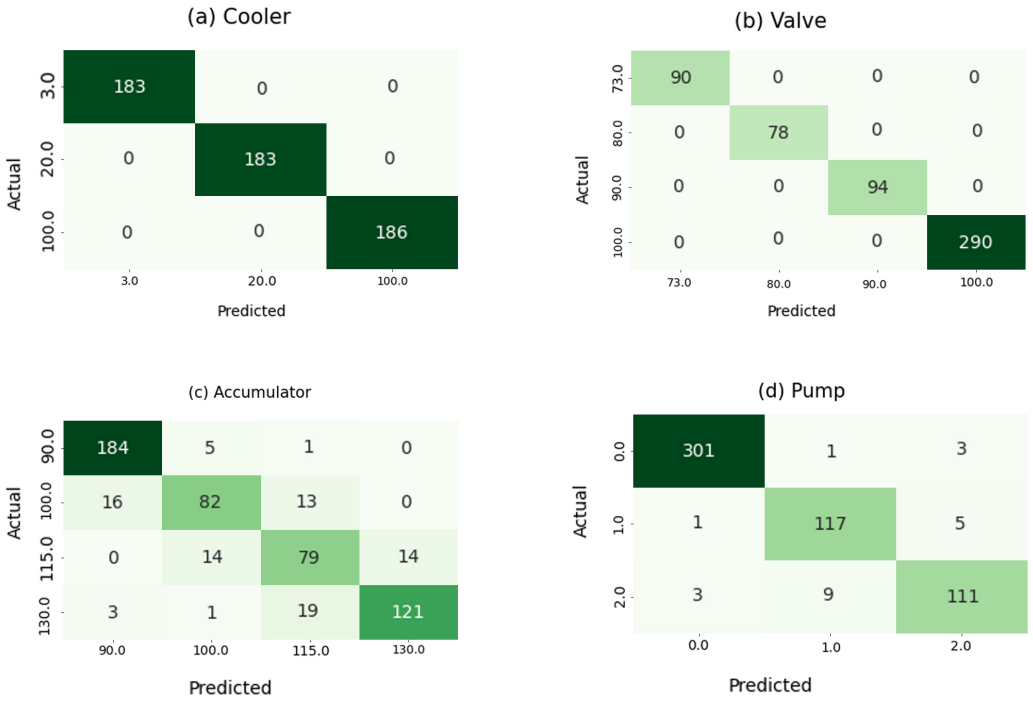


Fig. 3.8 Confusion matrix of ETSC model for different fault types (cooler, pump, accumulator, and valve)

when the cost time is very low, the number of subgroups in each class, and thus the complexity of the classes, do not influence the results; conversely, when the cost time increases, the decision task becomes harder, and the early decision-making is postponed. As for the sigmoid function parameter – denoted as λ in (3.2) – by varying its value in the range $\{10, 100, 1000\}$ no significant difference in the accuracy of the models is observed.

Since I face a multi-classification problem, the ETSC model is first quantitatively evaluated by the confusion matrix. Generally, the diagonal and off-diagonal elements of the confusion matrix represent successful and unsuccessful detection of fault/healthy signals, respectively. It is apparent that the diagonal values of the confusion matrices for all components (Fig. 3.8) are much larger than the off-diagonal values, meaning that the ETSC method perfectly classifies the fault/healthy condition of each working cycle.

In general, classification methods like ANN, SVM, and ETSC exhibit higher overall accuracy compared to statistical methods such as LDA in diagnosing faults

Table 3.3 Tuned values of main hyper-parameters for each HS component.

Component	Cost time ($C(\cdot)$)	Sigmoid parameter (λ)
Valve	0.001	1000
Cooler	0.01	1000
Pump	0.01	1000
Accumulator	0.001	1000

Table 3.4 Comparison of the proposed and baseline methods in terms of accuracy (%) for each HS component.

Component	LDA*	ANN*	SVM* (Linear)	SVM* (RBF)	**ETSC**
Valve	100	100	100	100	**100**
Cooler	100	100	100	95.7	**100**
Pump	73.6	80.0	72.4	64.2	**96.0**
Accumulator	54.0	50.4	51.6	65.7	**84.4**
Overall	81.9	82.6	81.0	81.4	**95.1**

* accuracy results taken from ([35]).

Table 3.5 Comparison of the proposed and baseline methods in terms of fault time (timestamp) for each HS component.

Component	Baseline methods	**ETSC**
Valve	6000	966, 972
Cooler	60	1
Pump	60	1, 12, 14
Accumulator	600	114, 126

in the HS being studied. As shown in Table 3.4, the accuracy of the ETSC is the maximum value (i.e., 100%) for the valve and the cooler diagnosis model, whilst it is respectively 96.0% and 84.4% for the pump the accumulator, which means that the diagnosis model performs quite well on the correlated sensors. The accuracy of all baseline methods is instead not satisfying for CBM of the accumulator.

The most important advantage of the ETSC model with respect to the baseline methods lies in the ability to estimate the time of a potential fault. In all the baseline models presented in Table 3.4, the detection of faults occurs at the end of a working cycle, while the ETSC model can diagnose the healthy/fault condition of the system

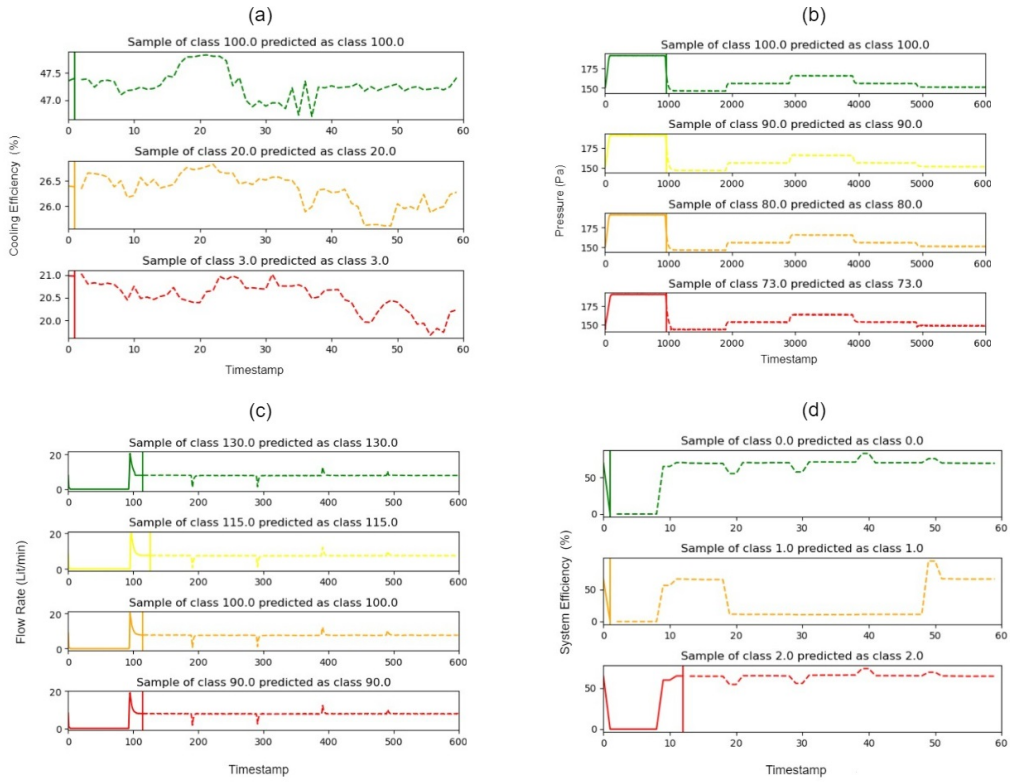


Fig. 3.9 Early fault time for the cooler (a), valve (b), accumulator (c), and pump (d): the vertical line shows the optimal time $t^\star$ (in terms of timestamps) dividing the signal in observed (solid) and unobserved (dash) parts.

in advance, with no need to wait up to the end of the working cycle. As it can be seen in Table 3.5, the estimated time of each fault type is much less than the duration of a complete working cycle. For instance, in the case of the accumulator, the early classification model can diagnose any fault after observing 114 or 126 measurements (i.e., the earliness is 81.0% or 79.0%), while the baseline models must explore the entire measurements which takes exactly 600 timestamps (i.e., the earliness is 0%). In particular, Fig. 3.9 represents the estimated fault time of some specific signals for the hydraulic pump component. Indeed, the threshold of measurements has been gradually increased during the working cycles to satisfy the trade-off between accuracy and earliness.

Summing up, two main outcomes are apparent: on the one hand, the applied ETSC method is able to assign each incoming sequence to the corresponding fault types with high accuracy; on the other hand, it can estimate an optimal fault diagnosis time, thus balancing between the classification accuracy and the cost of delaying the classification result.

3.5 Conclusion of the Early Time Series Classification Method

Nowadays, the research on intelligent fault prognosis for HSs is increasingly growing due to its potential to improve operational safety and reliability, lower maintenance costs, and boost productivity in several industrial sectors. Given the cycling operation of HSs, several relevant parameters may be monitored to detect any progressive change leading to some potential failures. At this phase, only a slight drop in running performance occurs, which is not detectable through traditional fault diagnosis techniques, and thus it is crucial to raise an early warning of fault considering the changing trend of working signals in HSs. In this paper, an early time-series classification (ETSC) algorithm is applied to support fault prognosis in a complex HS with several interconnected components. The proposed technique aims at early classifying the state of the system while keeping the loss of classification inaccuracy at the minimum level. In contrast to baseline models that detect eventual failures at the end of each working cycle, the ETSC model can predict any fault type of the HS components before observing the entire working cycle. Indeed, the early classification model successfully achieves a trade-off between the performance and the earliness criterion. Experimental results on a realistic HS dataset from the related literature show that the ETSC method can effectively identify different fault types with higher accuracy and earlier compared to baseline methodologies.

Chapter 4

Supervised Methods: An Integrated Approach for Failure Diagnosis and Analysis of Industrial Systems Based on Multi-Class Multi-Output Classification

4.1 Introduction to Multi-Class Multi-Output Classification

In contrast to the previous literature, the goal of this work is to combine ML with statistical FA to create a tool that can assess a complex system's behavior with a minimum of user intervention. Specifically, the ML model is used to detect the states of the hydraulic components and the ultimate state of the HS, while the statistical FA is utilized to determine the primary cause of the failure. In fact, I propose a hybrid model that includes both ML and statistical FA techniques. In order to process the sensor measurement data coming from the industrial HS and forecast not only the potential failures of the system components but also the overall state of the system and simultaneously identify the root cause of failure, a Multi-Class Multi-Output (MCMO) classification method is constructed. MCMO model is appropriate for

handling complex classification problems with multiple labels. This model allows for the assignment of multiple labels to a single data point, which can be beneficial in cases where a data point belongs to more than one output or has multiple classes.

The rest of this chapter is organized as follows. Section 4.2 presents the MCMO method to classify fault types industrial systems, used as inputs for the rule-based model for the purpose of FA. Section 4.3 describes a case study where the MCMO method is applied to the state of the components of the HS, and results are discussed and compared with baseline methods. Lastly, concluding remarks and outlooks for future work are presented in Section 4.4.

4.2 A Hybrid Model for Failure Analysis

This section outlines a two-step procedure presented for the complex multi-component system to diagnose not only the failure of the system but also the fault of each component at the same time (see Figure 5.1). The proposed hybrid model consisting of the MCMO classification and the rule-based system for FA is described in Algorithm 2.

4.2.1 Multi-Class Multi-Output Classification

An MCMO classification problem involves forecasting multiple target variables, where each target variable has more than two possible classes.

MCMO classification models based, for instance, on Logistic Regression (LR), K-Nearest Neighbor (KNN), Support Vector Machines (SVM), Decision Tree (DT), and Random Forest (RF) – denoted as MLR, MKNN, MSVM, MDT, and MRF, respectively – are an extension of traditional classification techniques that can handle multiple output variables simultaneously. In contrast to the corresponding single-output models LR, KNN, SVM, DT, and RF, the MCMO models can predict multiple dependent variables that are correlated with each other. In addition to labeling each basic component to its corresponding class, MCMO models can also label the entire state of the system to a specific binary class (Stable and Non-stable).

Let us consider a system with a cyclic operation, whose generic component working-cycle is represented by a set $\mathscr{X} = \left\langle \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(N)} \right\rangle$ of N time-series, each being composed of T real-valued measurements $\mathbf{X}^{(i)} = \left(x_1^i, x_2^i, \ldots, x_T^i\right)$ (for each

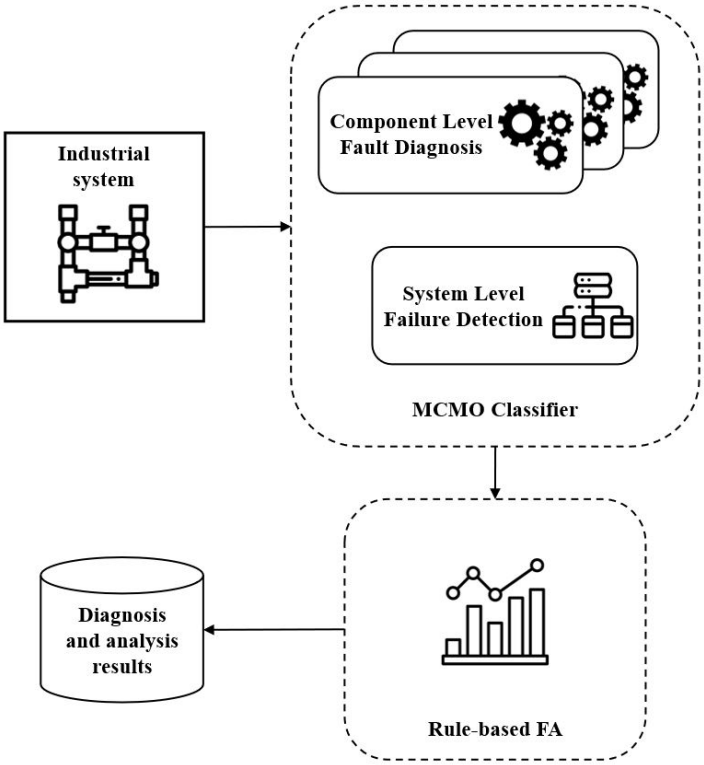


Fig. 4.1 Scheme of the failure diagnosis and analysis method based on MCMO classification.

$i = 1,2,\ldots,N$). Each time-series is classified in accordance with M labels defined in $\mathscr{C} = \{c_1, c_2, \ldots, c_M\}$: $\mathscr{Y} = \left\langle Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)} \right\rangle$ denotes the labels corresponding to the time-series in $\mathscr{X}$.

The goal of the MCMO classification is to optimally classify the time-series set $\mathscr{X}$ to the class labels $\mathscr{Y}$, i.e., determining the pairing $\left(\mathbf{X}^{(i)}, Y^{(i)}\right) \in \mathbb{R}^T \times \mathscr{C}$ for each $i = 1,2,\ldots,N$.

The proposed MCMO-based method is described in detail in steps 1-4 of Algorithm 2. The inputs to the algorithm are a set of time series $\mathscr{X}$ and its corresponding multi-labels $\mathscr{Y}$. The algorithm splits the dataset into training and testing sets and fits an MCMO classification model using the training set. It then uses the trained model to predict the labels $\mathscr{Y}_{pred}$ for the testing set.

Note that the reproducibility of the MCMO classification method depends on several factors, including the availability and quality of training data, the chosen model, and the specific implementation. To ensure reproducibility, it is important to

Algorithm 2: Failure Diagnosis and Analysis based on Multi-Class Multi-Output classification

Inputs: $\mathscr{X}$, $\mathscr{Y}$

Outputs: FA results

1: Split the dataset $\mathscr{X}$, $\mathscr{Y}$ in train data $\mathscr{X}_{tra}$, $\mathscr{Y}_{tra}$ and test data $\mathscr{X}_{test}$, $\mathscr{Y}_{test}$
2: Build the MCMO model using the train data $\mathscr{X}_{tra}$, $\mathscr{Y}_{tra}$
3: Determine the classes $\mathscr{Y}_{pred}$ associated with the test data $\mathscr{X}_{test}$
4: Compute the accuracy and precision of the MCMO model comparing $\mathscr{Y}_{test}$ and $\mathscr{Y}_{pred}$
5: Define all failure modes M_j from $\mathscr{Y}_{pred}$
6: Compute the failure probability of M_j
7: Find the steady-state and transient-state of the system
8: Perform the statistical FA

have a well-defined and representative training dataset that accurately represents the variability and complexity of the problem at hand. It should include samples from all classes of each target variable to provide sufficient information for the MCMO model to learn.

4.2.2 Rule-based Model for Failure Analysis

A rule-based model is a systematic approach used to analyze the possible causes of failure in a system. The procedure starts by defining all possible failure modes M_j that could occur in the system, having identified all the potential issues that could arise and lead to the failure of the system by taking a type of fault in the components. Subsequently, the failure probability (FP) for each of the identified failure modes M_j is computed. The probability of a given failure mode M_j is $\mathbb{P}\left(M_j\right) = N_{M_j}/N$, which corresponds to the ratio between the number N_{M_j} of observations and the total number N of failures in the dataset. The results of the statistical FA are used for the development of a risk management plan, with the final aim of minimizing the risk of failure and ensuring a safe and efficient operation. The statistical FA procedure is described in steps 5-8 of Algorithm 2. The algorithm calculates all possible failure mode M_j by taking the Cartesian product of the predicted labels in $\mathscr{Y}_{pred}$ and then proceeds to find the steady-state and transient state of the system, which may provide insights into the underlying causes of failure.

Note that the reliability of the mentioned approach for failure analysis depends on various factors, including the quality of data, the expertise of analysts, and the complexity of the system being studied. It also relies on a comprehensive database that covers a wide range of failure modes, with sufficient quantities to capture probabilities and system behavior. A more robust and comprehensive database leads to more informed and effective decision-making in the context of maintenance policies.

4.3 Case Study: Complex Hydraulic System

This section applies the proposed methodology to a realistic complex HS in section 1.5.1 which data from an experimental study on predictive maintenance is available in ([13]).

4.3.1 Setup of Experiments

Using the sensors signals listed in Table 1.1 as input, the MCMO model is able to simultaneously diagnose the multi-outputs of hydraulic components and the overall state of the HS described in Table 1.2, making it a powerful tool for decision-making problem.

In Figure 1.1, the state of the four hydraulic components and the overall state of the HS during the 2205 working cycles have been graphically represented, providing a comprehensive visual overview of the system's performance. For simplicity and dimensionality reduction, I used the average of time series $\mathscr{X}$ instead of processing the individual data points which can be useful in very specific situations. In this case, averaging the time series can help smooth out any noise or fluctuations in the data, making it easier to visualize trends or find patterns using ML algorithms. Then, to compare the performance of different classifiers, the dataset has been split into a training set (80%) to train the MCMO model and a testing set (20%) to evaluate the performance of the model. This split ratio is a commonly used ratio, although it may vary depending on the size of the dataset and the complexity of the problem being solved. For the sake of comparing the results achieved by the MCMO methods with those obtained by the baselines, accuracy, and precision are used to evaluate the performance of the models.

4.3.2 Results Analysis and Discussion

The results of the implementation of the proposed approaches for failure diagnosis in the HS can be evaluated in terms of accuracy and efficiency. The accuracy of the fault detection depends on the performance of the ML algorithm used for classification. Tables 4.1 and 4.2 show that the MCMO methods have better accuracy and precision in comparison with corresponding traditional methods. Among them, the use of an ensemble of decision trees in a Multi-Class Multi-Output Random Forest (MRF) can capture the non-linear and complex relationships between the input features and the output labels of the HS. This leads to better diagnosis performance compared to models that use linear or simpler non-linear models where the accuracy and precision for the cooler, valve, pump, accumulator, and HS are respectively, 100, 97.50, 99.77, 97.73, and 98.90. The MCMO model involves multiple target variables, which can lead to high-dimensional feature spaces. Therefore, understanding the relationships between input features and multiple output variables is more complex and challenging compared to traditional single-output classification models.

To evaluate the reliability of the complex HS, an FA is required to examine all possible FM in which the system may fail. However, in this case, the scope of the analysis is limited to the four main components, namely the cooler, valve, pump, and accumulator. Various hydraulic components and environmental effects may have an impact on the overall condition of the HS even though they were not taken into account in this analysis. Thus, the inspection and evaluation of the system's reliability are focused solely on these main components, while external factors and other components are excluded. To this aim, all possible combinations of FM are defined for the HS by taking only the four main components into consideration. These FM are then assigned to probabilities based on their likelihood of occurring. By calculating the probability of each FM, the most critical points of failure within the HS are identified, while facilitating the development of strategies to mitigate the corresponding risks. The results of this analysis can be seen in Table 4.3, which provides a clear picture of the system's reliability and identifies areas that require improvement.

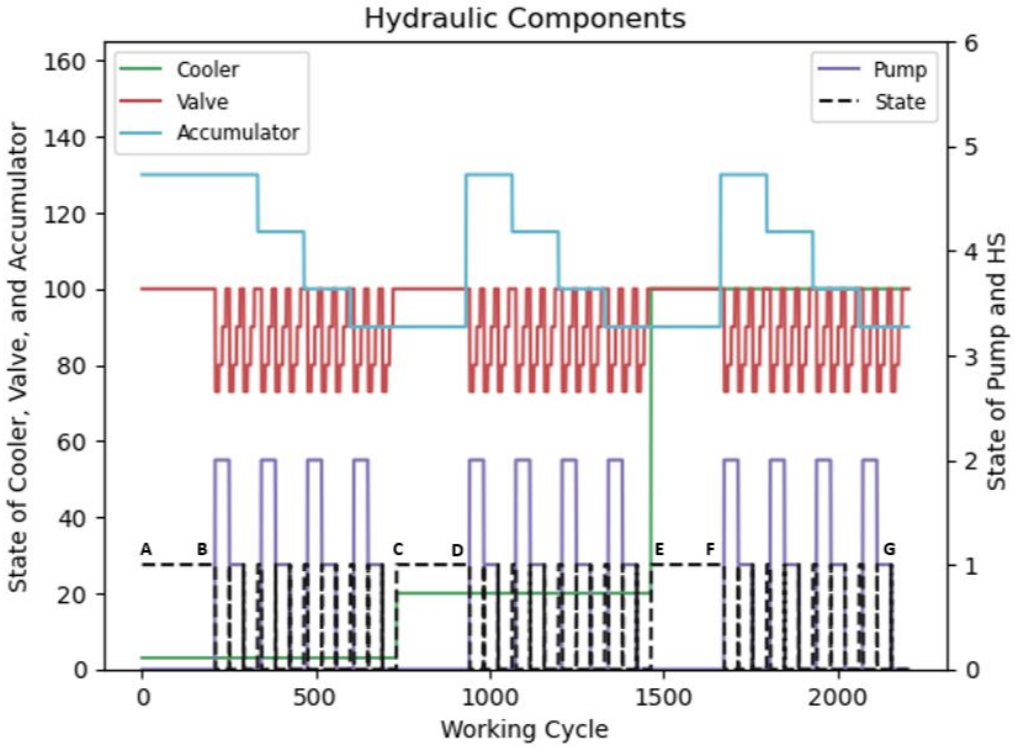


Fig. 4.2 Degradation of hydraulic components and stable state of the hydraulic system.

Table 4.1 Accuracy of component level through MCMO classification model.

Model	Cooler	Valve	Pump	Accumulator	HS
LR	99.77	71.65	98.86	61.22	92.97
KNN	99.83	84.12	97.95	94.55	92.06
SVC	99.83	48.75	95.01	57.14	92.97
DT	99.83	94.10	58.27	95.23	91.60
RF	99.83	96.37	58.27	97.73	91.83
MLR	100	50.79	75.96	41.95	87.30
MKNN	100	73.69	98.18	87.07	95.01
MSVM	53.51	46.48	54.42	36.50	68.70
MDT	100	94.10	99.77	95.23	97.27
MRF	**100**	**97.50**	**99.77**	**97.73**	**98.86**

Table 4.2 Precision of component level through MCMO classification model.

Model	Cooler	Valve	Pump	Accumulator	HS
LR	99.77	77.30	98.89	58.86	84.48
KNN	99.83	84.46	98.03	94.60	87.17
SVC	99.83	36.54	95.80	65.68	84.48
DT	99.83	94.09	43.71	95.24	88.00
RF	99.83	96.37	58.27	97.73	86.62
MLR	100	81.79	78.58	59.38	87.27
MKNN	100	74.55	98.24	87.18	95.06
MSVM	55.46	25.00	33.33	25.00	50.0
MDT	100	94.14	99.77	95.29	97.27
MRF	**100**	**97.59**	**99.77**	**99.73**	**98.90**

Table 4.3 Failure probability (FP) for all failure modes (FM) of the HS.

FM	FP(%)	FM	FP(%)	FM	FP(%)	FM	FP(%)
$C_1V_4P_1A_1$	0.13	$C_1V_4P_1A_2$	0.13	$C_1V_4P_1A_3$	0.13	$C_1V_4P_1A_4$	0.13
$C_2V_4P_1A_1$	0.13	$C_2V_4P_1A_2$	0.13	$C_2V_4P_1A_3$	0.13	$C_2V_4P_1A_4$	0.13
$C_3V_4P_1A_1$	0.13	$C_3V_4P_1A_2$	0.13	$C_3V_4P_1A_3$	0.13	$C_3V_4P_1A_4$	0.13
$C_1V_4P_2A_1$	0.13	$C_1V_4P_2A_2$	0.13	$C_1V_4P_2A_3$	0.13	$C_1V_4P_2A_4$	0.13
$C_2V_4P_2A_1$	0.13	$C_2V_4P_2A_2$	0.13	$C_2V_4P_2A_3$	0.13	$C_2V_4P_2A_4$	0.13
$C_3V_4P_2A_1$	0.13	$C_3V_4P_2A_2$	0.13	$C_3V_4P_2A_3$	0.13	$C_3V_4P_2A_4$	0.13
$C_1V_4P_3A_1$	1.45	$C_1V_4P_3A_2$	1.45	$C_1V_4P_3A_3$	1.45	$C_1V_4P_3A_4$	27.91
$C_2V_4P_3A_1$	27.91	$C_2V_4P_3A_2$	1.45	$C_2V_4P_3A_3$	1.45	$C_2V_4P_3A_4$	1.45
$C_3V_4P_3A_1$	27.91	$C_3V_4P_3A_2$	1.45	$C_3V_4P_3A_3$	1.45	$C_3V_4P_3A_4$	1.45

Table 4.4 HS Steady State. (**F**: failure; **H**: non-failure or healthy)

Steady-state	Cooler	Valve	Pump	Accumulator	HS
[AB]	F	H	H	H	SF_1
$[CD]$	$C_{i=2}$	H	H	F	SF_1
[EF]	H	H	H	F	SF_1

Table 4.5 HS Transient State. (**F**: failure; **H**; non-failure or healthy)

Transient State	Cooler	Valve	Pump	Accumulator	HS
$[BC]$	$C_{i=1..3}$	$V_{i=1..4}$ $P_{i=1}$	F	$A_{i=1..4}$	SF2
$[DE]$	$C_{i=1..3}$	$V_4 \rightarrow V_1$	$P_{i=2..3}$	$A_{i=1..4}$	SF1
$[FG]$	$C_{i=1..3}$	H	H	$A_i \rightarrow A_{i-1}$	SF1

In Table 4.3, if valve V_4 is substituted with either V_1, V_2, or V_3 in all FMs, then the resulting FP will be equal to zero. Table 4.3 shows that it is not uncommon for the HS to fail even when individual components such as coolers, valves, pumps, and accumulators are in optimal conditions (e.g., $C_3V_4P_3A_4$). This is due to two main reasons. First, some issues on the overall system design or configuration could occur, such as incorrect sizing or installation of components, or inadequate maintenance and monitoring of the system. Second, environmental effects or interactions and dependencies between components are not captured by the individual sensor measurements used for diagnosis. As a consequence, it is evident that the dataset related to the case study HS does not embed all failure system information. Alternatively, a thorough analysis of the system's design and operation may be conducted to identify any redundancies, backup systems, or alternative pathways that may allow the system to continue functioning despite component failures.

To this aim, Tables 4.4 and 4.5 respectively present the steady-state and transient-state conditions of the HS, highlighting the stable and consistent operation of the system under normal conditions [AB], [CD], and [EF] (see Figure 4.2), as well as its response to changes (transient-state [BC], [DE], and [FG]). Regardless of the cooler state (i.e., it is failed or at full efficiency), the system exhibits the same behavior pattern for [BC], [DE], and [FG] (see Figure 4.2), thus indicating that the cooler is not a critical component.

Comparing Table 4.3 with Tables 4.4 and 4.5, the dynamic nature of the HS's failure behavior is clearly evident, as opposed to being primarily static. Table 4.4 shows the critical component that leads to system failure, whereas Table 4.5 reveals that HS is failed with a sequence or a combination of events. A basic fault tree can model HS failure mode in the first case but not in the second one. Therefore, additional modeling techniques are needed to capture the system's behavior. Among them, dynamic fault trees and, stochastic Petri net which is an extension of basic fault trees, and also Markov models are potential probabilistic tools that can be used for this purpose.

4.4 Conclusion of Multi-Class Multi-Output Classification and Failure Analysis

For complex systems, a fault of one or several components does not necessarily lead to a failure of the system, but if the failed components are not immediately replaced, they may conduct some other components to an idle state. In this work, a data-driven model with a two-step decision approach is proposed to provide a comprehensive analysis of the potential failures and their causes. In the first step, a Multi-Class Multi-Output (MCMO) classification technique is used to diagnose potential failures based on sensor signals, and, in the second step, Failure Analysis (FA) is applied to investigate the root causes of those failures. The proposed approach is applied to a multi-component HS case study, showing the resulting effectiveness in improving system reliability, reducing downtime, and minimizing the impact of failures on system operations. The results show that MCMO classification is a promising approach for multi-component system failure diagnosis that offers several advantages over conventional methods.

Chapter 5

Semi-Supervised Methods: An Approach for Fault Detection and Diagnosis in Complex Mechanical Systems

5.1 Introduction to Semi-Supervised Learning

In this chapter, graph-based SSL relying on LP is combined with conventional classification algorithms in order to detect potential faults in complex and multi-component mechanical systems. Experimental results on realistic systems from the related literature shows that the proposed method can effectively enlarge the labeled datasets and interestingly identify different types of failures and faults with higher accuracy compared to baseline methodologies.

The rest of this chapter is organized as follows. Section 6.2 describes the SSL method aimed at classifying the eventual fault types that may occur during the working cycle of a system. In Section 6.3, two case studies are presented: first, a real HS is described, and the corresponding sensors' data are introduced; then, the pneumatic system and corresponding data are introduced as the second scenario. Subsequently, the proposed methodology is applied to the SSL of the state of the main HS components and PS, and the achieved results are comprehensively discussed

Algorithm 3: Semi-Supervised Learning (SSL) Algorithm

Inputs: X_L,X_U, Y_L, X_T
Outputs: Y_U, Y_T

1: Y_U = LabelPropagation(X_L, Y_L, X_U)
2: Train the Classifier Model using (X_L, X_U, Y_L, Y_U)
3: Y_T = Classifier(X_T)
4: **procedure** Y_U = LABELPROPAGATION(X_L, Y_L, X_U)
5: Compute normalized transition matrix $\bar{T}$
6: Initialize Y_L in accordance with Y_L
7: Compute Y_U through (5.6)
8: Extract Y_U from Y
9: **end procedure**

and compared with those obtained by baseline methods. Lastly, concluding remarks and outlooks for future works are summarized in Section 6.4.

5.2 The proposed Semi-Supervised Leaning methodology

The proposed graph-based SSL using LP is described in Algorithm 3. The labeled input X_L, the corresponding labeled output Y_L, and the unlabeled input X_U are the three main inputs of the SSL algorithm. The preliminary part of the algorithm is the LP function that uses X_L and Y_L, to estimate the unlabeled output Y_U corresponding to X_U to improve and enlarge the training data for the ML classifier, which is the core part of the SSL. The procedure data flow is schematized in Fig. 5.1.

5.2.1 Inputs Definition

Let us consider a collection of N training examples $X_t = (x_1,\ldots,x_L,x_{L+1},\ldots,x_N)$, each being determined by D features collected in the set $\mathscr{D} = \{\delta_1,\ldots,\delta_d,\ldots,\delta_D\}$. The first L examples x_i with $i \in \{1,\ldots,L\}$, denoted by X_L, are labeled according to a discrete label set $\mathscr{C} = \{\gamma_1,\ldots,\gamma_c,\ldots,\gamma_C\}$ composed of C classes, thus being associated to labels $Y_L = (y_1,\ldots,y_L)$ with $y_i \in \mathscr{C}$. The remaining $U = N - L$ examples x_i with $i \in \{L+1,\ldots,N\}$, denoted by X_U, are unlabeled. Moreover, M examples $X_T = (x_{N+1},\ldots,x_{N+M})$ are unseen (unlabeled) testing data points.

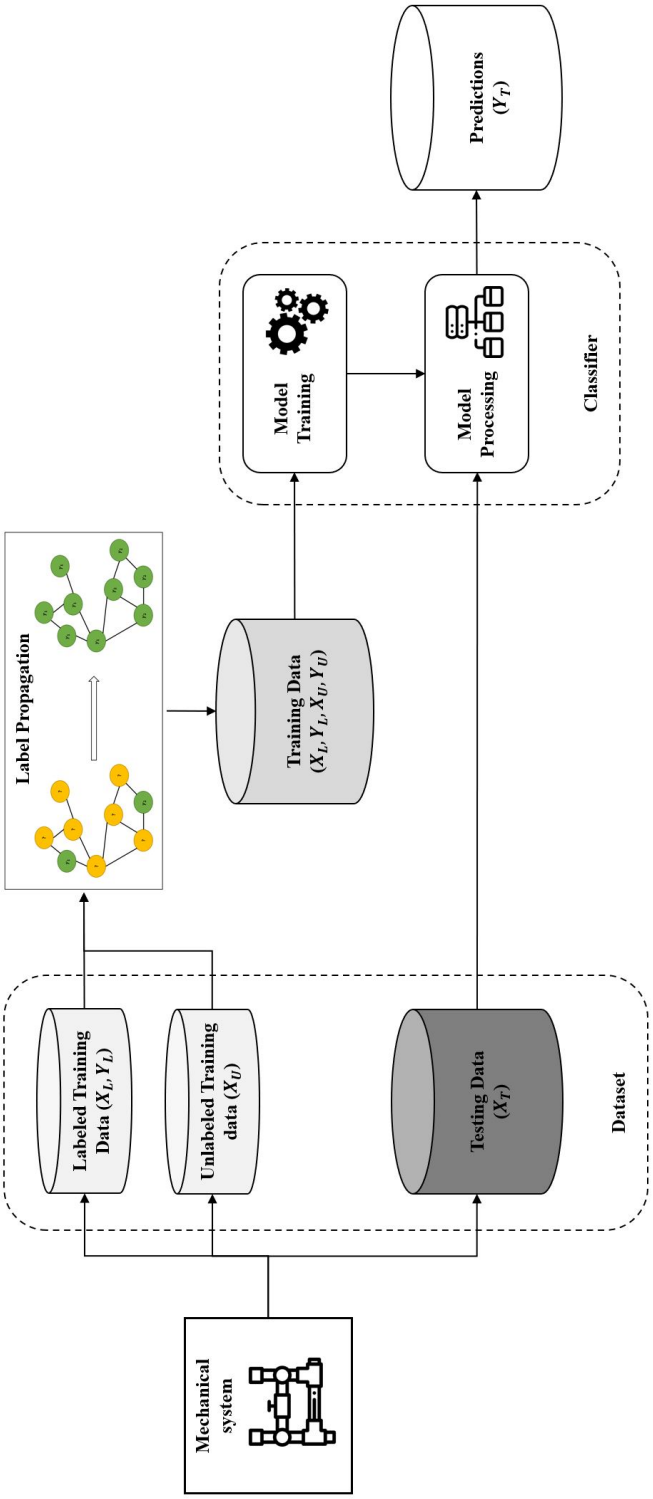


Fig. 5.1 Scheme of graph-based transductive SSL.

5.2.2 Label Propagation

The initial stage of the proposed SSL consists in taking all examples X_L and corresponding labels Y_L to estimate the labels Y_U corresponding to the remaining examples X_U (Algorithm 1, line 1). As can be seen in Fig. 5.1, this approach allows the training dataset to grow progressively, allowing for improved model performance with more data.

In general, LP is predicated on the notion that similar data points are probably part of the same class and that class labels of a small group of labeled data points can be used to predict class labels of the unlabeled data points [85]. In LP, the class labels of the labeled data points are used as initial estimates. Through a series of iterations, the labels are subsequently propagated to the unlabeled data points. During each iteration, the class labels of the unlabeled data points are updated based on the labels of their closest neighbors. This process continues until convergence, at which point the class labels of the unlabeled data points are considered to be the final class labels. To do this, LP constructs a fully linked graph in which all of the labeled and unlabeled data points are the nodes. There is a weighted edge between the two nodes. The weight increases as the Euclidean distance between two nodes decreases. In particular, the edge connecting any two nodes a,b is weighted by w_{ab} defined as follows:

$$w_{ab} = e^{-\frac{\Sigma_{\delta_d \in \mathscr{D}} \left(x_a^{\delta_d} - x_b^{\delta_d} \right)^2}{\sigma^2}} \tag{5.1}$$

where σ is a control parameter that determines the scale of the similarity. Note that the exponential function in (5.1) transforms the Euclidean distance into a similarity measure that ranges from 0 (no similarity) to 1 (maximum similarity). I allow a node label to propagate to all other nodes through the edges. Larger edge weights allow labels to travel through the graph more easily. Let us define a $(L+U) \times (L+U)$ probabilistic transition matrix T whose element a,b –denoting the probability to jump from node a to b– is computed as follows:

$$T_{ab} = \frac{w_{ab}}{\sum_{r=1}^{L+U} w_{rb}}, \forall a,b = 1,\ldots,L+U. \tag{5.2}$$

Matrix T is straightforwardly converted into the normalized matrix $\bar{T}$, whose element a,b is determined through a normalization step:

$$\bar{T}_{ab} = \frac{T_{ab}}{\sum_{r=1}^{L+U} T_{ar}}, \forall a,b = 1,\ldots,L+U \tag{5.3}$$

In turn, matrix $\bar{T}$ can be splitted into 4 sub-matrices:

$$\bar{T} = \begin{bmatrix} \bar{T}^{LL} & \bar{T}^{LU} \\ \bar{T}^{UL} & \bar{T}^{UU} \end{bmatrix} \tag{5.4}$$

where sub-matrices $\bar{T}^{UL}$, $\bar{T}^{LU}$ and $\bar{T}^{UU}$ represent the similarity between the unlabeled data points and the labeled data points, and between the unlabeled data points, respectively.

Moreover, let us define a $(L+U)\times C$ label matrix Y, whose a-th row contains the probability that node x_a has class γ_c for each $c = 1,\ldots,C$. The upper L-row block and the lower U-row block portion of matrix Y are denoted as Y_L and Y_U and are related to the labeled and unlabeled training data, respectively. It is not crucial to initialize the rows of Y_U since they correspond to the unlabeled data points. Following [85], the following iterative scheme allows to compute the labels of data point X_U:

$$\mathrm{Y} \leftarrow \bar{T}\mathrm{Y}. \tag{5.5}$$

Due to the clamping (i.e., fixing the labels of data points, typically the labeled data points), Y_L never changes, therefore Y_U is the only variable to be computed. As a consequence, (5.5) can be written as:

$$\mathrm{Y}_U = (I - \bar{T}_{UU})^{-1} \bar{T}_{UL}\mathrm{Y}_L \tag{5.6}$$

where I represents the identity matrix and $(I - \bar{T}_{UU})$ is invertible as long as there are no disconnected components in the similarity graph (i.e., all data points are connected to at least one other data point).

5.2.3 Training and Classification

The classifiers are then trained using the X_L, X_U, Y_L, and Y_U to finally map previously unseen samples X_T to class labels Y_T (Algorithm 3, line 2). The SSL method uses both labeled training (X_L, Y_L) and estimated data (X_U, Y_U) to learn the classification model. Finally, the trained model is used on the testing portion of the dataset (X_T) in order to predict the testing labels Y_T (Algorithm 3, line 3).

5.2.4 Implementation Details and Computational Considerations

The proposed algorithm was implemented in Python on a Lenovo Z400 laptop with an Intel Core i7 processor. The implementation leveraged the scikit-learn library for training the classifier and utilized the LabelPropagation package for the label propagation step. The preprocessing steps included standard data cleaning and normalization procedures. The algorithm was executed on a standard laptop, and while specific computation times may vary depending on the dataset size and complexity, the Intel Core i7 processor provided ample computational power for the task. The implementation demonstrated the feasibility of GSSL on the given hardware, showcasing the versatility of the scikit-learn for such ML tasks.

5.3 Case Study and Numerical Experiments: Complex Hydraulic and Pneumatic Systems

This section performs the FDD for two mechanical systems, a single-component PS and a multi-component HS, using the proposed SSL approach.

The first case study addresses a single-component PS deployed in a real industrial process in 1.5.1. This system refers to the robotic work-cell that has been produced by an Italian automotive company leader in the European market. Conversely, the second case study concerns a real complex and multi-component HS, whose predictive maintenance experimental data set is available in 1.5.2. The observable data are related to the failures due to different causes across different working cycles.

5.3.1 Data Pre-processing and Experimental Setup

Data pre-processing is necessary prior to any data-driven procedure being performed. The accuracy of classification techniques is, in fact, significantly impacted by the quality of the training data set. In this case study, the most significant components of the PS and HS for FDD are selected based on a quality measure. To do so, a quality measure has been applied on X_L and Y_L to find the most relevant features to the system fault.

In general, a quality measure Quality(d_i, Y_L) evaluates how suitable a feature $d_i \in \mathscr{D}$ is for predicting the value of the class attribute Y_L, relying on distributions over feature values and target values. Gain Information (GI) is a popular filter model and quality measure used in feature weight scoring. It is defined as:

$$\text{GI}(\mathscr{D}, d_i) = \text{Entropy}(\mathscr{D}) - \sum_{v \in \text{ Values}(d_i)} \frac{|\mathscr{D}_v|}{|\mathscr{D}|} \text{Entropy}\,(\mathscr{D}_v) \tag{5.7}$$

where Values(d_i) denotes the set of all possible distinct values that d_i can take in $\mathscr{D}$, $\mathscr{D}_v$ refers to the subset of $\mathscr{D}$ for which the attribute has the value v, and the entropy operator represents a measure of the pureness of data as follows:

$$\text{Entropy}(\mathscr{D}) = \sum_{\gamma_c \in \mathscr{C}} -p_c \log_2 (p_c) \tag{5.8}$$

being p_c the relative frequency of class γ_c in $\mathscr{D}$ GI is used to find the main failure indicators (MFIs) which are signs or symptoms that indicate a potential failure or problem in a system. MFIs can be monitored using a variety of techniques, including sensors, data analysis tools, and human observation. The results of the monitoring can be used to identify trends and patterns and to prioritize maintenance and risk mitigation activities. The MFIs for the PS data have been obtained for six features $D = \{$MaxFlowA, MaxFlowB, IFRA, IFRB, TimeA, TimeB$\}$ where the value of GI is 0.33731, 0.31845, 0.30832, 0.21980, 0.11030, and 0.09469 for MaxFlowA, MaxFlowB, IFRA, IFRB, TimeA, and TimeB, respectively. Moreover, the correlation matrix in Fig. 5.2 shows that the MaxFlowA (-0.43) and MaxFlowB (-0.46) are the two most correlated variables to the final state of the PS, meaning that changes in these features are strongly associated with changes in the final state of the single-unit PS. Similarly in the case of the HS dataset, where $D = \{C1, V10, MP1, A1 - A4\}$,

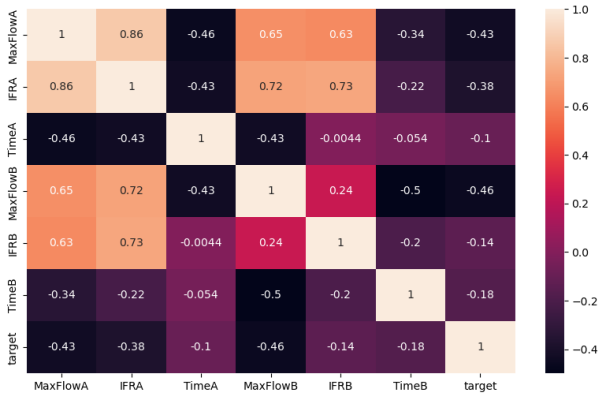


Fig. 5.2 Correlation matrix for the PS dataset.

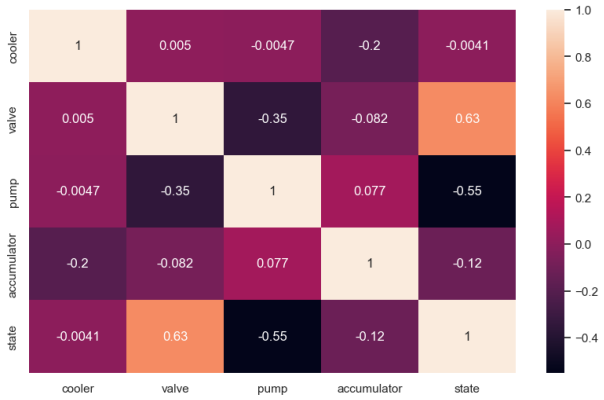


Fig. 5.3 Correlation matrix for the HS dataset.

MFIs are the most correlated components to each failure type which are evaluated according to (5.7). The results show that the value of GI for the valve $V10$, pump $MP1$, accumulator $A1-A4$, and cooler $C1$ are 0.46177, 0.31588, 0.07358, and 0.00012, respectively. This means that the valve $V10$ is the MFI in the HS when a failure occurs. On the other hand, the cooler has less effect on the final state prediction of the HS (the larger the GI of an attribute, the better the attribute for building the prediction model). Additionally, the correlation matrix in Fig. 5.3 shows the pairwise correlations between different components in the HS dataset. The values in this matrix range from -1 to 1, where -1 indicates a perfect negative correlation, 0 indicates no correlation, and 1 indicates a perfect positive correlation. Since the correlation matrix shows that the valve $V10$ (0.63) and pump $MP1$ (-0.55) are the two most correlated components to the final state of the HS, it suggests that changes in these components are strongly associated with changes in the final state of the HS.

5.3.2 Comparison with Baseline Methods

The proposed SSL models are applied independently to the PS and HS scenarios after being developed and adapted in Python. As per normal, two equal-sized subsets are created from the datasets that include both the features and the accompanying labels: the training (X_t) and the testing (X_T) set. Then, the training portion is divided into two subsets: the labeled (X_L, Y_L) and unlabeled (X_U) training data, where the size of the first subset is generally smaller than the latter. Using the LP to estimate the classes Y_U corresponding to the unlabeled training samples X_U, the SSL classification models are trained over (X_L, Y_L, X_U, Y_U). Various traditional classification algorithms are implemented, including Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), Naive Bayes (NB), and Random Forest (RF).

The accuracy, F1 score, precision, and recall are used to evaluate the performance of the models in order to compare results acquired by the SSL forecasting technique with those of the multi-class baselines. The accuracy of a classification task is calculated based on the so-called confusion matrix (CM), also known as the error matrix. By definition, a CM is a square matrix whose dimension corresponds to the number C of classes and whose element $\mathrm{CM}(k,p)$ is equal to the number of observations known to be in class γ_k (actual class) and predicted to be in class γ_p (predicted class). Accuracy is the ratio of the number of correct predictions to the total number of predictions made by the model. In multi-class SSL, accuracy over the (X_T, Y_T) is the percentage of correctly classified samples over all classes and is defined as follows:

$$\text{Accuracy} = 100\,\frac{\sum_{k=1}^{C}\mathrm{CM}(k,k)}{\sum_{k=1}^{C}\sum_{p=1}^{C}\mathrm{CM}(k,p)}. \tag{5.9}$$

If the entire set of predicted labels for a data set strictly matches the true set of labels, then the accuracy achieves the highest value, i.e., 100%. F1 score balances precision and recall in the assessment of model performance. Precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions.

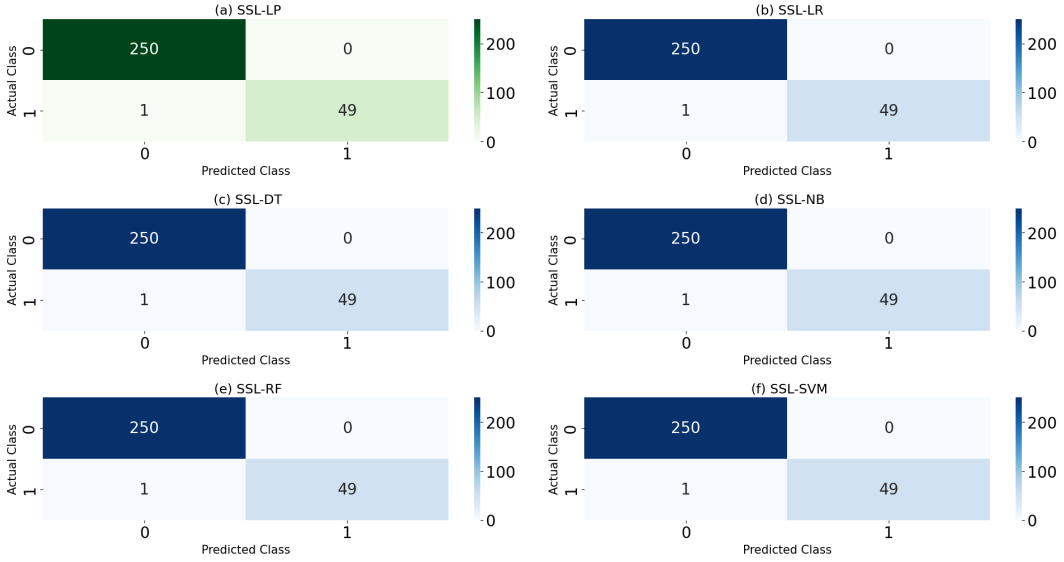


Fig. 5.4 Confusion matrix for (a) SSL-LP, (b) SSL-LR, (c) SSL-DT, (d) SSL-NB, (e) SSL-RF, and (f) SSL-SVM, considering the valve $V10$ and pump $MP1$ as MFIs for $X_L(5\%)$ and $X_T(50\%)$

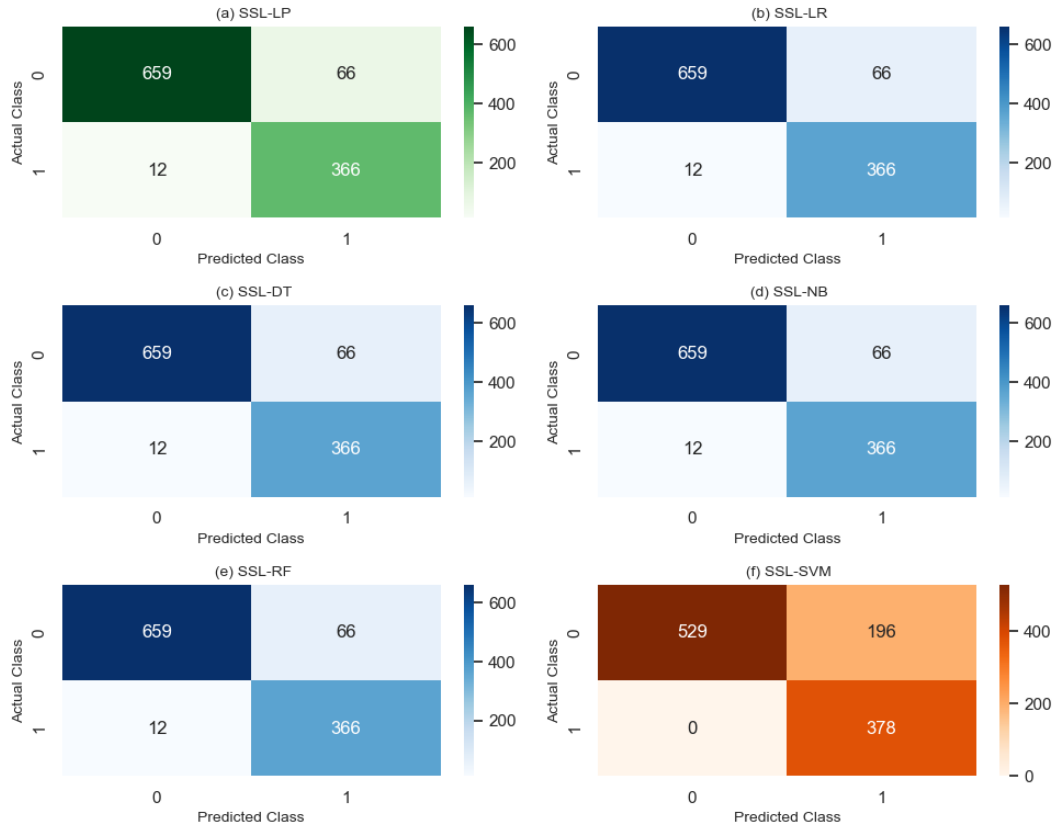


Fig. 5.5 Confusion matrix for (a) SSL-LP, (b) SSL-LR, (c) SSL-DT, (d) SSL-NB, (e) SSL-RF, and (f) SSL-SVM, considering MaxFlowA and MaxFlowB as the MFIs for $X_L(5\%)$ and $X_T(50\%)$

5.3.3 Results and Discussion

First, the proposed SSL model is first quantitatively evaluated by the CM over the testing dataset (X_T). The numerical experiments conducted on the two industrial datasets demonstrate that the SSL performance is influenced by several factors. In general, the MFIs in PS and HS data can interestingly affect the performance of SSL. The similarity metric (i.e., Euclidean distance) used to define the connections between data points can have a significant impact on the LP. The performance of LP is dependent on the number of labeled data points. As the number of labeled data points increases, the performance of the LP method and consequently the SSL improves. The quality of the labeled data can also have a significant impact on the performance of LP. If the labeled data is noisy or contains errors, the performance of the method may be negatively impacted.

As can be seen in the figure (Fig. 5.4), the SSL method classifies the faulty/healthy condition of the PS with high accuracy, since only one sample is misclassified. Moreover, the accuracy, $F1$ score, precision, and recall are respectively equal to 99.667, 99.935, 99.801, and 99.00 for both baseline and SSL methods, while MaxFlowA and MaxFlowB are the most significant feature for fault diagnosis in the PS dataset. The PS data set is well-separated and dense, therefore traditional ML models can often perform well without the need for SSL or additional labeled data. This is because traditional models are designed to work with labeled data and are often able to generalize well when the labeled data is representative of the underlying samples. In order to show the effectiveness of the LP, the corresponding results are plotted in Fig. 5.6, where the green and orange points are labeled and unlabeled training data of the pneumatic work-cell system, respectively (Fig. 5.6a). After applying LP, the unlabeled data are labeled, and the estimated labels are linked with labeled data (Fig. 5.6b). However, even in these cases, there may be some benefits to using SSL. For example, incorporating additional unlabeled data into the model can help it to better capture the underlying patterns and relationships in the data. This can be useful for maintaining system reliability and reducing the likelihood of unexpected failures.

It is apparent that the diagonal values of the confusion matrices for the HS (Fig. 5.5) are much larger than the off-diagonal values, meaning that the SSL method perfectly classifies the failure/healthy condition of each working cycle in the HS. Through a series of experiments conducted on Tables 5.1 and 5.2 to 5.3 and 5.4, I

Table 5.1 Accuracy (%) of the baseline methods with training data $X_L(25\%)$ for different MFIs

ComponentModel	SVM	LR	DT	NB	RF
$\{V10\}$	82.230	82.230	82.230	82.230	82.230
$\{MP1\}$	77.425	77.425	77.425	77.425	77.425
$\{V10, MP1\}$	82.230	92.928	92.928	87.761	92.928
$\{C1, V10,$ $MP1, A1-A4\}$	81.868	92.928	92.566	87.761	92.747

implemented a systematic reduction in X_L percentage from 25% to 5% while holding the X_T percentage constant at 50% across all runs. The overall accuracy of the models for the HS dataset can be seen in Tables 5.1 and 5.2. In Tables 5.3 and 5.4, by tuning the SSL model and taking MFIs into consideration, I found out that the most important components for failure detection of HS are the valve $V10$ and pump $MP1$. In Tables 5.3 and 5.4, the overall performance of the models for the HS dataset was evaluated using metrics such as accuracy, F1 score, precision, and recall. In this case, I trained and tested the learning models over the whole feature set while the size of labeled training data ($X_L = 5\%$) was much less than unlabelled data ($X_U = 45\%$).

Regarding Table 5.1 to 5.4, it seems that generally, SSL methods with estimated labels have better accuracy in comparison with real labels. To find the reason behind this difference, I explored the training data, and I found out that by learning on a fixed set of data that generates a given model, algorithms can quickly become ineffective or even counterproductive. This problem could occur because of the modification of data or the occurrence of new data constantly. I can claim that LP solves this problem during label estimation in the graph-based model. Consequently, the SSL algorithms have a better performance in some cases. In general, the overall accuracy of the SSL methods is equal to or higher than conventional supervised methods such as SVM and LR in FDD problems in both the PS and HS case studies. On the other hand, when I employ the MFIs related to the target states, LP produces better labels for the unlabeled training portion of data and consequently enhances the performance of the SSL method.

Summing up, three main outcomes are apparent: on the one hand, the applied SSL method is able to assign fault sets to the corresponding failure types with high accuracy; on the other hand, it can label incoming fault sets, where most of the data are unlabeled. Interestingly, by considering MFIs in the SSL process (see Table 5.3

Table 5.2 Accuracy (%) of the SSL-methods with training data $X_L(25\%) \cup X_U(25\%)$ for different MFIs

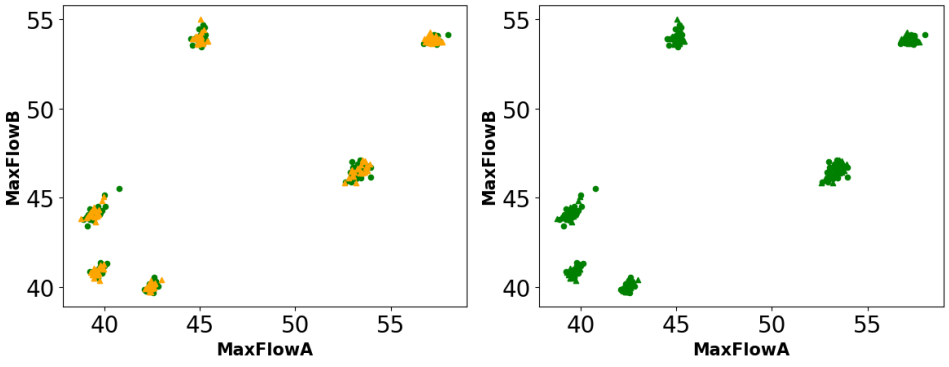
ComponentModel	**SSL-SVM**	**SSL-LR**	**SSL-DT**	**SSL-NB**	**SSL-RF**
$\{V10\}$	82.230	92.928	92.928	92.928	92.928
$\{MP1\}$	82.230	92.928	92.928	92.928	92.928
$\{V10, MP1\}$	82.230	92.928	92.928	92.928	92.928
$\{C1, V10,$ $MP1, A1-A4\}$	83.409	92.928	92.566	92.928	92.566

Table 5.3 Accuracy, F1 Score, Precision, and Recall (%) of the baseline methods with training data $X_L(5\%)$, considering $\{C1, V10, MP1, A1-A4\}$ as the feature set

MetricModel	SVM	LR	DT	NB	RF
Accuracy	75.431	92.112	92.566	92.928	92.566
F1 Score	71.961	91.569	91.739	92.391	91.739
Precision	72.755	90.588	91.792	91.467	91.792
Recall	71.434	93.367	91.686	93.861	91.686

Table 5.4 Accuracy, F1 Score, Precision, and Recall (%) of SSL-methods with training data $X_L(5\%) \cup X_U(45\%)$, considering $\{C1, V10, MP1, A1-A4\}$ as the feature set

MetricModel	SSL-SVM	SSL-LR	SSL-DT	SSL-NB	SSL-RF
Accuracy	83.409	92.928	91.024	92.928	91.024
F1 Score	82.912	92.391	89.928	92.391	89.928
Precision	83.018	91.467	90.413	91.467	90.413
Recall	86.620	93.861	89.500	93.861	89.500



(a) (b)

Fig. 5.6 Dataset of the PS: labeled (green) and unlabeled (orange) data points before (a) and after (b) label propagation.

and 5.4), the model can improve its performance by taking into account the specific factors that are most likely to cause it to fail. Based on my analysis of both PS and HS, it is evident that the rate of misclassified samples in comparison to the overall failure data is significantly low (figures 5.4 and 5.5), suggesting that the number of maintenance actions required in both the original and enlarged datasets may not differ significantly. In general, the proposed SSL models can be applied to both fault diagnosis of single components and fault detection of multi-component systems in a wide range of applications in various industries.

5.4 Conclusion of Graph-based Semi-Supervised Method based on Label Propagation

The integration of artificial intelligence in mechanical FDD helps to increase reliability, reduce costs, and improve the overall performance of mechanical systems in Industry 4.0 applications. Most interesting industrial applications nowadays come from dynamic environments where data are generated continuously over time and where the labeled data are scarce and expensive. Therefore, semi-supervised learning (SSL) can be particularly useful in FDD because faults may be rare or difficult to identify, and may not be fully represented in the labeled data. By using a combination of labeled and unlabeled data, SSL can help to identify these rare or difficult-to-detect faults, leading to a more effective FDD. In this paper, graph-based SSL relying on label propagation is combined with conventional classification algorithms to detect potential failures in complex mechanical systems. Experimental results on realistic pneumatic and hydraulic systems from the related literature show that the proposed method can effectively enlarge the labeled datasets and interestingly identify different types of non-nominal conditions with higher accuracy compared to baseline methodologies.

Chapter 6

Unsupervised Methods: An Adaptive Constrained Clustering Approach for Real-Time Fault Detection of Industrial Systems

6.1 Introduction to Unsupervised Learning and thesis contribution

In this chapter, a partitioned-based DS clustering approach is used to perform a fault detection strategy in a real-time fashion, with the final aim of discriminating the nominal and non-nominal working conditions of a real industrial machine. More specifically, the proposed method relies on a cyclical Adaptive Constrained Clustering algorithm, which is composed of two steps: *micro-clustering* and *constrained macro-clustering*. The DS is partitioned over a finite number of batches (also known as windows) of data. In the micro-clustering step, data are grouped into several so-called micro-clusters, which are thus dynamically updated over time. This represents a big challenge in DS mining, usually done by means of common clustering methods (e.g., K-means), where however no control over the final clusters is guaranteed. Moreover, in the DS mining context, a challenging issue arises when some constrained-based learning approaches must be applied to clusters which, in turn, are formed over time. To cope with such an issue, additional must-link and

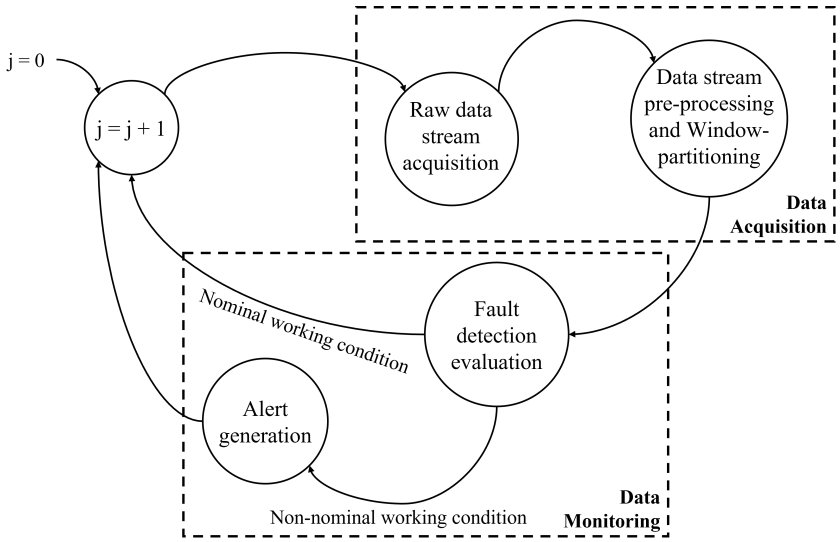


Fig. 6.1 Work-flow of the real-time fault detection approach.

cannot-link constraints are thus taken into account during the macro-clustering step in an innovative fashion, while dealing with the vectors of cluster features related to each generated micro-cluster. Then, leveraging the information acquired at the previous stage, the composition of macro-clusters is dynamically updated, so that the incoming monitoring data are grouped into two clusters over time, representing the nominal and non-nominal work conditions, respectively. Differently from the related literature where unsupervised methods are typically employed in DS mining, the proposed approach combines the use of a large amount of unlabeled data with a limited quantity of additional information, thus shifting the benefits of the offline-constrained clustering to the real-time scenario.

The rest of the chapter is organized as follows. Section 6.2 describes the fault detection system framework. The proposed algorithm is illustrated in Section 6.3. Section 6.4 is focused on the numerical experiments based on a real case study: the achieved results are comprehensively discussed and compared against those obtained with other static-based clustering methods. Lastly, concluding remarks and outlooks to future works are summarized in Section 6.5.

6.2 Fault detection system framework

Let us consider an industrial process with a repeated work-cycle, where a number of parameters are periodically monitored through the measurements collected by faultless sensors installed on the machine. It is worthwhile noting that the faultless hypothesis for the sensor results in assuming the retrieved data is accurate and reliable, without any errors or faults. Those parameters are used to perform an analysis of the work conditions of the industrial system. More specifically, the main goal is to define a real-time detection methodology aimed at detecting whether the industrial system is nominally working or any faulty condition is occurring. The flow-diagram showing the high-level activities performed by the proposed real-time fault detection approach is illustrated in Fig. 6.1. The overall framework comprises two main stages to be conducted for each iteration j corresponding to a given time window: *Data Acquisition* and *Data Monitoring*. The former is focused on acquiring a continuous stream of raw data from the industrial process, whereas the latter performs mining activities aimed at extracting information useful for determining the system's working state. Hence, during the Data Acquisition step, a continuous DS is retrieved by sensors, and, being properly pre-processed and rearranged, data are analyzed during the Data Monitoring step. As long as the process is working in accordance with the nominal behavior, no further action is taken; conversely, if any faulty condition occurs, an alert arises and eventual subsequent PHM actions are immediately triggered.

For the sake of handling the incremental stream of data, the single-pass paradigm is considered, according to which an instance of parameters that have been already processed cannot be processed again (e.g., random access is not allowed) [64, 86, 87]. Suppose that the monitoring information of the work-cycle i is stored in an m-dimensional vector $\vec{X}_i$. Due to memory and computational effort limitations, only a finite group of data extracted from the input stream is analyzed at a time. In particular, the stream of data is divided into equal *batches* (or *windows*) containing B data points, in accordance with the so-called landmark window model [64, 78]. As a consequence, the fault detection procedure is iterated by a sampling period $W = BT$, where T denotes the cycle time. In the j-th time window only the B instances $\vec{X}_i$ related to the cycle times iT belonging to the window $[(j-1)W, jW]$ are considered, as follows:

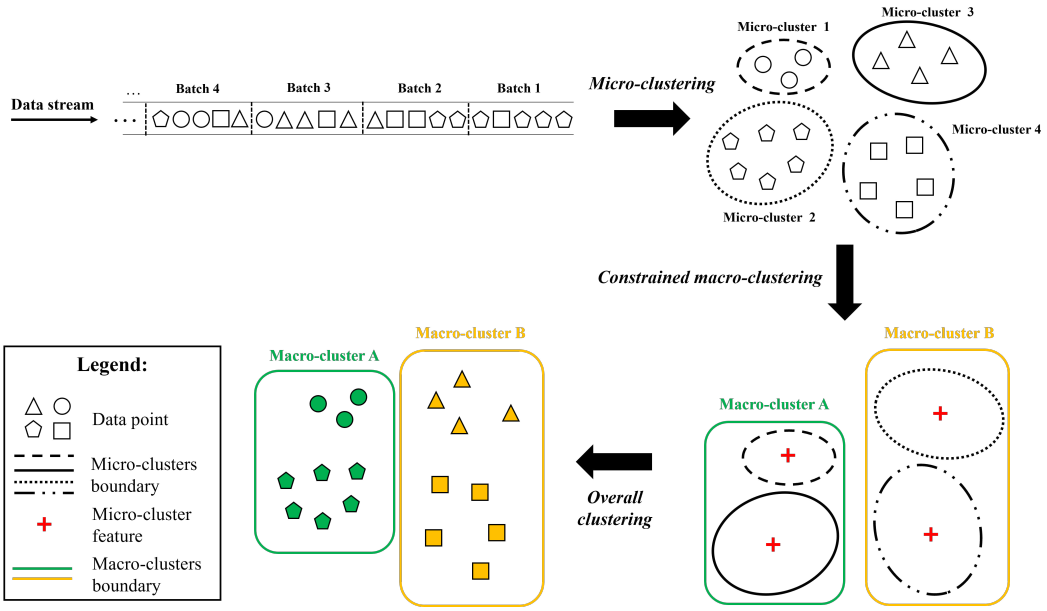


Fig. 6.2 Scheme of the proposed Adaptive Constrained Clustering Algorithm.

$$D_j = \{\vec{X}_i, \ldots, \vec{X}_{i+B-1}\}, \forall j = 1, 2, \ldots \tag{6.1}$$

In this way, it is ensured that two consecutive windows D_j and D_{j+1} ($\forall j = 1, 2, \ldots$) do not overlap, resulting in allowing real-time data processing. Therefore, lower memory usage and, at the same time, a faster run-time than traditional static approaches are reached for large data sets.

6.3 The proposed methodology based on Adaptive Constrained Clustering Algorithm

In this section, an Adaptive Constrained Clustering Algorithm (ACCA) is proposed to perform real-time fault detection in accordance with the problem formulated in Section 6.2. The basic idea is to cluster the incoming data points into two clusters over time, representing the nominal and non-nominal work conditions, respectively. The cluster of non-nominal conditions groups all the possible faults that may affect the industrial system dynamics.

Algorithm 4: Adaptive Constrained Clustering Algorithm

Inputs: Data stream $\mathscr{D}_j$ $(\forall j = 1,2,\ldots)$
Outputs: Data clusters $\{\mathscr{M}_j^H, \mathscr{M}_j^F\}$ $(\forall j = 1,2,\ldots)$

Initialize micro-clusters $\{\mathscr{N}_0^1,\ldots,\mathscr{N}_0^{I_0}\} = \{\emptyset\}$
Initialize macro-clusters $\mathscr{M}_0^H = \mathscr{M}_0^F = \emptyset$
for $j = 1,2,\ldots$ **do**
  $\left[\{\mathscr{N}_j^1,\ldots,\mathscr{N}_j^{I_j}\}, \{\mathrm{CF}_{\mathscr{N}_j^1},\ldots,\mathrm{CF}_{\mathscr{N}_j^{I_j}}\}\right]$ =
  Micro-clustering$\left(\mathscr{D}_j, \{\mathscr{N}_{j-1}^1,\ldots,\mathscr{N}_{j-1}^{I_{j-1}}\}\right)$
  $\{\mathscr{M}_j^H, \mathscr{M}_j^F\}$ =
  Macro-clustering$\left(\{\mathrm{CF}_{\mathscr{N}_j^1},\ldots,\mathrm{CF}_{\mathscr{N}_j^{I_j}}\}, \{\mathscr{M}_{j-1}^H, \mathscr{M}_{j-1}^F\}\right)$

To this aim, the proposed algorithm relies on a two-stage procedure including the *micro-clustering* and *macro-clustering* steps as shown in Fig. 6.2. The former is responsible for grouping the batches of data into *micro-clusters* while DS continuously arrives from the data acquisition system. Then, after processing each batch of data in the micro-clustering step, the micro-clusters are condensed into vectors of *cluster features*. Finally, leveraging on additional knowledge on the nominal working conditions set (i.e., clustering constraints on some samples), the macro-clustering aims at grouping the micro-clusters features into *macro-clusters*, so that each data point is clustered in accordance with the overall clusters. The procedure is repeatedly executed until the DS process is active.

The overall procedure is formally described in Algorithm 4, whilst the micro-clustering and macro-clustering steps – respectively defined in Algorithm 5 and 6 – are described in detail in the sequel.

6.3.1 Micro-clustering

The goal of the primary step is to identify and discover behavioral groups (or clusters) in the DS. The micro-clustering procedure shown in Algorithm 5 is based on leader clustering, which is a partition-based DS clustering technique [64]. Each batch of data is optimally clustered into micro-clusters using a distance criterion: whenever a new data point $\vec{X}$ arrives from the streams, the cluster i^* having the closest distance d^* is found. If such a distance is below the threshold $d_{threshold}$, the

Algorithm 5: Micro-clustering

Inputs: Data batch $\mathscr{D}_j$, previous micro-clustering $\{\mathscr{N}_{j-1}^1,\ldots,\mathscr{N}_{j-1}^{I_{j-1}}\}$

Outputs: Updated micro-clustering $\{\mathscr{N}_j^1,\ldots,\mathscr{N}_j^{I_j}\}$, micro-cluster features $\{\mathrm{CF}_{\mathscr{N}_j^1},\ldots,\mathrm{CF}_{\mathscr{N}_j^{I_j}}\}$

Parameters: Distance threshold d_{th}

$I_j \leftarrow I_{j-1}$

$\{\mathscr{N}_j^1,\ldots,\mathscr{N}_j^{I_j}\} \leftarrow \{\mathscr{N}_{j-1}^1,\ldots,\mathscr{N}_{j-1}^{I_{j-1}}\}$

for $\vec{X} \in \mathscr{D}_j$ **do**

  $i^* = \min_{i\in 1,\ldots,I_j}\left(\mathrm{dist}(\vec{X},\mathscr{N}_j^i)\right)$

  $d^* = \mathrm{dist}(\vec{X},\mathscr{N}_j^{i^*})$

  if $d^* < d_{th}$ **then**

    $\mathscr{N}_j^{i^*} \leftarrow \mathscr{N}_j^{i^*} \cup \{\vec{X}\}$

  else

    $I_j \leftarrow I_j + 1$

    $\mathscr{N}_j^{I_j} \leftarrow \{\vec{X}\}$

for $\mathscr{N} \in \{\mathscr{N}_j^1,\ldots,\mathscr{N}_j^{I_j}\}$ **do**

  compute $\mathrm{CF}_{\mathscr{N}} = (N_{\mathscr{N}}, \overrightarrow{\mathrm{LS}}_{\mathscr{N}}, \mathrm{SS}_{\mathscr{N}})$ through (6.3)-(6.5)

data point $\vec{X}$ is assigned to cluster i^*, otherwise a new cluster (leader) is created, being composed of the single data point $\vec{X}$. Note that no prior information on the number of micro-clusters is needed; however, the performance of the algorithm depends on the correctness of the threshold guess $d_{threshold}$. Indeed, $d_{threshold}$ is used to control the granularity of the micro-clustering and the trade-off between micro-cluster size and number of micro-clusters. A common approach is to set statically such a threshold to low values that capture the local structure of data while ensuring the resulting clusters not to be too small.

As an alternative, the Exponentially Weighted Moving Average (EWMA) [88, 89] method can be used to dynamically adjust and fine-tune the $d_{threshold}$ value in real-time, allowing for adaptive and responsive adjustments based on dynamic characteristics and the eventual occurrence of drift in data.

Once the data points in the batch are clustered into micro-clusters, the following features are computed for each micro-cluster. Given a cluster $\mathscr{N}$ composed of data

Algorithm 6: Macro-clustering

Inputs: Micro-cluster features $\{\mathrm{CF}_{\mathscr{N}_j^1}, \ldots, \mathrm{CF}_{\mathscr{N}_j^{I_j}}\}$, previous micro-clustering $\{\mathscr{M}_{j-1}^H, \mathscr{M}_{j-1}^F\}$

Outputs: Updated macro-clustering $\{\mathscr{M}_j^H, \mathscr{M}_j^F\}$

Parameters: Must-link constraints for nominal and non-nominal conditions $\mathscr{H}$ and $\mathscr{F}$

for $CF \in \{CF_{\mathscr{N}_j^1}, \ldots, CF_{\mathscr{N}_j^{I_j}}\}$ **do**
- $d^H = \mathrm{dist}(\mathrm{CF}, \mathscr{H})$
- $d^F = \mathrm{dist}(\mathrm{CF}, \mathscr{F})$
- **if** $d^H < d^F$ **then**
  - $\mathscr{M}_j^H \leftarrow \mathscr{M}_{j-1}^H \cup \{\mathrm{CF}\}$
- **else**
  - $\mathscr{M}_j^F \leftarrow \mathscr{M}_{j-1}^F \cup \{\mathrm{CF}\}$

points $\{\vec{X}_i\}$, the vector of corresponding cluster features is defined as the following triple:

$$\mathrm{CF}_{\mathscr{N}} = (N_{\mathscr{N}}, \overrightarrow{\mathrm{LS}}_{\mathscr{N}}, \mathrm{SS}_{\mathscr{N}}) \tag{6.2}$$

where $N_{\mathscr{N}}$ is the number of points in $\mathscr{N}$, $\overrightarrow{\mathrm{LS}}_{\mathscr{N}}$ is the linear sum of the data points, and $\mathrm{SS}_{\mathscr{N}}$ is the square sum of data points:

$$N_{\mathscr{N}} = |\mathscr{N}| \tag{6.3}$$

$$\overrightarrow{\mathrm{LS}}_{\mathscr{N}} = \sum_{i=1}^{N_{\mathscr{N}}} \vec{X}_i \tag{6.4}$$

$$\mathrm{SS}_{\mathscr{N}} = \sum_{i=1}^{N_{\mathscr{N}}} \vec{X}_i^2 = \sum_{i=1}^{N_{\mathscr{N}}} \left\langle \vec{X}_i, \vec{X}_i \right\rangle. \tag{6.5}$$

6.3.2 Constrained macro-clustering

The goal of the secondary step is to group the micro-clusters obtained by the previous step into macro-clusters based on a given set of constraints, which are additional information about the data set. As soon as the micro-clustering creates the micro-clusters features over batch data, the macro-clustering step updates the final clusters

using the constrained sets. In particular, as formally described in Algorithm 3, the observed I micro-clusters are grouped into $K = 2$ predefined groups (i.e., the macro-clusters of nominal and non-nominal conditions, which are denoted at iteration j by $\mathscr{M}_j^H$ and $\mathscr{M}_j^F$, respectively), where an observed micro-cluster is represented by its corresponding cluster feature vector [64]. Indeed, as usually done in constrained clustering [90], the Euclidean distance between each micro-cluster centroid and the must-link constraints sets is calculated in order to update the overall macro-clusters after each batch processing. In particular, knowing the corresponding cluster features, the centroid $\vec{X}^0_{\mathscr{N}}$ is computed in reference to cluster $\mathscr{N}$ as:

$$\vec{X}^0_{\mathscr{N}} = \frac{\sum_{i=1}^{N_{\mathscr{N}}} \vec{X}_i}{N_{\mathscr{N}}} = \frac{\overrightarrow{LS}_{\mathscr{N}}}{N_{\mathscr{N}}}. \tag{6.6}$$

The constraints sets can be represented by *must-link* (i.e., constraint specifying that two data points must be assigned to the same cluster) and/or *cannot-link* (i.e., constraint specifying that two data points cannot belong to the same cluster) constraints; however, without loss of generality, in Algorithm 3 I refer to the must-link set only. Hence, the distance of the must-link samples to micro-clusters is calculated, relying on the information on some samples related to the healthy and faulty behaviors of the considered industrial component (i.e., samples belonging to the macro-cluster of nominal and non-nominal conditions). As a result, the macro-clustering component finds the nominal micro-clusters and merges them in a final cluster of nominal conditions, whilst the remaining micro-clusters are combined in the final cluster of non-nominal conditions.

6.4 Case study: Complex Pneumatic System

For the sake of validating the proposed approach, the Adaptive Constrained Clustering algorithm is implemented in Python and used to perform the real-time fault detection of a pneumatic system deployed in a real industrial process. In particular, the experiments refer to a robotic work-cell produced by an automotive company leader of the Italian and European market in section 1.5.2. The whole cell – that is employed to analyze the quality and detect any possible defects of an automotive work-piece performing a 90-second work-cycle – consists of an anthropomorphic manipulator and three grippers. Among them, the “rotary gripper" (positioned on the

base of the framework) represents the most critical component and, consequently, is subject to frequent faults. Hence, it constitutes the case study for the proposed real-time fault detection procedure.

6.4.1 Experimental setup

A portion equal to 1% of the dataset is used to define the constrained set of sample working conditions. Specifically, the faulty must-link set $\mathscr{F}$ is formed by picking one sample from each of the five non-nominal categories in Table 1.4, while the healthy must-link set $\mathscr{H}$ is composed of two samples of nominal operating condition. Furthermore, for the sake of evaluating the effectiveness of the proposed approach, the labels are removed from the entire data set; subsequently, the Adaptive Constrained Clustering algorithm is applied to the unlabeled data; finally, the obtained macro-clusters are compared with the true classes (i.e., the *ground truth*), while different clustering methods can be compared through evaluation metrics. The experiment is repeated 100 times by changing both the batch size and the value of the algorithm parameters to show the performance sensitivity.

6.4.2 Performance evaluation

The performance of the micro-clustering procedure is evaluated using the *sum of squared errors* (SSQ) and the *Silhouette Score* (SS), which are internal methods (i.e., they evaluate the clustering performance without any reference). The former measures the sum of squared distances of data points from their cluster centroids:

$$\mathrm{SSQ} = \sum_{k=1}^{I} \sum_{\vec{X}_i \in \mathscr{N}_k} \left\| \vec{X}_i - \vec{X}^0_{\mathscr{N}_k} \right\|^2 . \tag{6.7}$$

Conversely, the SS of a data point aims at assessing how well a data point fits into the corresponding micro-cluster compared to others:

$$\mathrm{SS}(\vec{X}_i) = \frac{b(\vec{X}_i) - a(\vec{X}_i)}{\max\{a(\vec{X}_i), b(\vec{X}_i)\}} \tag{6.8}$$

where $a(\vec{X}_i)$ is the average dissimilarity of point $\vec{X}_i$ with all other points in the same cluster, and $b(\vec{X}_i)$ is the lowest average dissimilarity of point $\vec{X}_i$ to any other cluster to which it does not belong. The SS for the overall clustering is the mean of the SS for all data points, as follows:

$$\mathrm{SS} = \sum_{\vec{X}_i \in \{\mathscr{N}_1 \cup \ldots \cup \mathscr{N}_I\}} \frac{\mathrm{SS}(\vec{X}_i)}{N}. \tag{6.9}$$

As for the evaluation of the macro-clustering, the *Purity index* is considered, which is one the most commonly used external methods (i.e., they assess the accuracy of clustering by comparing the results to known ground truth). It is computed as follows:

$$\text{Purity } = \frac{1}{N} \sum_{h=1}^{K} \max_{k \in \{H,F\}} \left| \Omega_h \cap \mathscr{M}^k \right| \tag{6.10}$$

where Ω_h (for $h = 1,2$) denotes the sets of nominal and non-nominal data points with ground truth labels. Note that perfect clustering has a purity index equal to 1, while poor clustering has purity values close to 0.

6.4.3 Results analysis and discussion

Figure 6.3 shows the composition of all micro-clusters and two macro-clusters over time obtained in the case of batches with a fixed size equal to $B = 50$ and setting the threshold distance to $d_{threshold} = 1$ (i.e., a maximum size of 100 data points for each microcluster in real-time processing).

It is worthwhile noting two findings: on the one hand, the number and size of micro-clusters (represented in Fig. 6.3 by coloring the corresponding data points) increases over iterations; on the other hand, at each iteration, all data points are grouped into two macro-clusters (represented in Fig. 6.3 by coloring the corresponding micro-clusters), thus allowing to properly distinguish between the nominal and non-nominal working conditions in real-time.

Finally, since the most significant algorithm parameter is the distance threshold $d_{threshold}$, different experiments are analyzed and compared by varying $d_{threshold}$ within the range $[0,1,2,3,4,5,6,7,8]$.

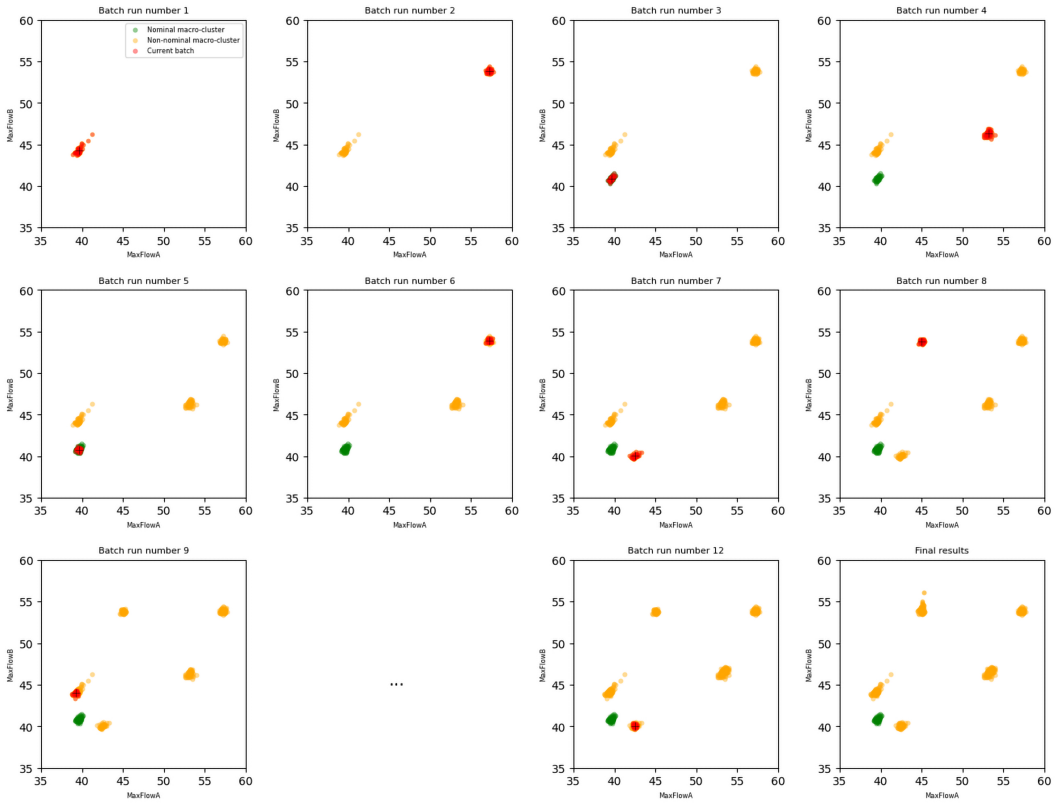
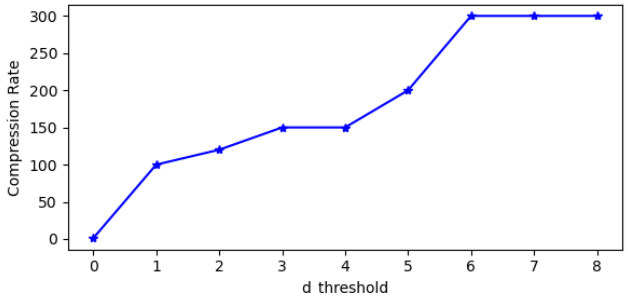


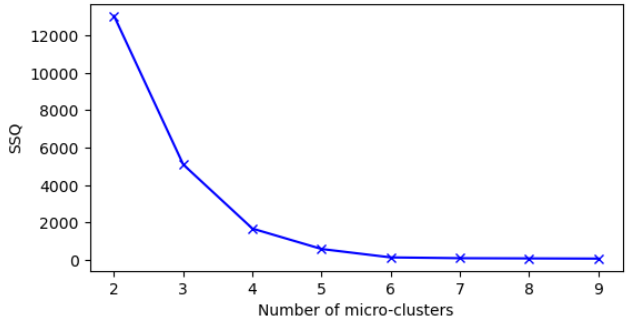
Fig. 6.3 Results of the Adaptive Constrained Clustering algorithm over batches in the case of batch size $B = 50$. Data points in the current batch are represented by red color, while the green and orange colors denote the micro-clusters belonging to the nominal and non-nominal macro-clusters, respectively.

In particular, for each value of $d_{threshold}$ the number of micro-clusters is evaluated over the whole DS by means of the so-called *compression rate* (i.e., the number of samples divided by the number of the micro-clusters) and the *Elbow method* (i.e., the heuristics aimed at determining the optimal number of clusters) [64]. From the plots shown in Fig. 6.4 it can be seen that for $d_{threshold} = 1$ the compression rate is equal to 100 (Fig. 6.4.a) while the optimal number of micro-clusters is equal to 6 (Fig. 6.4.b), implying that each micro-cluster optimally comprises 100 data points. This result is also confirmed by the *Silhouette analysis* that usefully aims at investigating the separation distance between the resulting clusters: Fig. 6.5 reports the average Silhouette score obtained varying the numbers of micro-clusters I, showing that the best values are achieved in the case of 6 clusters.

In addition, as for the macro-clustering, the purity metric is evaluated: the two obtained macro-clusters include respectively 100 and 500 data points, precisely



(a)



(b)

Fig. 6.4 Compression rate as a function of radius threshold (a) and sum of square error over the number of clusters (b) for the micro-clustering of the Adaptive Constrained Clustering approach.

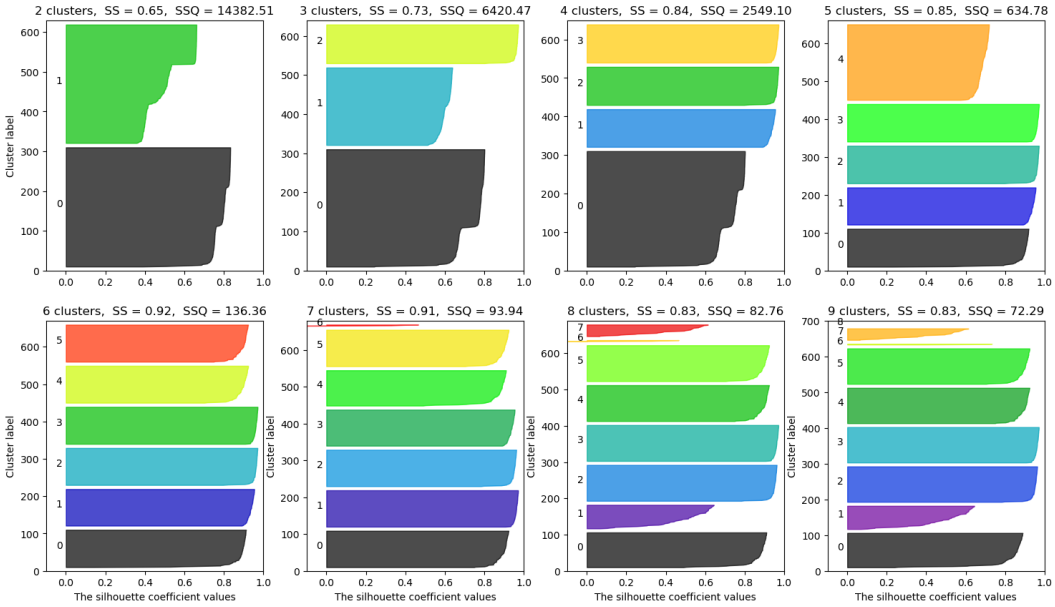


Fig. 6.5 Results of the silhouette analysis on micro-clustering varying the number of micro-clusters in the range [2,9].

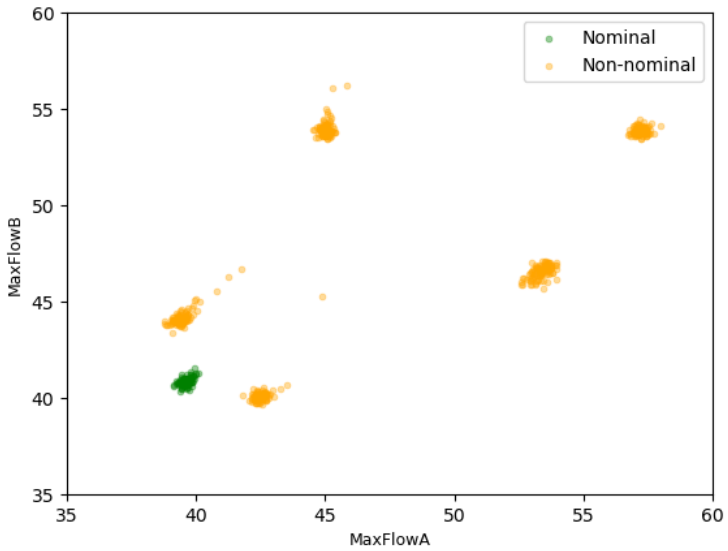


Fig. 6.6 Results obtained by the Constrained K-means algorithm with two final clusters.

corresponding to the data points representing the nominal and non-nominal classes (ground truth), thus, reaching a purity index equal to 1. This shows the accuracy and reliability of the method in properly classifying the data in accordance with the nominal/non-nominal working conditions.

6.4.4 Comparison with related clustering approaches

In order to show the effectiveness of the proposed Adaptive Constrained Clustering, it is compared with both a static (namely, the *Constrained K-means*) and a stream clustering (namely, the Stream K-means) approach.

The Constrained K-means performs a profitable modification of the traditional K-means algorithm: leveraging on background knowledge about the data set, it uses pairwise must-link and cannot-link constraints [8]. Imposing the number of final clusters equal to 2 and using the must-link constraints $\mathscr{F}$ and $\mathscr{H}$, the results are shown in Fig. 6.6. Although it is apparent that the obtained results in terms of final clusters are equal to those achieved by the proposed approach, it is worthwhile noting that my Adaptive Constrained Clustering algorithm works in real-time. Therefore, even though Constrained K-means correctly detect the effective working conditions, reaching a purity value equal to 1, it is an offline and therefore slower approach for data mining on the whole data set.

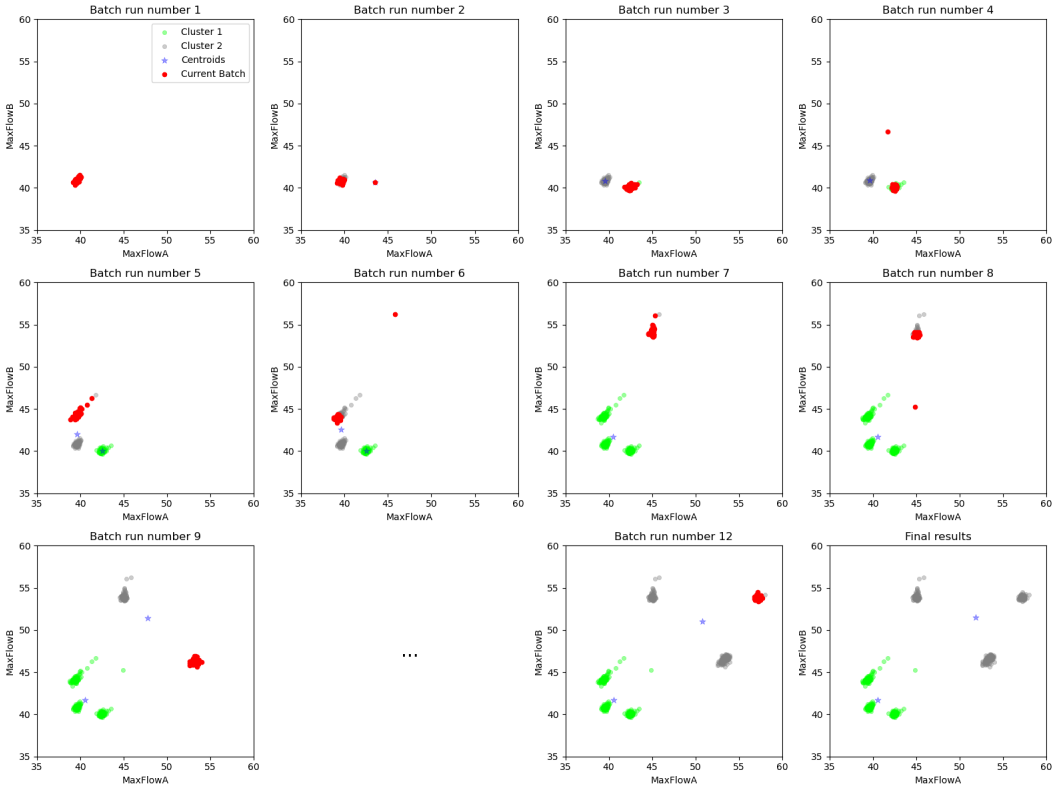


Fig. 6.7 Results of the Stream K-means algorithm over batches in the case of batch size $B = 50$. Data points in the current batch are represented by red color, while the light-green and grey colors denote the two clusters, whose updated centroids at each iteration are indicated by the blue stars.

Conversely, the well-known Stream K-means, also called online K-means, is a variant of the classical K-means algorithm designed to handle large data sets [64, 8]. In this approach, the data is processed sequentially in batches, rather than all at once. Each batch is used to update the current set of cluster centroids, which are then refined iteratively. By processing data in a streaming fashion, it is possible to handle large data sets efficiently and in real-time, without requiring the entire data set to be loaded into memory.

However, in this approach, the number of clusters must be predefined. Moreover, it is worthwhile noting that Stream K-means requires a longer computational time than my approach due to the iterative process of updating the current set of cluster centroids within each batch. Conversely, the proposed Adaptive Constrained Clustering algorithm always maintains a set of $\mathrm{CF}_{\mathscr{N}}$ that summarizes the features of each

corresponding micro-cluster, which makes it faster than the Stream K-means, despite requiring more storage memory.

Figure 6.7 shows the results of the Stream K-means algorithm applied to the case study, considering batches of 50 points each and two clusters ($K = 2$). It is evident that, during the first iterations, the clusters formed by the first batches are not well-separated; as new batches are added and the algorithm is applied to larger and larger portions of the data, the clusters become less distinct and begin to overlap in some iterations.

In addition, the clusters formed at each iteration are dependent on the order in which the batches of data points are processed. On the contrary, in the Adaptive Constrained Clustering method, the order in which the data points arrive does not affect the clustering results at each iteration or at the end of the procedure.

Indeed, the Stream K-means algorithm is not very effective at clustering this data set, as the clusters are not well-separated and do not remain stable over time. In terms of purity, the Stream K-means only scores 0.66. Such a result demonstrates that the proposed approach outperforms the other reference streaming methods in terms of accuracy.

6.5 Conclusion of Adaptive Constrained Clustering Algorithm

Thanks to the pervasive deployment of sensors in Industry 4.0, data-driven methods have recently played an important role in the fault diagnosis and prognosis of industrial systems. In this chapter, a novel Adaptive Constrained Clustering algorithm is defined to support real-time fault detection of an industrial machine, by clustering the incoming monitoring data into two clusters over time, representing the nominal and non-nominal work conditions, respectively. To this aim, the proposed algorithm relies on a two-stage procedure: *micro-clustering* and *constrained macro-clustering*. The former stage is responsible for grouping the batches of work-cycle data into *micro-clusters*, while the data stream continuously arrives from the data acquisition system. Then, after condensing the micro-clusters into vectors of cluster features, and leveraging on additional knowledge on the nominal and non-nominal working conditions (i.e., constraints on some samples), the second stage aims at offline

grouping the micro-clusters features into *macro-clusters*. Experimental results on a real-world industrial case study show that the proposed real-time framework achieves the same results of offline baseline methods (e.g., Constrained K-means) with a higher responsiveness and processing speed; in comparison to stream baseline methods (e.g., Stream K-means), the proposed approach obtains more accurate and easily interpretable results.

Chapter 7

Conclusion and Future Work

7.1 Conclusion of the Thesis

In this comprehensive study, I have applied various machine learning (ML) techniques to address fault detection and diagnosis (FDD) challenges in complex industrial systems. Firstly, I demonstrated the effectiveness of the Early Time-Series Classification (ETSC) model in detecting potential failures in a hydraulic system (HS). ETSC not only achieved high accuracy but also significantly reduced prediction time, enabling real-time monitoring and predictive maintenance. Secondly, a Multi-Class Multi-Output (MCMO) classification approach was proposed to forecast the states of multiple HS components simultaneously, outperforming traditional methods and providing consistent predictions. This approach also allowed for the diagnosis of overall HS states, emphasizing its practical utility. Furthermore, I explored semi-supervised learning (SSL) for fault detection in pneumatic and hydraulic systems, showcasing the capability of SSL to achieve high accuracy with limited labeled data. Lastly, an unsupervised ML method for real-time fault detection was proposed, which involved micro-clustering and macro-clustering stages. This approach demonstrated superior performance compared to baseline methods, including static-constrained K-means and online Stream K-means. Overall, my research contributes to the advancement of FDD in industrial systems through the application of diverse ML methodologies.

7.2 Future Work

Future research will explore diverse avenues to enhance my existing methodologies. I plan to incorporate additional base classifiers into the ETSC model to tailor its performance to various types of sequential data. Furthermore, I intend to address sensor data feature selection challenges by replacing traditional classifiers with deep learning models. Additionally, I aim to expand my horizons in failure analysis by considering probabilistic modeling approaches that account for dependencies, including dynamic fault trees and stochastic Petri nets. Another perspective is to incorporate the learning of event combinations involving dependencies into the ML phase of my approach. Moreover, I intend to apply the multi-class multi-output (MCMO) model to datasets with lower dynamism than the hydraulic system dataset. This will allow us to conduct a fundamental fault tree analysis and determine the system's reliability. In terms of data environments, my future efforts will focus on dynamic settings with continuously generated data, particularly stream data. I will also explore the integration of semi-supervised learning models with maintenance management to optimize maintenance intervals, thereby enhancing system reliability and minimizing inspection costs. Lastly, I will explore the inclusion of a fault identification sub-stage to provide deeper insights into the root causes of faults, culminating in a comprehensive unsupervised fault diagnosis algorithm. As I delve deeper into stream data mining, addressing the challenge of concept drift within the dataset emerges as a pivotal avenue for future research in this field.

References

[1] Keke Huang, Shujie Wu, Fanbiao Li, Chunhua Yang, and Weihua Gui. Fault diagnosis of hydraulic systems based on deep learning model with multirate data samples. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13, 2021.

[2] Atin Roy and Subrata Chakraborty. Support vector machine in structural reliability analysis: A review. *Reliability Engineering & System Safety*, page 109126, 2023.

[3] Malti Bansal, Apoorva Goyal, and Apoorva Choudhary. A comparative analysis of k-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning. *Decision Analytics Journal*, 3:100071, 2022.

[4] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[5] Kun-Cheng Ke and Ming-Shyan Huang. Quality prediction for injection molding by using a multilayer perceptron neural network. *Polymers*, 12(8):1812, 2020.

[6] Mohsen Hajighorbani, SM Reza Hashemi, B Minaei-Bidgoli, and Shabnam Safari. A review of some semi-supervised learning methods. In *IEEE-2016, first international conference on new research achievements in electrical and computer engineering*, pages 1–10, 2016.

[7] Yanwen Chong, Yun Ding, Qing Yan, and Shaoming Pan. Graph-based semi-supervised learning: A review. *Neurocomputing*, 408:216–230, 2020.

[8] Abiodun M Ikotun, Absalom E Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 2022.

[9] GeeksforGeeks. Ml - principal component analysis. https://www.geeksforgeeks.org/principal-component-analysis-pca/, March 2023. Accessed on October 2023.

[10] GeeksforGeeks. Ml - linear discriminant analysis. https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/, March 2023. Accessed on October 2023.

[11] Alberto Pliego Marugán. Applications of reinforcement learning for maintenance of engineering systems: A review. *Advances in Engineering Software*, 183:103487, 2023.

[12] Asma Dachraoui, Alexis Bondu, and Antoine Cornuéjols. Early classification of time series as a non myopic sequential decision making problem. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pages 433–447. Springer, 2015.

[13] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[14] Bahman Askari, Augusto Bozza, Graziana Cavone, Raffaele Carli, and Mariagrazia Dotoli. An adaptive constrained clustering approach for real-time fault detection of industrial systems. In *European Journal of Control (EJC)*, 2023. In press.

[15] Zeki Murat Çınar, Abubakar Abdussalam Nuhu, Qasim Zeeshan, Orhan Korhan, Mohammed Asmael, and Babak Safaei. Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0. *Sustainability*, 12(19):8211, 2020.

[16] Anam Abid, Muhammad Tahir Khan, and Javaid Iqbal. A review on fault detection and diagnosis techniques: basics and beyond. *Artificial Intelligence Review*, 54:3639–3664, 2021.

[17] Attila Ernő Frankó and Pál Varga. A survey on machine learning based smart maintenance and quality control solutions. *Infocommunications Journal*, 13(4):28–35, 2021.

[18] Cheng Ji and Wei Sun. A review on data-driven process monitoring methods: Characterization and mining of industrial data. *Processes*, 10(2):335, 2022.

[19] Miroslav Kubat, Ivan Bratko, and Ryszard S Michalski. A review of machine learning methods. *Machine Learning and Data Mining: Methods and Applications*, pages 3–69, 1998.

[20] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[21] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. A review of supervised machine learning algorithms. In *2016 3rd international conference on computing for sustainable global development (INDIACom)*, pages 1310–1315. Ieee, 2016.

[22] Tammy Jiang, Jaimie L Gradus, and Anthony J Rosellini. Supervised machine learning: a brief primer. *Behavior Therapy*, 51(5):675–687, 2020.

[23] Shan Suthaharan. Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, 36:1–12, 2016.

[24] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13(4):18–28, 1998.

[25] B. Scholkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Muller, G. Ratsch, and A. J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5):1000–1017, 1999.

[26] Vinícius G Costa and Carlos E Pedreira. Recent advances in decision trees: An updated survey. *Artificial Intelligence Review*, 56(5):4765–4800, 2023.

[27] J Ross Quinlan. An empirical comparison of genetic and decision-tree classifiers. In *Machine Learning Proceedings 1988*, pages 135–141. Elsevier, 1988.

[28] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[29] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2):13–19, 2014.

[30] Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, and Raihani Mohamed. A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru. *arXiv preprint arXiv:2305.17473*, 2023.

[31] Francesca Calabrese, Alberto Regattieri, Lucia Botti, Cristina Mora, and Francesco Gabriele Galizia. Unsupervised fault detection and prediction of remaining useful life for online prognostic health management of mechanical systems. *Applied Sciences*, 10(12):4120, 2020.

[32] Nikolai Helwig, Eliseo Pignanelli, and Andreas Schütze. Condition monitoring of a complex hydraulic system using multivariate statistics. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pages 210–215, 2015.

[33] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mechanical Systems and Signal Processing*, 104:799–834, 2018.

[34] Sudarshan S. Chawathe. Condition monitoring of hydraulic systems by classifying sensor data streams. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0898–0904, 2019.

[35] Z. Xu, H. Yu, J. Tao, and C. Liu. Compound fault diagnosis in hydraulic system with multi-output svm. In *CSAA/IET International Conference on Aircraft Utility Systems (AUS 2020)*, volume 2020, pages 84–89, 2020.

[36] Yafei Lei, Wanlu Jiang, Anqi Jiang, Yong Zhu, Hongjie Niu, and Sheng Zhang. Fault diagnosis method for hydraulic directional valves integrating pca and xgboost. *Processes*, 7(9), 2019.

[37] Xiaohang Zhao, Ke Zhang, and Yi Chai. A multivariate time series classification based multiple fault diagnosis method for hydraulic systems. In *2019 Chinese Control Conference (CCC)*, pages 6819–6824, 2019.

[38] Zhijie Peng, Ke Zhang, and Yi Chai. Multiple fault diagnosis for hydraulic systems using nearest-centroid-with-dba and random-forest-based-time-series-classification. In *2020 39th Chinese Control Conference (CCC)*, pages 29–86, 2020.

[39] Wei Wang, Yan Li, and Yuling Song. Fault diagnosis method of hydraulic system based on multi-source information fusion and fractal dimension. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 2021.

[40] Kyutae Kim and Jongpil Jeong. Real-time monitoring for hydraulic states based on convolutional bidirectional lstm with attention mechanism. *Sensors (Basel, Switzerland)*, 20, 2020.

[41] Majdi Mansouri, Mohamed Trabelsi, Hazem Nounou, and Mohamed Nounou. Deep learning-based fault diagnosis of photovoltaic systems: A comprehensive review and enhancement prospects. *IEEE Access*, 9:126286–126306, 2021.

[42] Zhaoyi Xu and Joseph Homer Saleh. Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliability Engineering & System Safety*, 211:107530, 2021.

[43] Bahman Askari, Raffaele Carli, Graziana Cavone, and Mariagrazia Dotoli. Data-driven fault diagnosis in a complex hydraulic system based on early classification. *IFAC-PapersOnLine*, 55(40):187–192, 2022.

[44] Qadeer Ahmed, Syed Asif Raza, and Dahham M Al-Anazi. Reliability-based fault analysis models with industrial applications: A systematic literature review. *Quality and Reliability Engineering International*, 37(4):1307–1333, 2021.

[45] Felipe Augusto Sviaghin Ferri, Leonardo Ramos Rodrigues, João Paulo Pordeus Gomes, Ivo Paixão de Medeiros, Roberto Kawakami Harrop Galvão, and Cairo Lúcio Nascimento. Combining phm information and system architecture to support aircraft maintenance planning. pages 60–65, 2013.

[46] Deniz Tuncay and Nuray Demirel. Reliability analysis of a dragline using fault tree analysis. *Scientific Mining Journal*, 56(2):55–64, 2017.

[47] Rajkumar B Patil, Digvijay A Mhamane, Pruthwiraj B Kothavale, and Basavraj Kothavale. Fault tree analysis: a case study from machine tool industry. In *Proceedings of TRIBOINDIA-2018 An International Conference on Tribology*, 2018.

[48] Shizheng Li, Zhaojun Yang, Hailong Tian, Chuanhai Chen, Yongfu Zhu, Fuqin Deng, and Song Lu. Failure analysis for hydraulic system of heavy-duty machine tool with incomplete failure data. volume 11, page 1249. MDPI, 2021.

[49] Kerelous Waghen and Mohamed-Salah Ouali. Multi-level interpretable logic tree analysis: A data-driven approach for hierarchical causality analysis. *Expert Systems with Applications*, 178:115035, 2021.

[50] Yong-bum Lee, Gi-chun Lee, Jong-dae Yang, Jong-won Park, and Dong-cheon Baek. Failure analysis of a hydraulic power system in the wind turbine. *Engineering Failure Analysis*, 107:104218, 2020.

[51] Yi-Min Mo, Yi Gong, and Zhen-Guo Yang. Failure analysis on the o-ring of radial thrust bearing room of main pump in a nuclear power plant. *Engineering Failure Analysis*, 115:104673, 2020.

[52] Tong-Wei Ni and Zhen-Guo Yang. Failure analysis on unexpected leakage of electro-hydraulic servo valve in digital electric hydraulic control system of 300 mw thermal power plant. *Engineering Failure Analysis*, 119:104992, 2021.

[53] Andre Lemos, Walmir Caminhas, and Fernando Gomide. Adaptive fault detection and diagnosis using an evolving fuzzy classifier. *Information Sciences*, 220:64–85, 2013.

[54] Isaac Monroy, Raul Benitez, Gerard Escudero, and Moisès Graells. A semi-supervised approach to fault diagnosis for chemical processes. *Computers & Chemical Engineering*, 34(5):631–642, 2010. Selected Paper of Symposium ESCAPE 19, June 14-17, 2009, Krakow, Poland.

[55] Shaozhi Chen, Rui Yang, and Maiying Zhong. Graph-based semi-supervised random forest for rotating machinery gearbox fault diagnosis. *Control Engineering Practice*, 117:104952, 2021.

[56] Ke Yan, Chaowen Zhong, Zhiwei Ji, and Jing Huang. Semi-supervised learning for early detection and diagnosis of various air handling unit faults. *Energy and Buildings*, 181:75–83, 2018.

[57] Ikenna A Okaro, Sarini Jayasinghe, Chris Sutcliffe, Kate Black, Paolo Paoletti, and Peter L Green. Automatic fault detection for laser powder-bed fusion using semi-supervised machine learning. *Additive Manufacturing*, 27:42–53, 2019.

[58] Taeyoung Ko and Heeyoung Kim. Fault classification in high-dimensional complex processes using semi-supervised deep convolutional generative models. *IEEE Transactions on Industrial Informatics*, 16(4):2868–2877, 2020.

[59] Kun Yu, Tian Ran Lin, Hui Ma, Xiang Li, and Xu Li. A multi-stage semi-supervised learning approach for intelligent fault diagnosis of rolling bearing using data augmentation and metric learning. *Mechanical Systems and Signal Processing*, 146:107043, 2021.

[60] Shuyuan Zhang and Tong Qiu. Semi-supervised lstm ladder autoencoder for chemical process fault diagnosis and localization. *Chemical Engineering Science*, 251:117467, 2022.

[61] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[62] Xiaogang Wang, Haichao Feng, and Yunpeng Fan. Fault detection and classification for complex processes using semi-supervised learning algorithm. *Chemometrics and intelligent laboratory systems*, 149:24–32, 2015.

[63] Mauricio Zuluaga Rodriguez, Cesar Henrique Comin, Daladier Casanova, Odemir Martinez Bruno, and Diego Raphael Amancio. Clustering algorithms: A comparative approach. *PLOS ONE*, 14(1):e0210236, 2019.

[64] Stratos Mansalis, Eirini Ntoutsi, Nikos Pelekis, and Yannis Theodoridis. An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 11(4):167–187, 2018.

[65] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[66] Salima Omar, Asri Ngadi, and Hamid H Jebur. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79(2), 2013.

[67] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[68] J. Montoya-Bedoya, R. Graves, S. Joshi, and M. Satra. Feature design, selection, and model optimization for fault classification in a hydraulic rock drill. *Annual Conference of the PHM Society*, 14(1), Oct 2022.

[69] Gian Paolo Incremona and Antonella Ferrara. Fault diagnosis for robot manipulators via vision servoing based suboptimal second order sliding mode. In *2019 18th European Control Conference (ECC)*, pages 3090–3095. IEEE, 2019.

[70] Gustavo Dalmarco, Filipa R Ramalho, Ana C Barros, and Antonio L Soares. Providing industry 4.0 technologies: The case of a production technology cluster. *The journal of high technology management research*, 30(2):100355, 2019.

[71] Angelo Cenedese, Michele Luvisotto, and Giulia Michieletto. Distributed clustering strategies in industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics*, 13(1):228–237, 2017.

[72] Cristiano Hora Fontes and Hector Budman. A hybrid clustering approach for multivariate time series–a case study applied to failure analysis in a gas turbine. *ISA transactions*, 71:513–529, 2017.

[73] M Amine Atoui and Achraf Cohen. Fault diagnosis using pca-bayesian network classifier with unknown faults. In *2020 European Control Conference (ECC)*, pages 2039–2044. IEEE, 2020.

[74] Chenhao Wu, Jiguang Yue, Li Wang, and Feng Lyu. Fault diagnosis of recessive weakness in superbuck converter based on kpca-ipnn. In *2020 European Control Conference (ECC)*, pages 2045–2050. IEEE, 2020.

[75] Eckart Uhlmann, Rodrigo Pastl Pontes, Claudio Geisert, and Eckhard Hohwieler. Cluster identification of sensor data for predictive maintenance in a selective laser melting machine tool. *Procedia Manufacturing*, 24:60–65, 2018. 4th International Conference on System-Integrated Intelligence: Intelligent, Flexible and Connected Systems in Products and Production.

[76] Javier Diaz-Rozo, Concha Bielza, and Pedro Larrañaga. Clustering of data streams with dynamic gaussian mixture models: An iot application in industrial processes. *IEEE Internet of Things Journal*, 5(5):3533–3547, 2018.

[77] Ahmad Alzghoul, Magnus Löfstrand, and Björn Backe. Data stream forecasting for system fault prediction. *Computers & industrial engineering*, 62(4):972–978, 2012.

[78] Alaettin Zubaroğlu and Volkan Atalay. Data stream clustering: a review. *Artificial Intelligence Review*, 54(2):1201–1236, 2021.

[79] Sergio Ramírez-Gallego, Bartosz Krawczyk, Salvador García, Michał Woźniak, and Francisco Herrera. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57, 2017.

[80] Jan Zenisek, Florian Holzinger, and Michael Affenzeller. Machine learning based concept drift detection for predictive maintenance. *Computers & Industrial Engineering*, 137:60–65, 2019.

[81] Nailong Zhang and Wujun Si. Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks. *Reliability Engineering & System Safety*, 203:107094, 2020.

[82] Charalampos P Andriotis and Konstantinos G Papakonstantinou. Managing engineering systems with large state and action spaces through deep reinforcement learning. *Reliability Engineering & System Safety*, 191:106483, 2019.

[83] Adam Thelen, Xiaoge Zhang, Olga Fink, Yan Lu, Sayan Ghosh, Byeng D Youn, Michael D Todd, Sankaran Mahadevan, Chao Hu, and Zhen Hu. A comprehensive review of digital twin—part 1: modeling and twinning enabling technologies. *Structural and Multidisciplinary Optimization*, 65(12):354, 2022.

[84] Amir Ebrahimifakhar, Adel Kabirikopaei, and David Yuill. Data-driven fault detection and diagnosis for packaged rooftop units using statistical machine learning classification methods. *Energy and Buildings*, 225:110318, 2020.

[85] Zhu Xiaojin and Ghahramani Zoubin. Learning from labeled and unlabeled data with label propagation. In *Tech. Rep., Technical Report CMU-CALD-02–107*. Carnegie Mellon University, 2002.

[86] Jonathan A Silva, Elaine R Faria, Rodrigo C Barros, Eduardo R Hruschka, André CPLF de Carvalho, and João Gama. Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):1–31, 2013.

[87] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. State-of-the-art on clustering data streams. *Big Data Analytics*, 1(1):1–27 , 2016.

[88] Damianos P Melidis, Myra Spiliopoulou, and Eirini Ntoutsi. Learning under feature drifts in textual streams. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 527–536, 2018.

[89] Supriya Agrahari and Anil Kumar Singh. Concept drift detection in data stream mining: A literature review. *Journal of King Saud University-Computer and Information Sciences*, 34(10):9523–9540, 2022.

[90] Bahman Askari and Sattar Hashemi. Text document clustering by using semi-supervised learning and outlier detection. *International Journal of Mechatronic, Electrical and Computer Technology (IJMEC)*, 7(23):3255–3262, January 2017.

Appendix A

Baseline Machine Learning Methods

A.1 Mathematical Basics of Classification Methods

In the context of classification, it is necessary to define $a_1x_1 + a_2x_2 - b = 0$ so that we can create a plane that can divide the data domain X_1X_2 into two subdomains D_1 and D_2, such that all the points in a subdomain can be labeled the same:

$$D_1 = \left\{(x_1, x_2) : [a_1, a_2]\,[x_1, x_2]' - b \leq 0\right\} \tag{A.1}$$

$$D_2 = \left\{(x_1, x_2) : [a_1, a_2]\,[x_1, x_2]' - b > 0\right\} \tag{A.2}$$

Therefore, our classifiers may be defined as:

$$[a_1, a_2]\,[x_1, x_2]' - b = -1 \text{ for the data points in domain } D_1$$

and

$$[a_1, a_2]\,[x_1, x_2]' - b = 1 \text{ for the data points in domain } D_2.$$

By applying the same parametrization process, we can establish high-dimensional classifiers for data points within a three-dimensional data domain, a four-dimensional data domain, and so forth. This classification model can be extended and represented in the following matrix form:

$$A\mathbf{x}' - b = -1; \mathbf{x} \in D_1 \tag{A.3}$$

$$A\mathbf{x}' - b = 1; \mathbf{x} \in D_2 \tag{A.4}$$

and the subdomains are:

$$D_1 = \left\{\mathbf{x} : A\mathbf{x}' - B \leq 0\right\} \tag{A.5}$$

$$D_2 = \left\{\mathbf{x} : A\mathbf{x}' - B > 0\right\} \tag{A.6}$$

These equations represent parameter models, but there exists a multitude of potential parameter values. The challenge is to choose values that meet the parametrization goals. After defining these values, the next step is to determine the optimal parameter values [23].

A.1.1 Logistic Regression

LR offers a classification model, which is discussed in this section using fewer variables and a straightforward two-class classification example [23]. In LR, the primary goal is to minimize the entropy E, and for a two-class problem, it can be defined as follows:

$$E = y\log(p) + (1-y)\log(1-p) \tag{A.7}$$

In this equation, we have the class label y, which can take values 0 or 1, and p represents the probability of detecting class label y. This can be simplified as follows:

$$E = \log(1-p) + y\log\frac{p}{(1-p)} \tag{A.8}$$

If we assume that the probability p follows a logistic function with a parameter a, we can express the probabilities p and $1-p$ as follows:

$$p = \frac{e^{ax}}{1+e^{ax}} \tag{A.9}$$

$$1-p = \frac{1}{1+e^{ax}} \tag{A.10}$$

where equation A.10 indicates the logistic (sigmoid) function. By replacing the mathematical expressions for both p and $1-p$ in Eq. (A.8), we can rewrite it with the following equation:

$$E = \log \frac{1}{1+e^{ax}} + y \log e^{ax} \tag{A.11}$$

$$E = -\log\left(1+e^{ax}\right) + yax \tag{A.12}$$

When we apply a partial derivative operator with respect to the parameter a, we obtain the following equation:

$$\frac{\partial E}{\partial a} = yx - \frac{e^{ax}}{1+e^{ax}}x = 0 \tag{A.13}$$

if we substitute Eq. (A.9) in the Eq. (A.13), we can get the following:

$$\frac{\partial E}{\partial a} = (y-p)x = 0 \tag{A.14}$$

The next step is to derive an expression for the second derivative from Eq. (A.13) that is important to update the parameter a. The results of the application of the second partial derivative are:

$$\frac{\partial^2 E}{\partial a^2} = -\frac{(1+e^{ax})\,xe^{ax}x - e^{ax}xe^{ax}x}{(1+e^{ax})^2} = 0 \tag{A.15}$$

$$\frac{\partial^2 E}{\partial a^2} = -\frac{xe^{ax}x}{(1+e^{ax})^2} = 0 \tag{A.16}$$

$$\frac{\partial^2 E}{\partial a^2} = -x\frac{e^{ax}}{(1+e^{ax})}\frac{1}{(1+e^{ax})}x = 0 \tag{A.17}$$

By substituting the mathematical expression in Eq. (A.34) for p, we can simplify the above equation and obtain the following:

$$\frac{\partial^2 E}{\partial a^2} = -xp(1-p)x = 0 \tag{A.18}$$

By representing $p(1-p)$ with w, we can rewrite this equation as follows:

$$\frac{\partial^2 E}{\partial a^2} = -xwx = 0 \tag{A.19}$$

We can now use the Newton–Raphson approach [**?**] to update the logistic regression parameter a:

$$a_{\text{curr}} = a_{\text{prev}} - \left(\frac{\partial^2 E}{\partial a^2}\right)^{-1} \frac{\partial E}{\partial a} \tag{A.20}$$

using the Eq. (A.14) and (A.19), we can change the Eq. (A.20) to:

$$a_{\text{curr}} = a_{\text{prev}} - (xwx)^{-1}(y-p)x \tag{A.21}$$

If we extend this equation to encompass updates for the parameters in a broader sense, we arrive at the following:

$$a_{\text{curr}} = a_{\text{prev}} - \left(\mathbf{x}'W\mathbf{x}\right)^{-1}(y-p)\mathbf{x} \tag{A.22}$$

where W is a matrix with the diagonal elements w that need to be optimized by tuning and finding the optimal values.

A.1.2 Support Vector Machine

Let us start with the following straight-line equation (A.23) to divide the data domain:

$$\mathbf{wx} + \mathbf{b} = 0 \tag{A.23}$$

Considering a data domain, this parameterized straight line divides the data domain into two subdomains, and we may call them left subdomain and right subdomain, denote them by D_1 and D_2, and define them as follows:

$$\begin{aligned} D_1 &= \{\mathbf{x} : \mathbf{wx} + \mathbf{b} \leq 0\} \\ D_2 &= \{\mathbf{x} : \mathbf{wx} + \mathbf{b} > 0\} \end{aligned} \tag{A.24}$$

The points falling in these subdomains may be distinguished with labels **1** for the subdomain D_1 and **-1** for the subdomain D_2. Therefore, the parametrization objective of the support vector machine can be defined as follows:

$$\begin{aligned} \mathbf{wx} + \mathbf{b} &= 1, \mathbf{x} \in D_1 \\ \mathbf{wx} + \mathbf{b} &= -1, \mathbf{x} \in D_2 \end{aligned} \tag{A.25}$$

In the parametrization objectives, we have modeled two straight lines (or hyperplanes) that can help to define boundaries between the classes. The optimization objective is to define an objective function (in this case, the distance between the straight lines) and search for the parameter values that maximize the distance. These lines are parallel to each other; therefore, we can simply use the standard distance formula between two parallel lines $y = mx + b_1$ and $y = mx + b_2$ as follows:

$$d = \frac{(b_2 - b_1)}{\sqrt{m^2 + 1}} \tag{A.26}$$

where the slopes of the straight lines are $m = \mathbf{w}$, and their intercepts are $b_1 = \mathbf{b} + 1$ and $b_2 = \mathbf{b} - 1$. By substituting these variables, we can establish the following:

$$d = \frac{\pm 2}{\sqrt{ww' + 1}} \tag{A.27}$$

Ultimately, this distance formula will be the measure for the optimization problem that we build; therefore, without loss of generality, we can rewrite it as follows:

$$d = \frac{\pm 2}{\sqrt{ww'}} \tag{A.28}$$

In practice, the support vector machine optimization problem is written using the mathematical *norm* notation, therefore we rewrite the above equation as follows:

$$d = \frac{\pm 2}{\|\mathbf{w}\|^2} \tag{A.29}$$

By squaring both sides of the equation, and then dividing both sides of the equation by the value of 2, we can obtain the following simple mathematical relationship:

$$\frac{d^2}{2} = \frac{1}{\frac{\|\mathbf{w}\|^2}{2}} \tag{A.30}$$

It states that instead of maximizing the distance function $d^2/2$, we can minimize $\|\mathbf{w}\|^2/2$. In other words, we can minimize the prediction error with respect to the above classifier while maximizing the distance between them (this is the optimization objective). Therefore, the following mathematical expression can be defined for the prediction error between $\mathbf{x} \in D$ and its response variables y:

$$e = 1 - y(\mathbf{w}\mathbf{x} + \mathbf{b}) \tag{A.31}$$

This error function plays a major role in the development of an optimization problem for the support vector machine.

Suppose there are p features $X_1, X_2, \ldots, X_p$ and the i th observation is denoted by $x_{i1}, x_{i2}, \ldots, x_{ip}$, where $i = 1, \ldots, n$. Then we can plot these n data points in a p-dimensional space to form a multidimensional space. This space is the vector space, and it displays both the magnitude and the directional information of the data. This set of p features may be transformed into a new set of d features using a polynomial kernel [3]. This space is called the feature space and, in general, each data point in the feature space carries information about a single data point in the vector space. The advantage of a feature space is that the nonseparable classes in the vector space may be turned into separable classes using the right choice of kernel. However, finding a kernel and generating such a transformation is not simple.

$$\phi : R^p \rightarrow R^d \tag{A.32}$$

where R^p is the vector space (original domain) and R^d is the feature space, which is high dimensional (generally $d >> p$). It is possible to find ϕ that helps transform

the vector space to a feature space where the classes are linearly separable. Hence, the support vector machine classifier in the feature space can be written as follows:

$$\begin{aligned} \underset{\mathbf{w},\mathbf{b}}{\text{Minimize}} \quad & \frac{\|\mathbf{w}\|^2}{2} \\ \text{subject to:} \quad & \mathbf{s}\left(\mathbf{w}\phi\left(\mathbf{x}\right)+\mathbf{bI}\right) \geq \mathbf{I} \end{aligned} \tag{A.33}$$

However, $\phi(\mathbf{x})$ is high dimensional, thus the computation becomes very expensive. Apart from the classification in linear space, the SVM approach can be readily applied for nonlinear classification as well by utilizing kernel tricks to implicitly map the inputs into high-dimensional feature spaces. Figure (2.3) explains one such mapping where a two-dimensional input space (x_1, x_2) is mapped into a three-dimensional feature space (z_1, z_2, z_3) by using the mapping, $z_1 = x_1^2, z_2 = \sqrt{2}x_1x_2, z_3 = x_2^2$.

A.1.3 Decision Tree Algorithm

Algorithm 9 outlines the step-by-step process for constructing a decision tree.

Algorithm 7: Decision Tree Construction

Parameters: Training dataset D, chosen splitting criterion (e.g., Gini impurity, entropy)
Result: Decision tree T

if *stopping criterion met* **then**
  return a leaf node with the majority class label
Select the best feature F and threshold T_F using the chosen splitting criterion
Create a decision node for feature F and threshold T_F
Partition the data into child datasets D_1 and D_2 based on F and T_F
$T_{\text{left}} \leftarrow$ Decision Tree Construction(D_1)
$T_{\text{right}} \leftarrow$ Decision Tree Construction(D_2)
Attach T_{left} as the left child node and T_{right} as the right child node of the current decision node
return the constructed decision tree T

A.1.4 Random Forest Algorithm

Algorithm 15 provides a comprehensive overview of the process for constructing a Random Forest, a robust ensemble learning method that combines multiple decision trees to improve predictive accuracy and reduce overfitting.

Algorithm 8: Random Forest Classification

Parameters: Training dataset D, Number of decision trees N_{trees}, Number of features considered N_{features}
Result: Random Forest model

```
Initialize an empty list, Forest, to store decision trees
for i = 1 to N_trees do
    Create a bootstrap sample D_i by randomly selecting N samples with replacement from D
    Randomly select N_features features from the total set of features
    Train a decision tree T_i using D_i and the selected features
    for each node in T_i do
        Randomly select a subset of N_features features
        Calculate the best split using a chosen criterion (e.g., Gini impurity)
    Add the trained tree T_i to the Forest
Make predictions using the Random Forest
for each input data point X do
    for each tree T_i in Forest do
        Traverse T_i and obtain a class prediction
    Aggregate predictions from all trees (e.g., a majority vote for classification)
return the Random Forest model
```

A.1.5 KNN Algorithm

Algorithm 8 outlines the K-Nearest Neighbors (KNN) classification method, which is a simple yet effective machine learning algorithm used for classifying data points based on the majority class among their nearest neighbors within a specified range, making it a popular choice for various classification tasks.

Algorithm 9: K-Nearest Neighbors (KNN) Classification

Parameters: Training dataset D, test data point X, number of neighbors K
Result: Predicted class label for X

for *each data point* D_i *in training dataset* D **do**
    Calculate the distance between X and D_i (e.g., Euclidean distance)
    Store the distance along with the class label of D_i
Sort the distances in ascending order
Select the top K data points with the smallest distances
Count the occurrences of each class label among the selected K data points
Assign the class label with the highest count as the predicted class label for X
return Predicted class label

A.1.6 Deep Learning

In figure 2.9:

- $x_k^{(s)}$: the k th input data at the sth set of data
- m: the total number of input data
- $n_{lr,p_{lr}}$: the p_{lr} th neural node of the lr th layer
- and N_{lr}: the total number of neurons in the lr th layer
- The notation $x^{(s)}$: the vector of input data
- N_{set} : the total number of data points in the input dataset
- L: the summation of the layers except the input layer

Equations below present the expressions for the output vectors of the first layer, lrth layer in the hidden layer and output layer, respectively. In the aforementioned equations, wlr represents the weighting vector of the lr th layer. The weighting values range between 0 and 1. These values change with the training data and represent the memory of the neural network related to the input and output after model training.

For $lr = 1$,

$$O_1^{(s)} = \varphi_1\left(x^{(s)} \times w_1 + b_1^T\right) \tag{A.34}$$

where

$$O_1^{(s)} = \left[\begin{array}{cccc} O_{1,1}^{(s)} & O_{1,2}^{(s)} & \cdots & O_{1,N_1}^{(s)} \end{array}\right] \tag{A.35}$$

$$w_1 = \left[\begin{array}{cccc} w_{1,1,1} & w_{1,2,1} & \cdots & w_{1,N_1,1} \\ w_{1,1,2} & w_{1,2,2} & & w_{1,N_1,2} \\ \vdots & \vdots & & \vdots \\ w_{1,1,m} & w_{1,2,m} & \cdots & w_{1,N_1,m} \end{array}\right] \tag{A.36}$$

$$b_1 = \left[\begin{array}{c} b_{1,1} \\ b_{1,2} \\ \vdots \\ b_{1,N_1} \end{array}\right] \tag{A.37}$$

For $L >| r \geq 2$,

$$O_{lr}^{(s)} = \varphi_1\left(O_{lr-1}^{(s)} \times w_{lr} + {b_{lr}}^T\right) \tag{A.38}$$

where

$$O_{lr}^{(s)} = \left[\begin{array}{cccc} O_{lr,1}^{(s)} & O_{lr,2}^{(s)} & \cdots & O_{lr,N_{lr}}^{(s)} \end{array}\right] \tag{A.39}$$

$$w_{lr} = \left[\begin{array}{cccc} w_{lr,1,1} & w_{lr,2,1} & \cdots & w_{lr,N_{lr},1} \\ w_{lr,1,2} & w_{lr,2,2} & & w_{lr,N_{lr},2} \\ \vdots & \vdots & \ddots & \\ w_{lr,1,N_{lr-1}} & w_{lr,2,N_{lr-1}} & \cdots & w_{lr,N_{lr},N_{lr-1}} \end{array}\right] \tag{A.40}$$

$$b_{lr} = \begin{bmatrix} b_{lr,1} \\ b_{lr,2} \\ \vdots \\ b_{lr,N_{lr}} \end{bmatrix} \tag{A.41}$$

For $lr = L$,

$$O_L^{(s)} = \varphi_2 \left(O_{L-1}^{(s)} \times w_L + b_L^T \right) \tag{A.42}$$

where

$$O_L^{(s)} = \begin{bmatrix} O_{lr,1}^{(s)} & O_{lr,2}^{(s)} & \cdots & O_{lr,N_{lr}}^{(s)} \end{bmatrix} \tag{A.43}$$

$$w_L = \begin{bmatrix} w_{L,1,1} & w_{L,2,1} & \cdots & w_{L,N_L,1} \\ w_{L,1,2} & w_{L,2,2} & & w_{L,N_L,2} \\ \vdots & \vdots & & \vdots \\ w_{L,1,N_{L-1}} & w_{L,2,N_{L-1}} & \cdots & w_{L,N_L,N_{L-1}} \end{bmatrix} \tag{A.44}$$

$$b_{lr} = \begin{bmatrix} b_{L,1} \\ b_{L,2} \\ \vdots \\ b_{L,N_L} \end{bmatrix} \tag{A.45}$$

The term $O_{lr}^{(s)}$ represents the output vector of the Irth layer after training from the first dataset to the sth dataset.

A.2 Clustering Methods

A.2.1 Kmeans Algorithm

Algorithm 10: K-Means Clustering Algorithm

Input: $X = \{x_1, x_2, \ldots, x_n\}$ (Dataset to be clustered), k (Number of required clusters) **Output:** $C = \{c_1, c_2, \ldots, c_k\}$ (Cluster centroids)
Initialization:
for $i = 1$ *to* k **do**
    Initialize cluster c_i randomly from X
while *Convergence not reached* **do**
    // Distance calculations
    for $i = 1$ *to* n **do**
        for $j = 1$ *to* k **do**
            Compute the Euclidean distance from data object x_i to cluster c_j
    // Data object assignment
    for $i = 1$ *to* n **do**
        Assign data object x_i to the closest cluster c_j
    // Update cluster centroids
    for $j = 1$ *to* k **do**
        Update cluster c_j to the mean of data objects assigned to it
End

A.2.2 Apriori Algorithm

The process of Apriori algorithm can be summarized as follows:

where we begin with item sets of size 1 and find all frequent item sets, which have support greater than a predefined threshold. After that candidate item sets of size, $k+1$ are generated from the frequent item sets of size k. To do this, join two frequent item sets of size k only if their first $k-1$ items are identical. Prune any item sets that contain subsets not in the frequent item set of size k.

then we count the support (number of transactions containing the item set) for each candidate item set and remove any candidates below the support threshold. The

Algorithm 11: Apriori Algorithm

Parameters: Transaction database T, support threshold $min\_support$, confidence threshold $min\_confidence$

Result: Association rules

```
L_1 ← find_frequent_1_itemsets(T, min_support);
L ← L_1;
k ← 2;
while L_{k-1} ≠ ∅ do
    C_k ← generate_candidate_itemsets(L_{k-1});
    L_k ← find_frequent_itemsets(T, C_k, min_support);
    L ← L ∪ L_k;
    k ← k + 1;
A ← generate_association_rules(L, min_confidence);
return A;
```

item set size (k) is increased and goes back to the candidate generation step until no new candidate item sets can be generated. Finally, Once all frequent item sets are found, we generate association rules based on them. An association rule typically has the form $A \rightarrow B$, where A and B are item sets. These rules express the likelihood that if A is in the basket, then B is also in the basket.

Appendix B

Python Codes

The Python code implementations for the activities conducted throughout this research are available in my GitHub repositories. You can access and download the code from the following repository link: https://github.com/bahmanskr.

In these repositories, you will find the Python scripts and Jupyter Notebooks used for data analysis, modeling, and other relevant tasks related to the research presented in this thesis. Feel free to explore the code and documentation for a detailed understanding of the methods and techniques used in the study. We encourage readers and researchers interested in replicating or building upon this work to make use of the code and resources available in the GitHub repository.