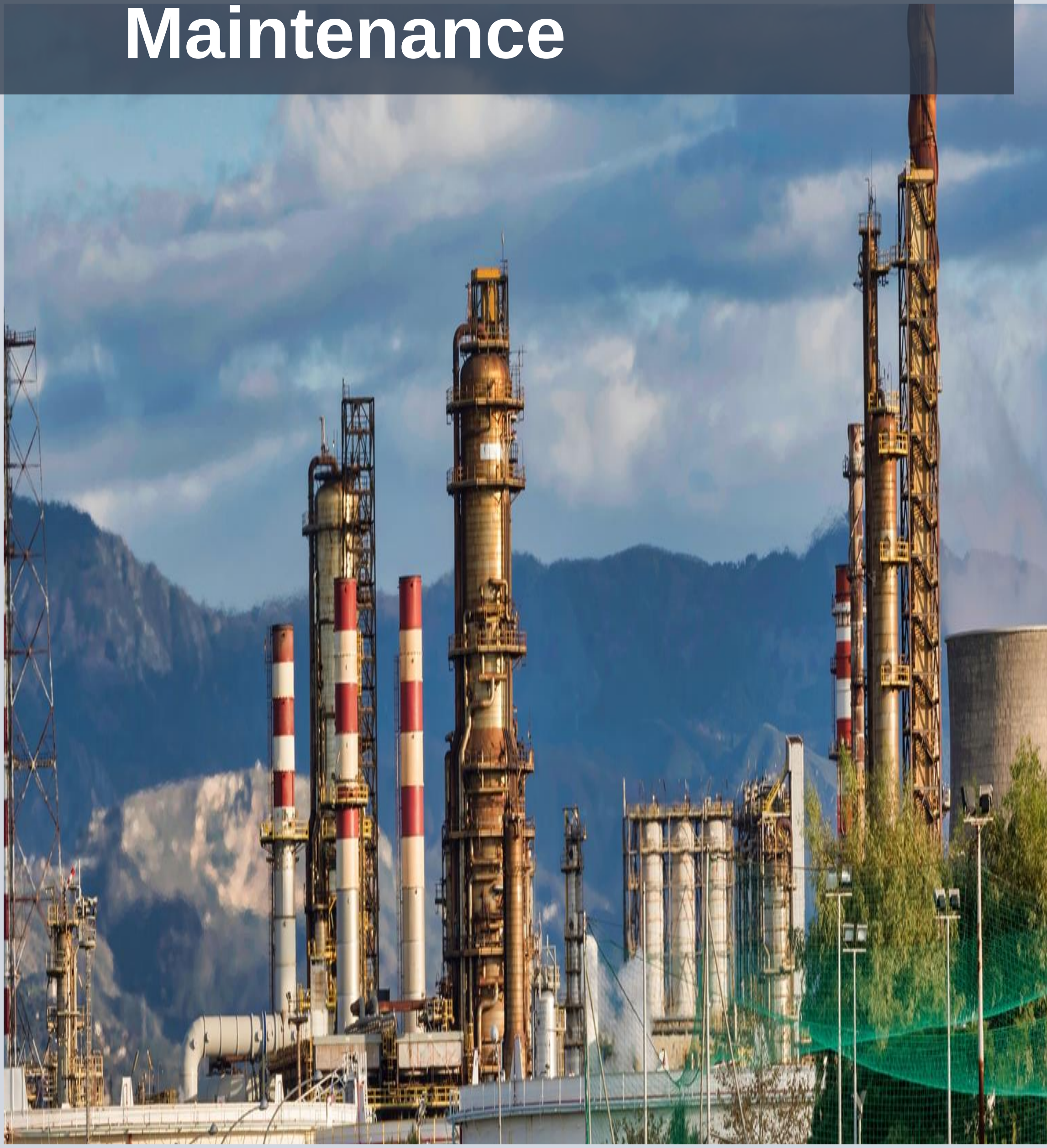
ML for Process Industry Series

Machine Learning in Python for Process and Equipment Condition Monitoring, and Predictive Maintenance

From Data to Process Insights



2024

First Edition

Ankur Kumar, Jesus Flores-Cerrillo

Machine Learning in Python for Process and Equipment Condition Monitoring, and Predictive Maintenance

From Data to Process Insights

Ankur Kumar
Jesus Flores-Cerrillo

Dedicated to our spouses, families, friends, motherlands, and all the data-science enthusiasts

आचार्यात्पादमादत्ते   पादं शिष्यः स्वमेधया ।
पादं सब्रह्मचारिभ्यः   पादं कालक्रमेण च ॥

A student receives a quarter (of his/her learning) from the teacher,
a quarter by way of his/her intelligence,
a quarter from fellow students,
and a quarter through the course of time.

- *A popular Sanskrit shloka*

Machine Learning in Python for Process and Equipment Condition Monitoring, and Predictive Maintenance

www.MLforPSE.com

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented and obtain permissions for usage of copyrighted materials. However, the authors make no warranties, expressed or implied, regarding errors or omissions, and assume no legal liability or responsibility for loss or damage resulting from the use of information contained in this book.

Plant image on cover page obtained from https://pixabay.com/.
.

To request permissions, contact the authors at MLforPSE@gmail.com

First published: January 2024

About the Authors

Ankur Kumar holds a PhD degree (2016) in Process Systems Engineering from the University of Texas at Austin and a bachelor's degree (2012) in Chemical Engineering from the Indian Institute of Technology Bombay. He currently works at Linde in the Advanced Digital Technologies & Systems Group in Linde's Center of Excellence, where he has developed several in-house machine learning-based monitoring and process control solutions for Linde's hydrogen and air-separation plants. Ankur's tools have won several awards both within and outside Linde. One of his tools, PlantWatch (a plantwide fault detection and diagnosis tool), received the 2021 Industry 4.0 Award by the Confederation of Industry of the Czech Republic. Ankur has authored or co-authored several peer-reviewed journal papers (in the areas of data-driven process modeling and monitoring), is a frequent reviewer for many top-ranked Journals, and has served as Session Chair at several international conferences. Ankur served as an Associate Editor of the Journal of Process Control from 2019 to 2021, and currently serves on the Editorial Advisory Board of Industrial & Engineering Chemistry Research Journal. Most recently, he was included in the 'Engineering Leaders Under 40, Class of 2023' by *Plant Engineering* Magazine.

Jesus Flores-Cerrillo is currently an Associate Director - R&D at Linde and manages the Advanced Digital Technologies & Systems Group in Linde's Center of Excellence. He has over 20 years of experience in the development and implementation of monitoring technologies and advanced process control & optimization solutions. Jesus holds a PhD degree in Chemical Engineering from McMaster University and has authored or co-authored more than 40 peer-reviewed journal papers in the areas of multivariate statistics and advanced process control among others. His team develops and implements novel plant monitoring, machine learning, IIOT solutions to improve the efficiency and reliability of Linde's processes. Jesus's team received the Artificial Intelligence and Advanced Analytics Leadership 2020 award from the National Association of Manufacturers' Manufacturing Leadership Council.

Note to the readers

Jupyter notebooks and Spyder scripts with complete code implementations are available for download at https://github.com/ML-PSE/Machine_Learning_for_PM_and_PdM. Code updates, when necessary, will be made and updated on the GitHub repository. Updates to the book's text material will be available on Leanpub (www.leanpub.com) and Google Play (https://play.google.com/store/books). We would greatly appreciate any information about any corrections and/or typos in the book.

Series Introduction

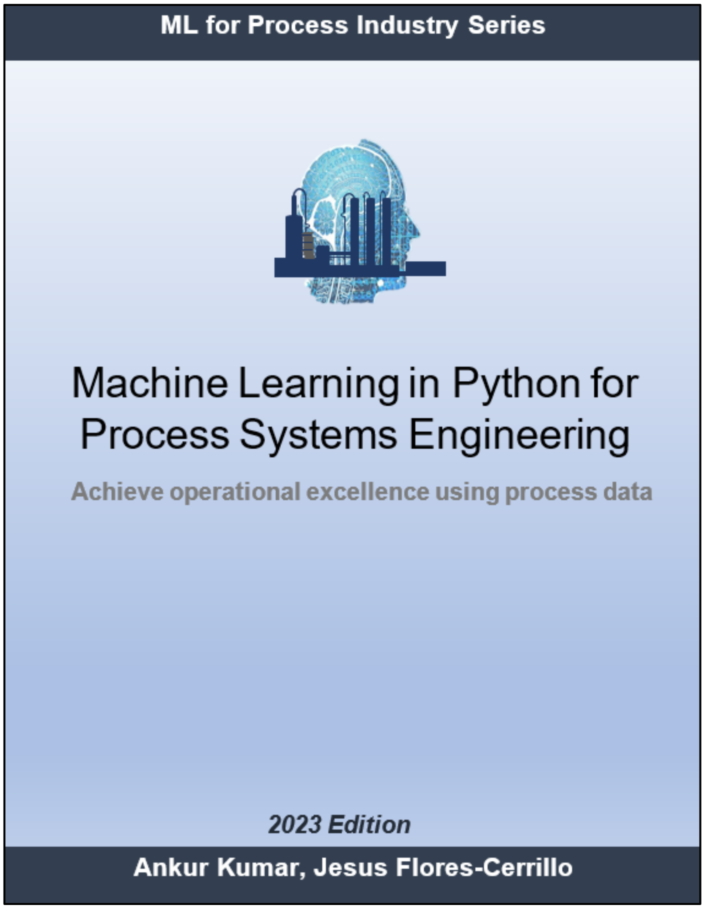
In the 21st century, data science has become an integral part of the work culture at every manufacturing industry and process industry is no exception to this modern phenomenon. From predictive maintenance to process monitoring, fault diagnosis to advanced process control, machine learning-based solutions are being used to achieve higher process reliability and efficiency. However, few books are available that adequately cater to the needs of budding process data scientists. The scant available resources include: 1) generic data science books that fail to account for the specific characteristics and needs of process plants 2) process domain-specific books with rigorous and verbose treatment of underlying mathematical details that become too theoretical for industrial practitioners. Understandably, this leaves a lot to be desired. Books are sought that have process systems in the backdrop, stress application aspects, and provide a guided tour of ML techniques that have proven useful in process industry. This series '***Machine Learning for Process Industry***' addresses this gap to reduce the barrier-to-entry for those new to process data science.

The first book of the series '***Machine Learning in Python for Process Systems Engineering***' covers the basic foundations of machine learning and provides an overview of broad spectrum of ML methods primarily suited for static systems. Step-by-step guidance on building ML solutions for process monitoring, soft sensing, predictive maintenance, etc. are provided using real process datasets. Aspects relevant to process systems such as modeling correlated variables via PCA/PLS, handling outliers in noisy multidimensional dataset, controlling processes using reinforcement learning, etc. are covered. The second book of the series '***Machine Learning in Python for Dynamic Process Systems***' focuses on dynamic systems and provides a guided tour along the wide range of available dynamic modeling choices. Emphasis is paid to both the classical methods (ARX, CVA, ARMAX, OE, etc.) and modern neural network methods. Applications on time series analysis, noise modeling, system identification, and process fault detection are illustrated with examples. This third book of the series takes a deep dive into an important application area of ML, viz, prognostics and health management. ML methods that are widely employed for the different aspects of plant health management, namely, fault detection, fault isolation, fault diagnosis, and fault prognosis, are covered in detail. Emphasis is placed on conceptual understanding and practical implementations. Future books of the series will continue to focus on other aspects and needs of process industry. It is hoped that these books can help process data scientists find innovative ML solutions to the real-world problems faced by the process industry.

With the growing trend in usage of machine learning in the process industry, there is growing demand for process domain experts/process engineers with data science/ML skills. These books have been written to cover the existing gap in ML resources for such process data scientists. Specifically, books of this series will be useful to budding process data scientists, practicing process engineers looking to ‘pick up’ machine learning, and data scientists looking to understand the needs and characteristics of process systems. With the focus on practical guidelines and industrial-scale case studies, we hope that these books lead to wider spread of data science in the process industry.

Other book(s) from the series
(https://MLforPSE.com/books/)

Book 1



ML for Process Industry Series

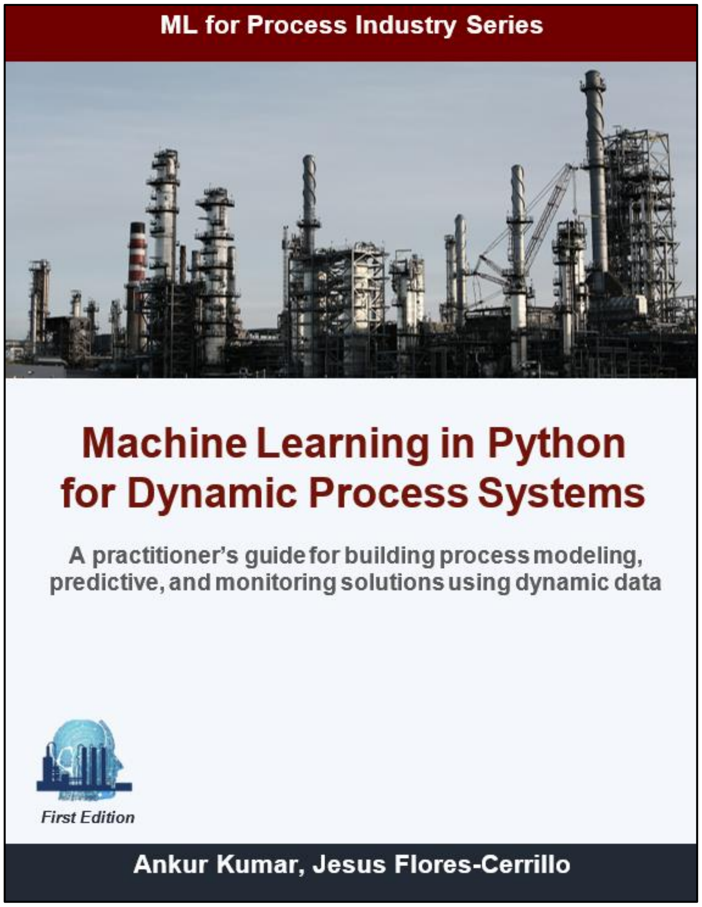
Machine Learning in Python for Process Systems Engineering

Achieve operational excellence using process data

2023 Edition

Ankur Kumar, Jesus Flores-Cerrillo

Book 2



ML for Process Industry Series

Machine Learning in Python for Dynamic Process Systems

A practitioner's guide for building process modeling, predictive, and monitoring solutions using dynamic data

First Edition

Ankur Kumar, Jesus Flores-Cerrillo

Preface

Imagine yourself in the shoes of a process engineer/analyst who has been assigned his/her first machine learning-based project with the objective of building a plantwide monitoring tool. Although an exciting task, it may easily turn into a frustrating effort due to the difficulty in finding the right methodology that works for the process system at hand. Building a successful process monitoring tool is challenging due to the different characteristics a process dataset may possess which precludes the possibility of a single methodology that works for all scenarios. Consequently, a number of powerful techniques have been devised over the past several decades. While it is good to be spoilt with choices, it is easy for a newcomer to get 'drowned' in the huge (and still burgeoning) literature on process monitoring (PM) and predictive maintenance (PdM). There are a lot of scattered resources on PM and PdM. However, unfortunately, no textbook exists that focusses on practical implementation aspects and provides comprehensive coverage of commonly used PM/PdM techniques that have proven useful in process industry. There is a gap in available machine learning resources for PM/PdM catering to industrial practitioners and this book attempts to cover this gap. Specifically, we wished to create a reader-friendly and easy-to-understand book that can help its readers become 'experts' on ML-based PM/PdM 'quickly' (disclaimer: there is no magic potion; hard work is still required!) with the right guidance.

In this book, we cover all three main aspects of process monitoring and predictive maintenance, namely, fault/anomaly detection, fault diagnosis/identification, and fault prognostics/remaining useful life estimation (RUL). Our intent is not to give a full treatise on all the PM/PdM techniques that exist out there; albeit our focus is to help budding process data scientists (PDSs) gain a bird's-eye view of the PM/PdM landscape, obtain working-level knowledge of the mainstream techniques, and have the practical know-how to make the right choice of models. In terms of the spectrum of methodologies covered, we place equal emphasis on modern deep-learning methods and classical statistical methods. While deep-learning has provided remarkable results in recent times, the classical statistical (and ML/data mining) methods are not yet obsolete. Infact MSPM (multivariate statistical process monitoring) techniques are still widely used for process monitoring. Accordingly, this book covers the complete spectrum of methodologies with univariate Shewhart-/CUSUM-/EWMA-based control charts on one end and deep-learning-based RUL estimations on the other.

Guided by our own experiences from building monitoring models for varied industrial applications over the past several years, this book covers a curated set of ML techniques that have proven useful for PM/PdM. The broad objectives of the book can be summarized as follows:

- reduce barrier-to-entry for those new to the field of PM/PdM
- provide working-level knowledge of PM/PdM techniques to the readers
- enable readers to make judicious selection of PM/PdM techniques appropriate for their problems through intuitive understanding of the advantages and drawbacks of different techniques
- provide step-by-step guidance for developing industrial level solutions for PM/PdM
- provide practical guidance on how to choose model hyperparameters judiciously

This book adopts a tutorial-style approach. The focus is on guidelines and practical illustrations with a delicate balance between theory and conceptual insights. Hands-on-learning is emphasized and therefore detailed code examples with industrial-scale datasets are provided to concretize the implementation details. A deliberate attempt is made to not weigh readers down with mathematical details, but rather use it as a vehicle for better conceptual understanding. Complete code implementations have been provided in the GitHub repository.

We are quite confident that this text will enable its readers to build process monitoring and prognostics models for challenging problems with confidence. We wish them the best of luck in their career.

Who should read this book

The application-oriented approach in this book is meant to give a quick and comprehensive coverage of mainstream PM/PdM methodologies in a coherent, reader-friendly, and easy-to-understand manner. The following categories of readers will find the book useful:

1) Data scientists new to the field of process monitoring, equipment condition monitoring, and predictive maintenance
2) Regular users of commercial anomaly detection software (such as Aspen Mtell) looking to obtain a deeper understanding of the underlying concepts
3) Practicing process data scientists looking for guidance for developing process monitoring and predictive maintenance solutions
4) Process engineers or process engineering students making their entry into the world of data science
5) Industrial practitioners looking to build fault detection and diagnosis solutions for rotating machinery using vibration data

Pre-requisites

No prior experience with machine learning or Python is needed. Undergraduate-level knowledge of basic linear algebra and calculus is assumed.

Book organization

The book follows a holistic and hands-on approach to learning ML where readers first gain conceptual insight and develop intuitive understanding of a methodology, and then consolidate their learning by experimenting with code examples. Under the broad theme of ML for process systems engineering, this book is an extension of the first two book of the series (which dealt with fundamentals of ML, varied applications of ML in process industry, and ML methods for dynamic system modeling); however, it can also be used as a standalone text. Industrial process data could show varied characteristics such as multidimensionality, non-Gaussianity, multimodality, nonlinearity, dynamics, etc. Therefore, to give due treatment to the different modeling methodologies designed for dealing with systems with different data characteristics, this book has been divided into seven parts.

Part 1 lays down the basic foundations of ML-assisted process and equipment condition monitoring, and predictive maintenance. **Part 2** provides in-detail presentation of classical ML techniques for univariate signal monitoring. Different types of control charts and time-series pattern matching methodologies are discussed. **Part 3** is focused on the widely popular multivariate statistical process monitoring (MSPM) techniques. Emphasis is paid to both the fault detection and fault isolation/diagnosis aspects. **Part 4** covers the process monitoring applications of classical machine learning techniques such as k-NN, isolation forests, support vector machines, etc. These techniques come in handy for processes that cannot be satisfactorily handled via MSPM techniques. **Part 5** navigates the world of artificial neural networks (ANN) and studies the different ANN structures that are commonly employed for fault detection and diagnosis in process industry. **Part 6** focusses on vibration-based monitoring of rotating machinery and **Part 7** deals with prognostic techniques for predictive maintenance applications.

This book attempts to cover a lot of concepts. Therefore, to avoid the book from getting bulky, we have not included contents that are not directly relevant to PM/PdM and have already been covered in detail in the first two books of the series. For example, ML fundamentals related to cross-validation, regularization, noise removal, etc., are illustrated in great detail in Book 1 of the series and not in this book.

Symbol notation

The following notation has been adopted in the book for representing different types of variables:

- lower-case letters refer to vectors ($x \in \mathbb{R}^{m \times 1}$) and upper-case letters denote matrices ($X \in \mathbb{R}^{n \times m}$)
- individual element of a vector and a matrix are denoted as x_j and x_{ij}, respectively.
- any *i*th vector in a dataset gets represented as subscripted lower-case letter ($x_i \in \mathbb{R}^{m \times 1}$)

Table of Contents

Part 1: Introduction and Fundamentals

Part 2: Univariate Signal Monitoring

Chapter 5: Control Charts for Statistical Process Control

Chapter 6: Process Fault Detection via Time Series Pattern Matching

Part 3: Multivariate Statistical Process Monitoring

Chapter 7: Multivariate Statistical Process Monitoring for Linear and Steady-State Processes: Part 1

Chapter 8: Multivariate Statistical Process Monitoring for Linear and Steady-State Processes: Part 2

Part 5: Artificial Neural Networks for Process Monitoring

Part 1

Introduction & Fundamentals

Chapter 1

Machine Learning, Process and Equipment Condition Monitoring, and Predictive Maintenance: An Introduction

Ask a plant manager about what gives him/her sleepless nights and you will invariably get plant equipment failures and process abnormalities causing downtimes among the top answers. Such concerns about plant reliability are not unfounded. Incipient abnormalities, if left undetected, can cause cascading damages leading to economic losses, plant downtimes, and even fatalities. Several major disasters in the process industry (Philadelphia refinery explosion in 2019, Bhopal gas tragedy in 1984, etc.) were the results of failures in timely detection and correction of process faults. While such disasters are fortunately infrequent, 'innocuous' process abnormalities that lead to non-optimal plant efficiencies and degradations in product quality occur routinely. Without exaggeration, it can be said that 24X7 monitoring of process performance and plant equipment health status, and forecast of impending failures are no longer a 'nice to have' but an absolute necessity!

Process industry has responded to the above challenges by putting more sensors and collecting more real-time data. Unfortunately, this has led to data deluge and operators being overwhelmed with 'too much information but little insights'. Thankfully, machine learning comes to the rescue with its ability to parse huge amount of data and find hidden patterns in real-time. ML allows smart process monitoring (PM) wherein objective is not just to detect process abnormalities but to catch the issues at early stages. Furthermore, ML facilitates predictive maintenance (PdM) through advance prediction of equipment failure times.

In this chapter we will take a bird's-eye view of the ML landscape for PM/PdM and understand what it takes to achieve the above objectives. Specifically, the following topics are covered

- Introduction to process/equipment abnormalities and faults
- Typical workflow for ML-based process monitoring and predictive maintenance
- ML landscape for process monitoring and predictive maintenance
- Common PM/PdM solution deployment infrastructure employed in industry

Machine learning is a great tool, but it's not magic; it still takes a lot 'ML art' to get the things right. Let's now take the first step towards mastering this art.

1.1 Process Industry and ML-based Plant Health Management

Process industry is a parent term used to refer to industries like petrochemical, oil & gas, chemical, power, paper, cement, pharmaceutical, etc. These industries use processing plants to manufacture intermediate or final consumer products. As emphasized in Figure 1.1, the prime concerns of the management of these plants include, amongst others, optimal and safe operations, quality control, and high reliability through proactive process monitoring and predictive maintenance. All these tasks fall under the ambit of process systems engineering (PSE). While ML is being slowly incorporated in the PSE tasks (for example, deep learning-based process controller[1]), ML has had the biggest influence on the tasks related to plant health management, viz, fault detection, fault diagnosis, and predictive maintenance.

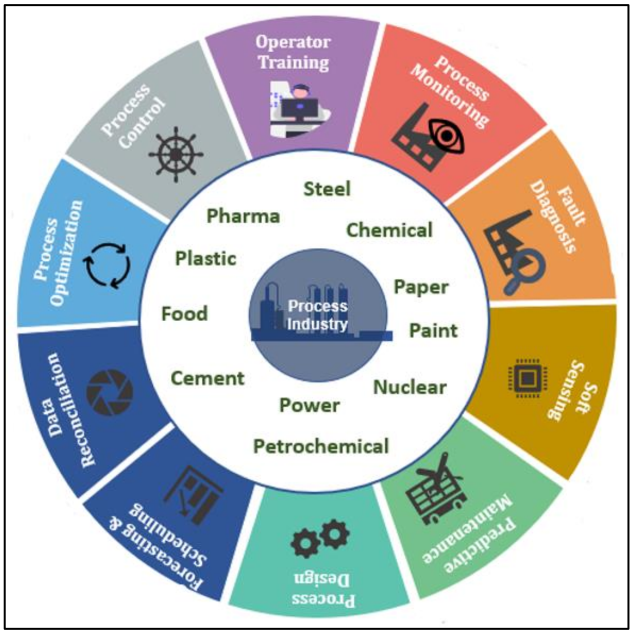


Figure 1.1: Overview of industries constituting process industry and the common PSE tasks

Figure 1.2 shows a sample process flowsheet with traditional measurements of flow, temperature, pressure, level, composition, power, and vibration. Such complex and highly integrated operations, tight product specifications, and the economic compulsion to push processes to their limits are making industrial operations more prone to failures. Nonetheless, there is an increasing trend to have unmanned or lean-staffed plants with less human eyes to monitor the process. This is where automated plant health management comes into play to resolve this dichotomy. Process models combined with sensor data are used for continuous monitoring of processes to detect, isolate, and diagnose faults, and for predicting fault

[1] https://www.aspentech.com/en/products/msc/aspen-dmc3

progression. The obvious gains are prevention of costly downtimes through better planned maintenance. For developing process models, data-driven/ML models have become more popular due to the relative ease of implementation and model maintenance compared to first principle-based models.

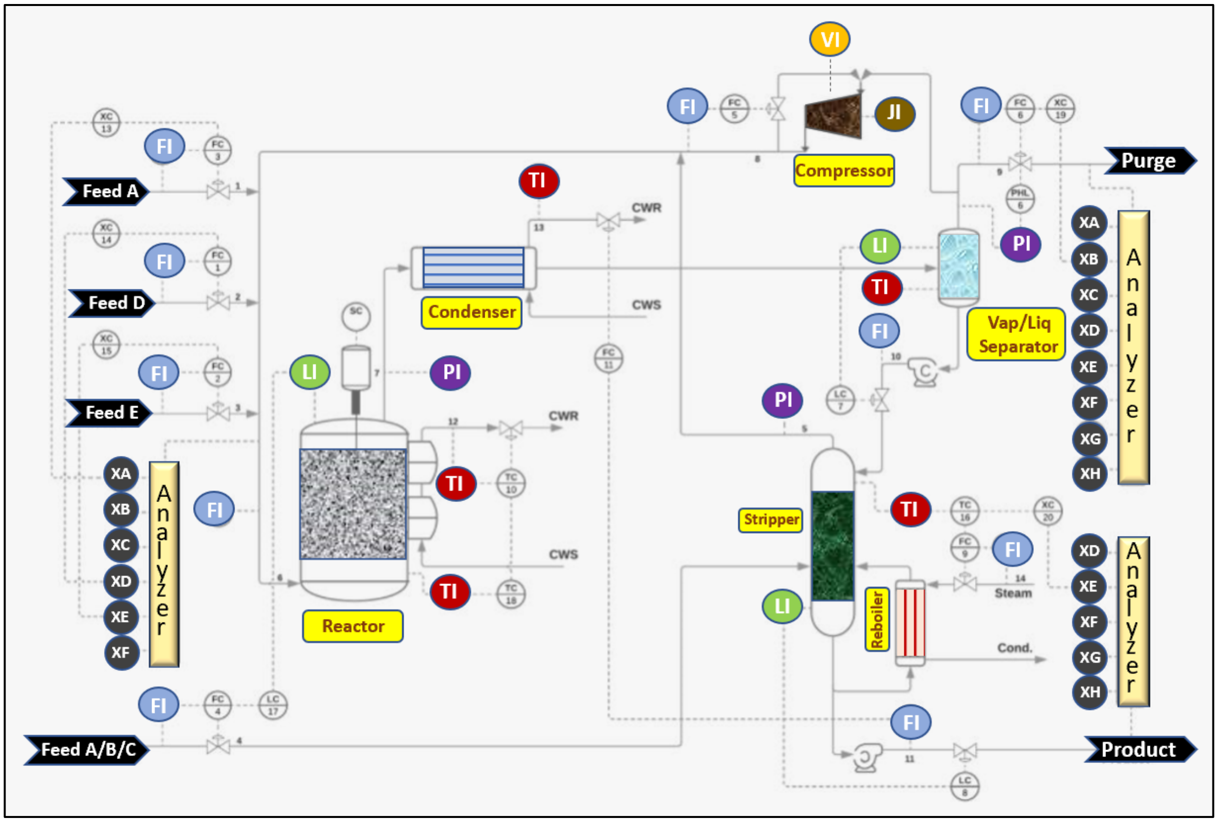


Figure 1.2: A typical process flowsheet[2] with flow (FI), temperature (TI), pressure (PI), composition (Analyzers), level (LI), power (JI), vibration (VI) measurements.

Let's continue learning about the ML-based plant health management by first taking a closer look at the meaning of process faults and abnormalities.

What are process faults and abnormalities?

Colloquially speaking, process faults or abnormalities are unexpected and unfavorable deviations/patterns in process variables that defy the normal/acceptable process behavior. The deviations could be undesirable decreases in product yield and product purity, fluctuations in critical liquid levels, rise in temperatures, increase in rotating machine vibrations, etc. There are various causes of faults in process system including, amongst others, fouling, pipe blockages, leaks, catalyst poisoning, and valve stiction. The flowsheet below illustrates some common fault sources.

[2] Adapted from the original flowsheet by Gilberto Xavier (https://github.com/gmxavier/TEP-meets-LSTM) provided under Creative-Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

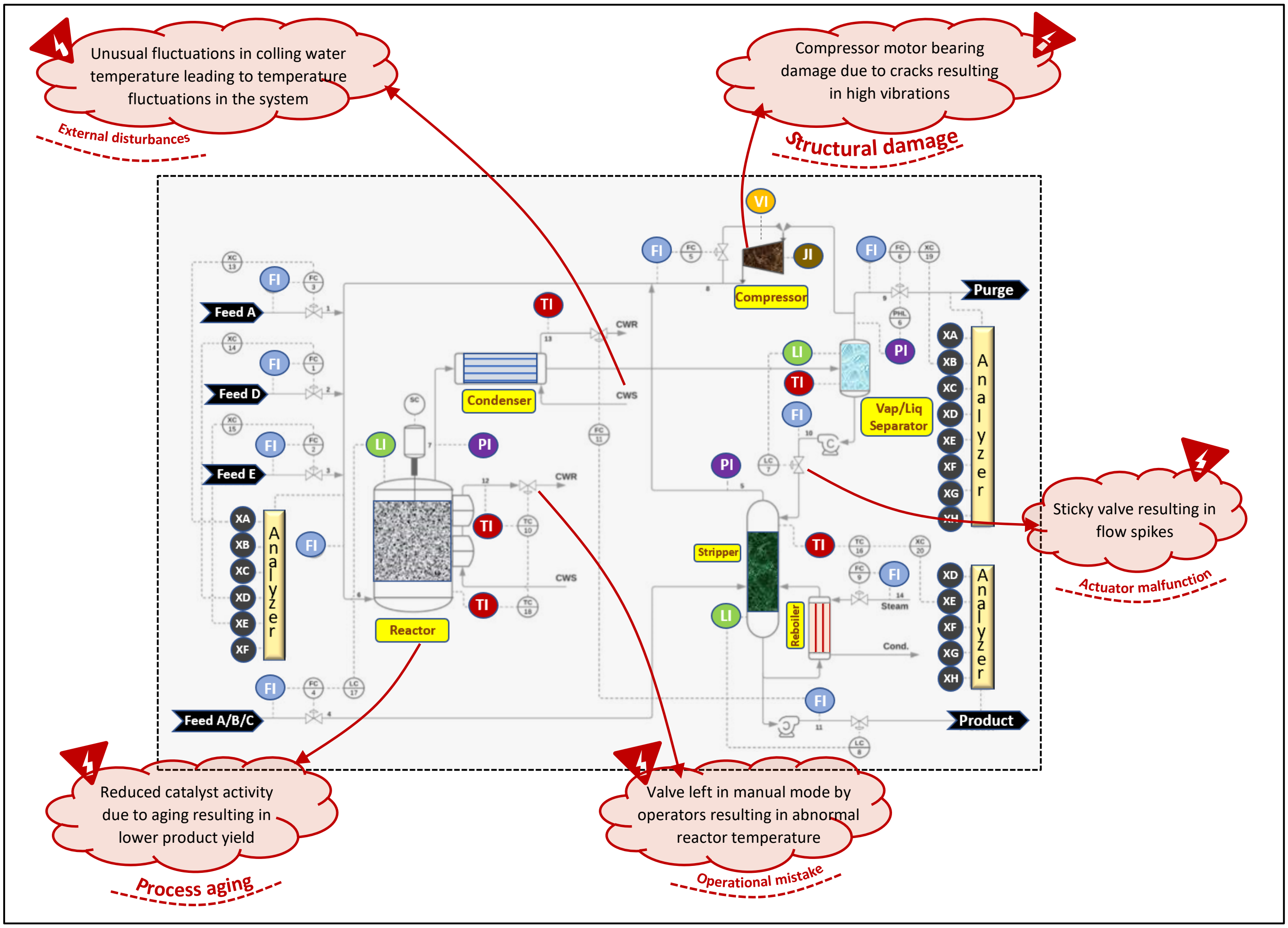
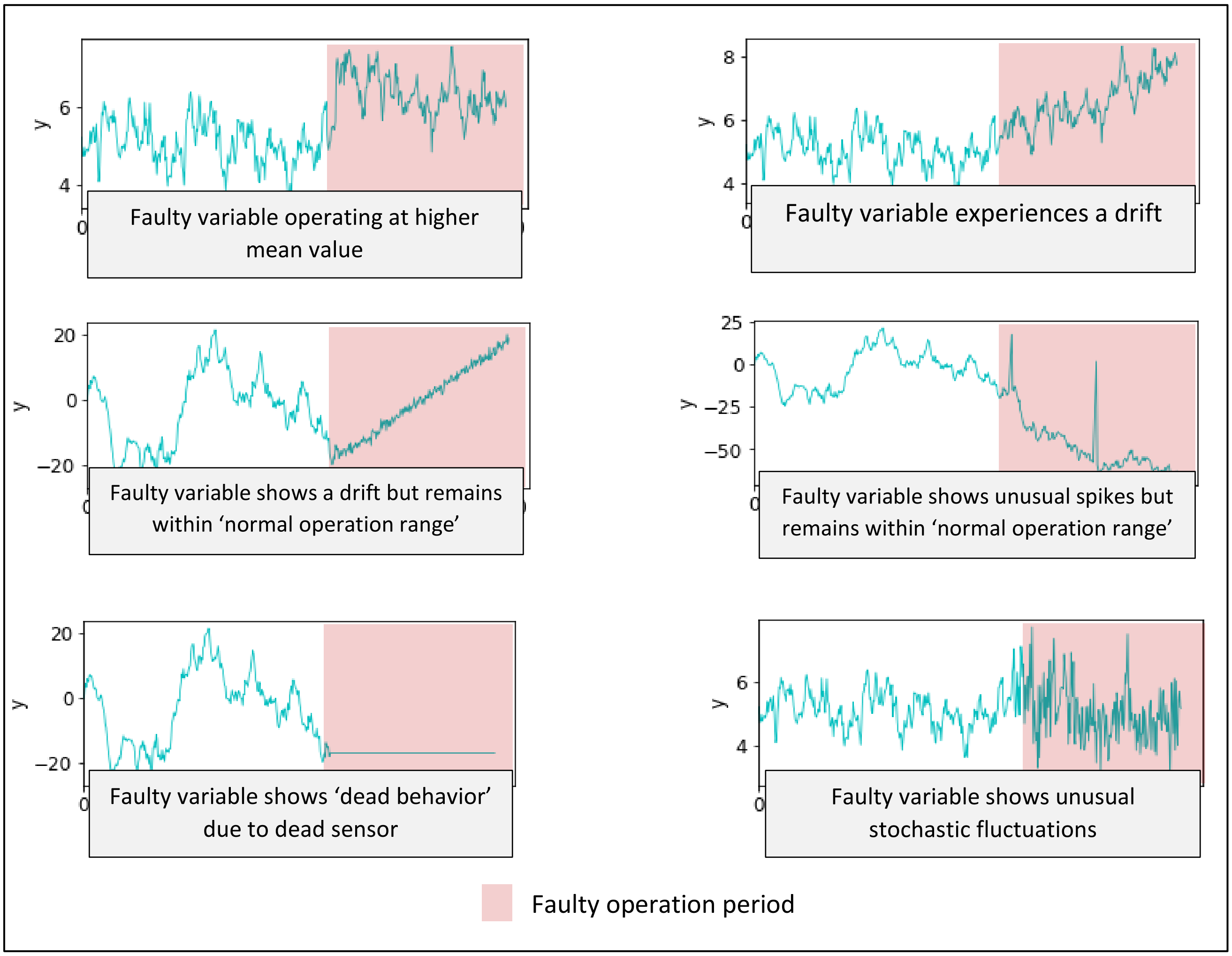

Unusual fluctuations in colling water temperature leading to temperature fluctuations in the system
External disturbances
Compressor motor bearing damage due to cracks resulting in high vibrations
Structural damage
Sticky valve resulting in flow spikes
Actuator malfunction
Reduced catalyst activity due to aging resulting in lower product yield
Process aging
Valve left in manual mode by operators resulting in abnormal reactor temperature
Operational mistake
Feed A
Feed D
Feed E
Feed A/B/C
Condenser
Reactor
Compressor
Vap/Liq Separator
Stripper
Reboiler
Analyzer
Purge
Product
CWR
CWS
Steam
Cond.
FI
TI
PI
LI
VI
JI
XA
XB
XC
XD
XE
XF
XG
XH

Figure 1.3: Some common sources of process faults in a process plant

Equipment monitoring vs plantwide monitoring

A common approach for process monitoring in process industry is to monitor the different critical equipment of a plant separately. The ML models are built separately for each equipment. While this approach makes ML model development easier as each ML model handles only a subset of the plant variables, one is left with having to maintain and analyze results from multiple ML models. An alternative approach is plantwide monitoring wherein the whole plant comprising of multiple equipment is monitored using a single ML model. The downside of this approach is high dimensionality of the variable-set and reduced fault detection performance. Same ML models can be employed for either of the approaches. Nonetheless, some specialized techniques have been devised for plantwide monitoring. We will remark upon these techniques as suitable in the upcoming chapters.

As remarked before, faults entail unwanted deviations in process variables. Figure below shows samples of data patterns that may be observed under the influence of process faults.



Faulty variable operating at higher mean value

Faulty variable experiences a drift

Faulty variable shows a drift but remains within ‘normal operation range’

Faulty variable shows unusual spikes but remains within ‘normal operation range’

Faulty variable shows ‘dead behavior’ due to dead sensor

Faulty variable shows unusual stochastic fluctuations

🟥 Faulty operation period

Figure 1.4: Sample of data patterns under faulty conditions

Figure 1.4 shows why automated fault detection is not a very straightforward activity. Under normal plant operation, process variables do not remain at fixed values but show stochastic fluctuations and normal variations due to changing plant load, product grade, ambient conditions, etc. Therefore, one can’t just compare each process variable against some fixed thresholds to ascertain the healthy state of the process. Modeling the multivariable relationships among the plant variables become indispensable in most of the scenarios.

The takeaway message is that modern process plants are prone to multiple failures, and it takes an ‘army’ to ensure reliable operations. In the industry 4.0 era, ML is being employed as that ‘army’. Before we look at the different ML models available at our disposal, let’s try and understand what exactly an ML model is expected to do.

1.2 Plant Health Management (PHM) Workflow

In the previous section we discussed in some detail the fault detection aspect of plant health management. However, it is only a part of the journey towards reliable plant operation. Figure 1.5 shows the different milestones of the journey. As shown, fault detection is followed by fault isolation or fault diagnosis wherein the objective is to identify the process variables that have been affected by the fault or determine the underlying cause of the fault, respectively. For example, for the valve malfunction problem illustrated in Figure 1.3, fault isolation pinpoints the flow from the separator to the stripper as the faulty variable and fault diagnosis pinpoints the valve stiction as the root cause of faulty behavior.

FDI vs FDD

In the process monitoring literature, you will find the acronyms FDI and FDD very often. FDI stands for fault detection and isolation, and FDD stands for fault detection and diagnosis. As alluded to before, although fault diagnosis is different from fault isolation, it is often used (incorrectly) to refer to the task of finding variables showing abnormal behavior.

Other terms that you may encounter are fault identification and fault classification. While fault identification is same as fault isolation, fault classification refers to categorizing/classifying a fault into one of several pre-defined fault types.

Following FDI/FDD, lies the task of fault prognosis which entails forecasting the progression of the identified fault. Fault prognosis helps to determine the amount of time left before the equipment affected by the fault needs to be taken out of service for maintenance or the whole plant needs to be shutdown for fault repair. For equipment-level monitoring, fault prognosis provides what is popularly known as remaining useful life (RUL). For example, for the compressor bearing damage problem illustrated in Figure 1.3, the vibrations will only be slightly higher than normal during the initial stages of crack development. However, with time the crack grows leading to greater and greater vibrations, and ultimately the compressor fails or becomes too dangerous to operate. A good fault prognostic model can accurately estimate the time left until the failure point of compressor is reached.

The advancement in fault prognosis algorithms have popularized the concept of predictive maintenance, wherein the plant management can plan well-in-advance the maintenance schedule based on actual equipment/process health condition. As you can imagine, this approach has obvious economic benefits (compared to time-based/preventive maintenance) and, to nobody's surprise, has caught fascination of process industry executives!

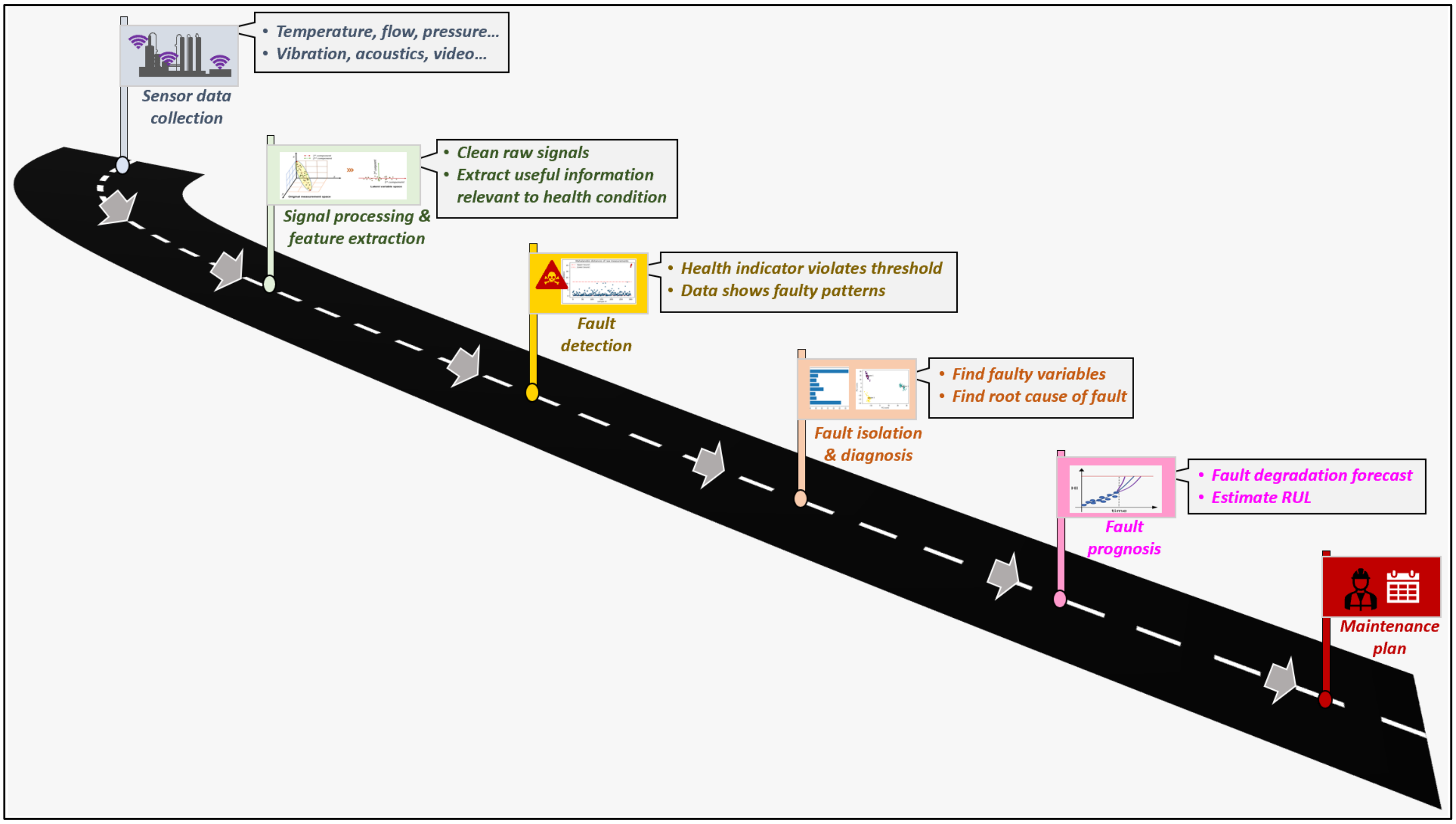

Sensor data collection
• Temperature, flow, pressure…
• Vibration, acoustics, video…
Signal processing & feature extraction
• Clean raw signals
• Extract useful information relevant to health condition
Fault detection
• Health indicator violates threshold
• Data shows faulty patterns
Fault isolation & diagnosis
• Find faulty variables
• Find root cause of fault
Fault prognosis
• Fault degradation forecast
• Estimate RUL
Maintenance plan

Figure 1.5: Plant health management workflow

Prognostic and Health Management

In industrial community, the PM/PdM workflow shown in Figure 1.5 is commonly called as prognostic and health management[3]. This includes both condition monitoring (fault detection, isolation, and diagnosis) and predictive maintenance (fault prognosis) aspects.

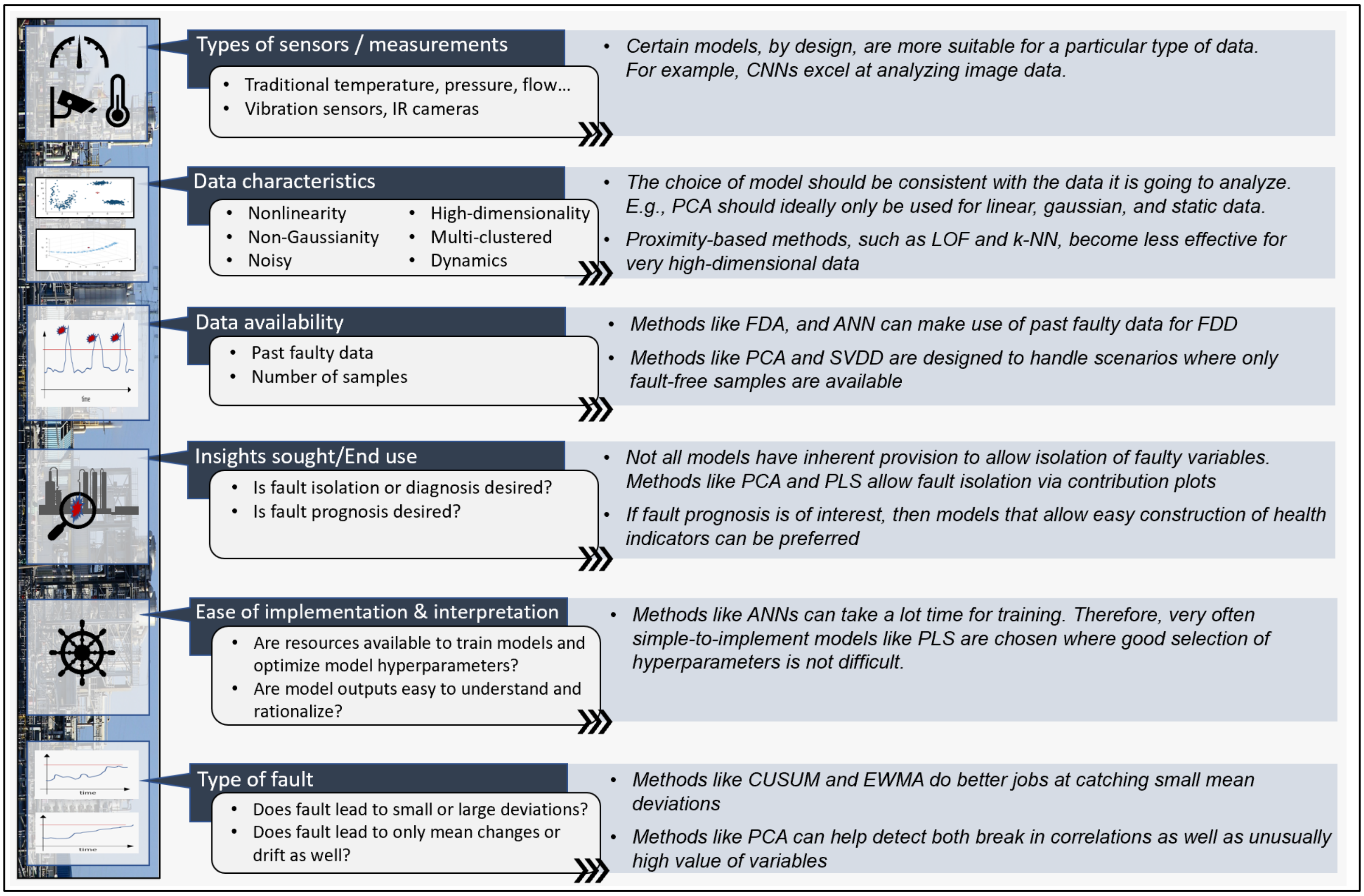
In the upcoming chapters, we will study in detail all the shown major aspects of PHM and learn how to implement the end-to-end solutions.

.

[3] Although the acronym 'PHM' is commonly used by the prognostic research community to refer to prognostic and health management, we will use it to denote 'plant health management' in this book.

1.3 ML Modeling Landscape for Plant Health Management

As process data scientists, we have to live with the harsh truth that there is no single universally good model for all occasions. One reason for this is that process data can show different characteristics (such as nonlinearity, non-Gaussianity, dynamics, multi-modality, etc.) which necessitates selection of different modeling methodologies. Additionally, the availability of historical faulty data, the user's end goal, and the type of installed sensors can also influence the model selection as shown in Figure 1.6. This makes the task of correct selection of ML model daunting (and potentially overwhelming for beginner PDSs). Fortunately, the recourse is open-secret and is as simple as having a good understanding of your data and system, and conceptually sound knowledge of pros and cons of the available methods.



Types of sensors / measurements

- Traditional temperature, pressure, flow…
- Vibration sensors, IR cameras

- Certain models, by design, are more suitable for a particular type of data. For example, CNNs excel at analyzing image data.

Data characteristics

- Nonlinearity
- Non-Gaussianity
- Noisy
- High-dimensionality
- Multi-clustered
- Dynamics

- The choice of model should be consistent with the data it is going to analyze. E.g., PCA should ideally only be used for linear, gaussian, and static data.
- Proximity-based methods, such as LOF and k-NN, become less effective for very high-dimensional data

Data availability

- Past faulty data
- Number of samples

- Methods like FDA, and ANN can make use of past faulty data for FDD
- Methods like PCA and SVDD are designed to handle scenarios where only fault-free samples are available

Insights sought/End use

- Is fault isolation or diagnosis desired?
- Is fault prognosis desired?

- Not all models have inherent provision to allow isolation of faulty variables. Methods like PCA and PLS allow fault isolation via contribution plots
- If fault prognosis is of interest, then models that allow easy construction of health indicators can be preferred

Ease of implementation & interpretation

- Are resources available to train models and optimize model hyperparameters?
- Are model outputs easy to understand and rationalize?

- Methods like ANNs can take a lot time for training. Therefore, very often simple-to-implement models like PLS are chosen where good selection of hyperparameters is not difficult.

Type of fault

- Does fault lead to small or large deviations?
- Does fault lead to only mean changes or drift as well?

- Methods like CUSUM and EWMA do better jobs at catching small mean deviations
- Methods like PCA can help detect both break in correlations as well as unusually high value of variables

Figure 1.6: Sample factors that influence ML model selection for PM/PdM

Before you embark upon modeling your process system, you would already have knowledge of the various factors listed in the above figure, except possibly for the data characteristics. We will study the techniques used to ascertain data characteristics in Chapter 3. Now that we understand the factors that influence model selection, we are ready to see what models are available at our disposal.

Traditional process measurements such as flow, temperature, pressure, composition, etc., and vibration measurements dominate the signals recorded in process industry. Therefore, case studies presented in this book use only these signals. Computer vision-based ML solutions are not covered.

Figure 1.7 below shows the modeling methodologies for process monitoring that we will cover in this book. Fault detection and diagnosis are precursors to fault prognosis and therefore the same methodologies are employed for building predictive maintenance solutions as well. As the category topics show, these methods cater to process data with different characteristics. The methods range from 'simple' traditional control charts to modern deep learning.

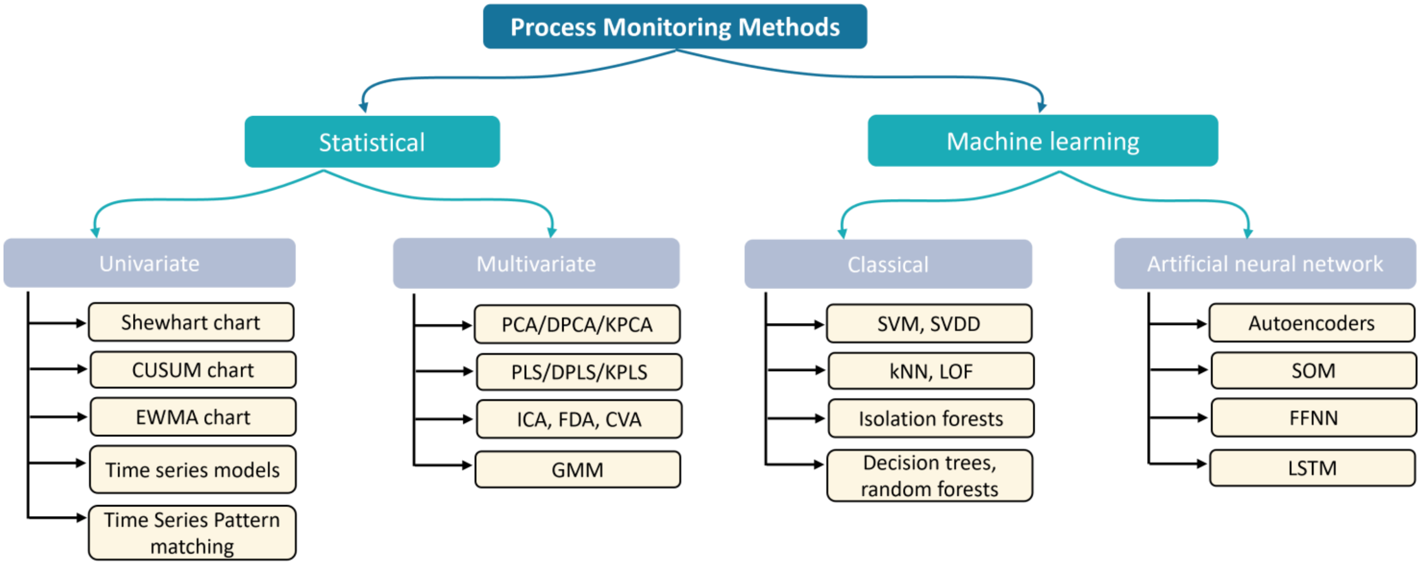


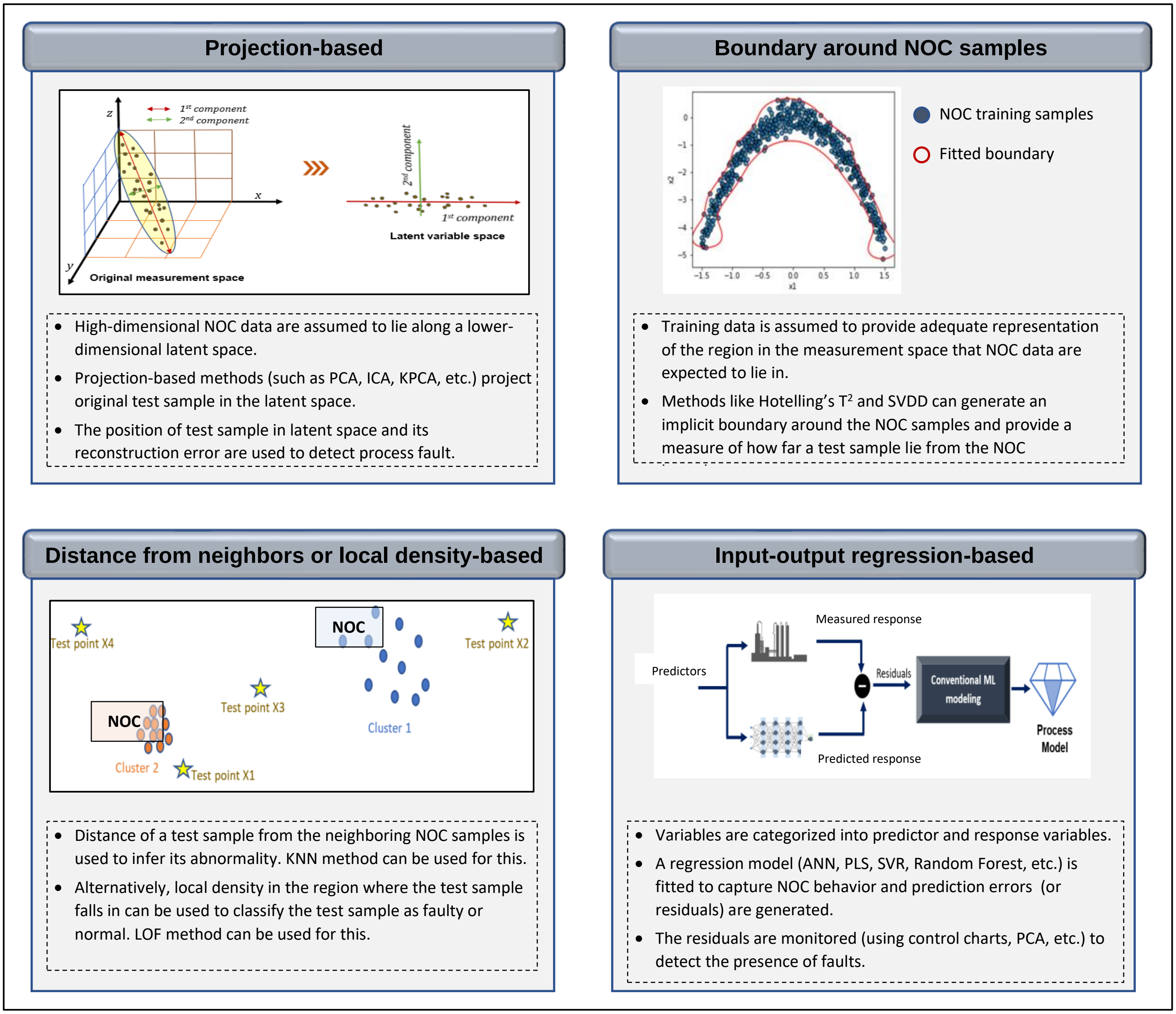
Figure 1.7: Model tree for process monitoring

The category of MSPM (multivariate statistical process monitoring) methods (PCA, PLS, GMM, etc.) deserves special attention as it has been the bedrock of health monitoring of complex process plants. A large section of the book will therefore cater to these methods. However, irrespective of their popularity, MSPM methods have shortcomings. Therefore, machine learning and deep learning models like Autoencoders, LSTMs, LOFs are covered as well.

In Figure 1.7, the modeling methodologies have been broadly divided into four categories, viz, univariate statistical models, multivariate statistical models, classical machine learning models, and artificial neural networks (ANN) models. Each of these categories are dealt with in separate parts of the book. The statistical[4] PM models extract a statistical model of the system using past data. Within this category lies simple control-chart models that are used for single variable monitoring. Though useful, these univariate models are understandably too restrictive to handle plantwide monitoring of complex industrial plants. On the other end of the model spectrum lies complex deep learning models that can theoretically handle any type of process systems; the downside is cumbersome model training procedure and hyperparameter optimization. In between these two extremes, lie the MSPM methods whose ease of implementation and interpretable results have led to wide popularity. However, MSPM methods tend to falter for highly nonlinear processes with complex data distributions. Therefore, classical ML and deep learning methods have been receiving considerable attention for process monitoring solution development for complex industrial processes.

The models in Figure 1.7 cater to the different scenarios that you may encounter in practice. If you have abundant past faulty samples then classification models such as FDA, SVM, ANN, etc. can be employed. However, in process industry, most of the time you will not have the luxury of having past faulty data and therefore, many of the fault detection techniques covered in this book cater to this scenario. The figure below illustrates the different principles employed to detect the presence of process faults using only NOC data during model training.

[4] In legacy process monitoring terminology, statistical process monitoring is also called statistical process control (SPC). Although SPC methods do not involve any feedback to the process controllers, the word ‘control’ signifies the objective of keeping the process ‘in-control’ through continuous monitoring.



Projection-based

- High-dimensional NOC data are assumed to lie along a lower-dimensional latent space.
- Projection-based methods (such as PCA, ICA, KPCA, etc.) project original test sample in the latent space.
- The position of test sample in latent space and its reconstruction error are used to detect process fault.

Boundary around NOC samples

- Training data is assumed to provide adequate representation of the region in the measurement space that NOC data are expected to lie in.
- Methods like Hotelling's T^2 and SVDD can generate an implicit boundary around the NOC samples and provide a measure of how far a test sample lie from the NOC

Distance from neighbors or local density-based

- Distance of a test sample from the neighboring NOC samples is used to infer its abnormality. KNN method can be used for this.
- Alternatively, local density in the region where the test sample falls in can be used to classify the test sample as faulty or normal. LOF method can be used for this.

Input-output regression-based

- Variables are categorized into predictor and response variables.
- A regression model (ANN, PLS, SVR, Random Forest, etc.) is fitted to capture NOC behavior and prediction errors (or residuals) are generated.
- The residuals are monitored (using control charts, PCA, etc.) to detect the presence of faults.

Figure 1.8: Popular fault detection methodologies using only NOC data

Note that the models in Figure 1.7 are applicable to both equipment level monitoring and plantwide monitoring. Let us now move to an overview of how these models are actually developed.

1.4 ML Model Development Workflow

In Figure 1.7, we saw different types of ML models for PM applications. Fortunately, the workflow for model development and deployment remains similar, and is shown in Figure 1.9. As is typical for a ML project, the tasks can be categorized into offline computations and online/real-time computations. In online computations, process data are parsed through the model to provide real-time insights and results. The models are built offline using historical process data. This offline exercise is performed once or repeated at regular intervals for model update. Brief description of the essential steps performed are provided below:

- *Exploratory data analysis:* Exploratory data analysis (EDA) involves preliminary investigation of data to get a 'feel' of the process dataset characteristics. The activities may include assessment of the presence of nonlinear relationships among process variables, non-Gaussian distribution, etc. Inferences made during EDA help make the right model selection. EDA is covered in detail in Chapter 3.

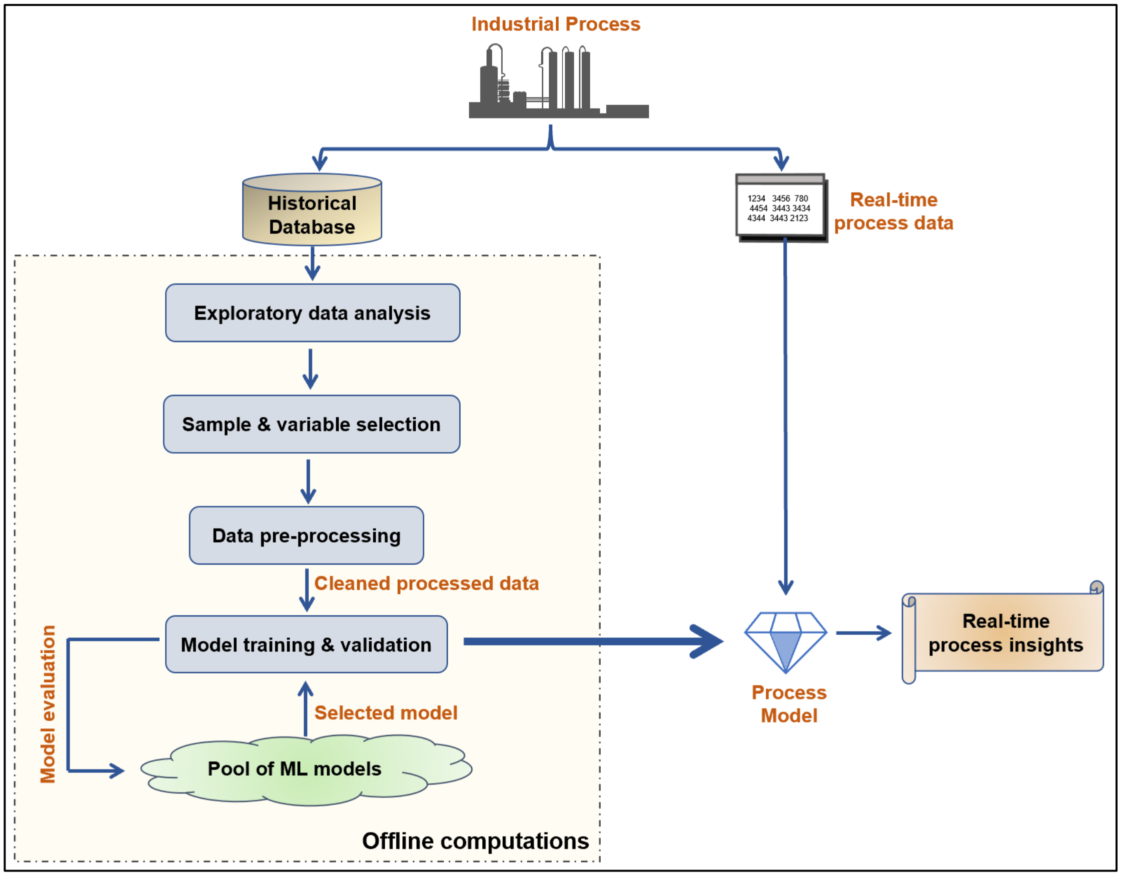


Figure 1.9: Steps involved in a typical ML model development for process monitoring

- *Sample and variable selection:* One does not simply dump all the available historical data and sensor measurements into a model training module. If a model is being built to identify the normal process behavior, then care must be taken to include only samples from fault-free operations in the model training dataset. Furthermore, if your model does not handle dynamics then data from periods of process transitions should be excluded.

  Variable selection warrants judicious consideration as well. Inclusion of unnecessary variables makes data noisier and reduces effectiveness of fault detection model. A generic guidance is to include only those variables that can assist in early fault detection; a variable that does not show any change in behavior under the influence of process faults of interest should be excluded.

- *Data pre-processing*: “*Garbage in, garbage out*” is an age-old principle in computer simulations. The same holds for ML model training for PHM. Your model will be practically useless if training data is not ‘clean’. Your process monitoring model won’t be able to detect process abnormalities accurately if it has been trained with outlier-infested training data. Data pre-processing includes, amongst others, identification and removal of outliers, noise reduction, transformation of variables, and extraction of features. The overall objective of this step is to increase the ‘information content’ of your training dataset so that the PM model’s ability to distinguish between normal and faulty operations is bolstered. Several aspects of data pre-processing are dealt with in Chapter 4.

- *Model training and validation*: Model training imply estimating the parameters of the chosen ML model, for example, the neuron weights in an ANN model. Model validation is employed for finding optimal values of model hyperparameters, for example, the number of neurons in the ANN model. At the end of this step, the coveted process model is obtained.

Additional activities related to computation of health indicator and subsequent RUL estimation involved in fault prognosis are covered in Part 7 of the book which deals specifically with prognostic techniques for predictive maintenance applications.

1.5 ML-based Plant Health Management Solution Deployment

After you have developed a satisfactory PHM model, the real test of your solution lies in how well it is received by the end-users. The end-users could be reliability personnel/engineers at the local plant sites or the central team of experts remotely supervising the plants. Figure 1.10 below shows a (simplified) common architecture for bringing your tool's results to these end-users. As shown, the ML model could be setup to run on local PCs at every site or a central server machine/cloud resource[5]. Plant operators may access the tool's results on the local control-room screens or via web browsers in case of centralized deployment. The web user interface could be either built using third-party visualization software (Tableau, Sisense, Power BI, etc.) or completely custom-built using front-end web frameworks like bootstrap.

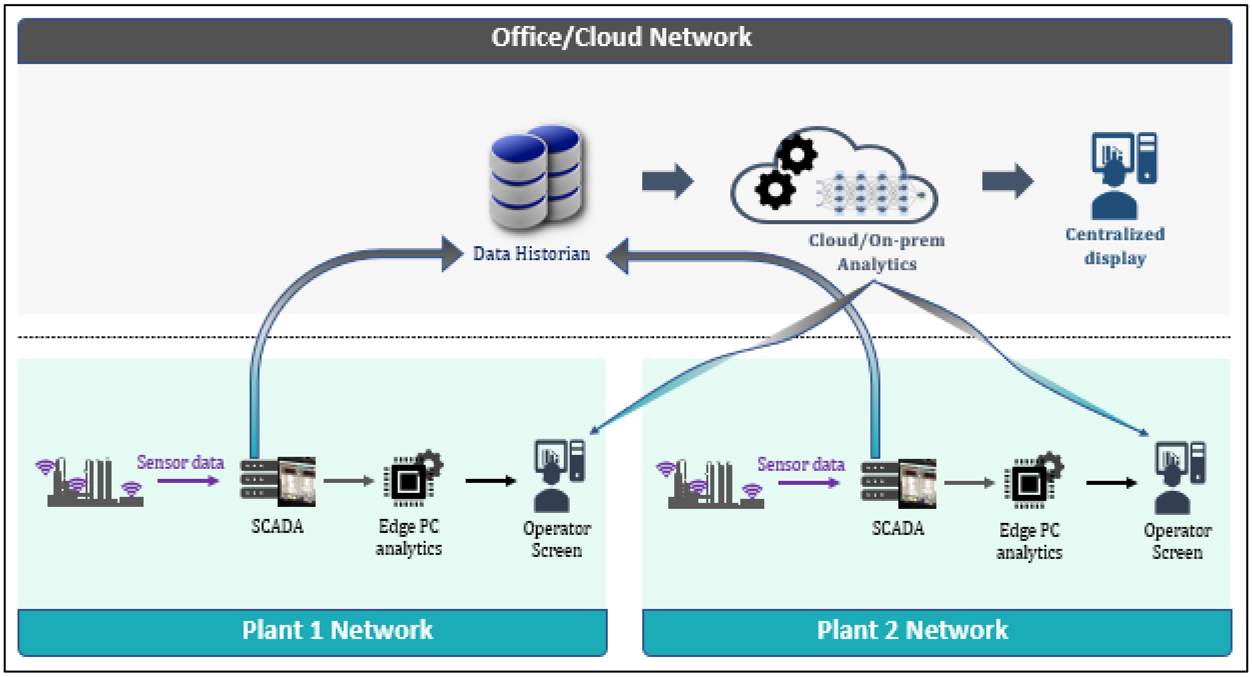


Figure 1.10: ML solution deployment

That is all it takes to deploy a ML-based PHM solution in a production environment. This concludes our quick attempt to impress upon you the importance of process monitoring and predictive maintenance in process industry. Hopefully, you also now have a good idea of what resources you have to achieve your PM/PdM goals and what is takes to build a PM/PdM solution.

[5] There exists a specialized branch of machine learning engineering, called MLOps (short for machine learning operations) that deals with reliable and scalable deployment of ML models in production.

Summary

In this chapter, we looked at the importance of plant health management for increasing process safety, reducing downtime costs, and increasing equipment life. We understood the meaning of process faults and abnormalities. We looked at the different stages of plant health management, familiarized ourselves with the factors that influence model selection, and looked at the different models available at our disposal to achieve the PHM goals. We also briefly looked at the generic workflow for process monitoring model development and understood how PM/PdM solutions are deployed in modern industrial settings. In the next chapter, we will take the first step and learn about the environment we will use to execute our Python scripts containing ML code for PHM.

Chapter 2

The Scripting Environment

In the previous chapter, we studied the various aspects of machine learning-based process monitoring and predictive maintenance. In this chapter, we will quickly familiarize ourselves with the Python language and the scripting environment that we will use to write ML codes, execute them, and see results. This chapter won't make you an expert in Python but will give you enough understanding of the language to get you started and help understand the several in-chapter code implementations in the upcoming chapters. If you already know the basics of Python, have a preferred code editor, and know the general structure of a typical ML script, then you can skip to Chapter 3.

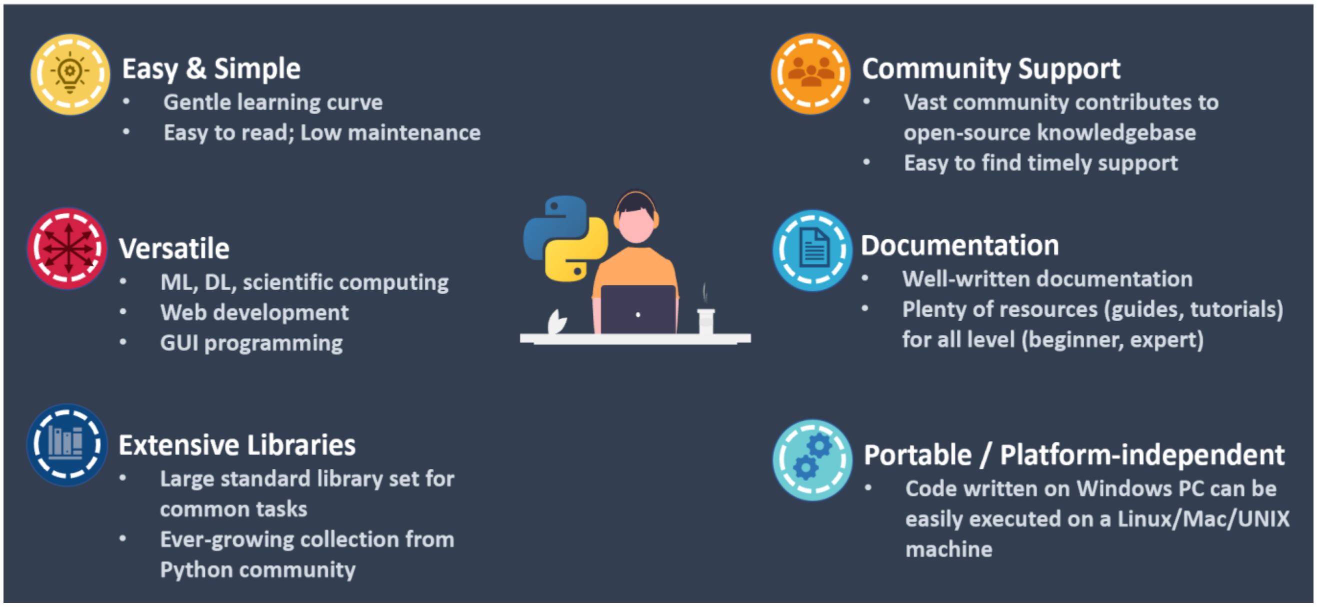
The ease of using and learning Python, along with the availability of a plethora of open-access useful packages developed by the user-community over the years, has led to immense popularity of Python. In recent years, development of specialized libraries for machine and deep learning has made Python the default language of choice among ML community. These advancements have greatly lowered the entry barrier into the world of machine learning for new users.

With this chapter, you are putting your first foot into the ML world. Specifically, the following topics are covered

- Introduction to Python language
- Introduction to Spyder and Jupyter, two popular code editors
- Overview of Python data structures and scientific computing libraries

2.1 Introduction to Python

In simple terms, Python is a high-level general-purpose computer programming language that can be used, amongst others, for application development and scientific computing. If you have used other computer languages like Visual Basic, C#, C++, Java, Javascript, then you would understand the fact that Python is an interpreted and dynamic language. If not, then think of Python as just another name in the list of computer programming languages. What is more important is that Python offers several features that sets it apart from the rest of the pack making it the most preferred language for machine learning. Figure 2.1 lists some of these features. Python provides all the tools to conveniently carry out all steps of an ML-based PM/PdM project, namely, data collection, data pre-processing, data exploration, ML modeling, visualization, and solution deployment to end-users. In addition, several freely available tools make writing Python code very easy[6].



Easy & Simple
- Gentle learning curve
- Easy to read; Low maintenance

Versatile
- ML, DL, scientific computing
- Web development
- GUI programming

Extensive Libraries
- Large standard library set for common tasks
- Ever-growing collection from Python community

Community Support
- Vast community contributes to open-source knowledgebase
- Easy to find timely support

Documentation
- Well-written documentation
- Plenty of resources (guides, tutorials) for all level (beginner, expert)

Portable / Platform-independent
- Code written on Windows PC can be easily executed on a Linux/Mac/UNIX machine

Figure 2.1: Features contributing to Python language's popularity

Installing Python

One can download official and the latest version of Python from the *python.com* website. However, the most convenient way to install and use Python is to install Anaconda (*www.anaconda.com*) which is an open-source distribution of Python. Along with the core Python, Anaconda installs a lot of other useful packages. Anaconda comes with a GUI called Anaconda Navigator (Figure 2.2) from where you can launch several other tools.

[6] Most of the content of this chapter is like that of Chapter 2 of the book 'Machine Learning in Python for Process Systems Engineering' and have been re-produced with appropriate changes to maintain standalone nature of this book.

Rest of the Chapter 2 not shown in this preview

Chapter 3

Exploratory Data Analysis: Getting to Know Your Data Better

Getting to know your enemy is a time-tested strategy for emerging victorious in any battle. For developing a satisfactory process monitoring model, this strategy translates to 'knowing your process data well'. This task is formally termed as exploratory data analysis (EDA). Most of the machine learning models make some assumptions regarding the distribution (e.g., Gaussian vs uniform distribution) and characteristics (e.g., dynamic vs steady state nature) of the data they operate upon. Therefore, it only serves us well investing some time in EDA so that the consistency between our chosen model's assumptions and the characteristics of process data at hand can be ascertained. Failure to do so will lead to high rate of false alerts and/or missed/delayed fault detection which will most likely lead to 'death' of your monitoring tool due to loss of user confidence!

In this chapter, we will learn how to assess the presence of four important properties in a dataset, viz, nonlinearity, non-Gaussianity, dynamics, and multimodality. We will motivate the study of these properties by understanding their impact on process monitoring performance. We will especially focus on techniques that render themselves convenient for implementation in an automated setting. As is obvious, the concepts learnt in this chapter will help you get better at correct model selection. Specifically, the following topics are covered

- Impact of non-ideal data properties on fault detection performance
- Techniques for assessing nonlinearity, non-Gaussianity, dynamic, and multimodality
- EDA of Tennessee Eastman Process dataset

3.1 Why Exploratory Data Analysis Matters?

Ask any expert process data scientist about some advice to get better at ML model selection and you will very likely get the suggestions to understand your data better. It's true, you cannot over-exaggerate the importance of gathering as many insights about the data as possible before getting your hands dirty. Most of the process monitoring methodologies make assumptions about the data characteristics and therefore, it is imperative to ascertain these characteristics in our process data to ensure selection of appropriate monitoring model. Let's consider the classical PCA model (inarguably the most popular model for monitoring multivariate industrial processes): the ideal dataset is linear, Gaussian distributed, single-clustered, and with no dynamics; Figure 3.1 uses simple datasets to illustrate what the deviations from these ideal characteristics look like.

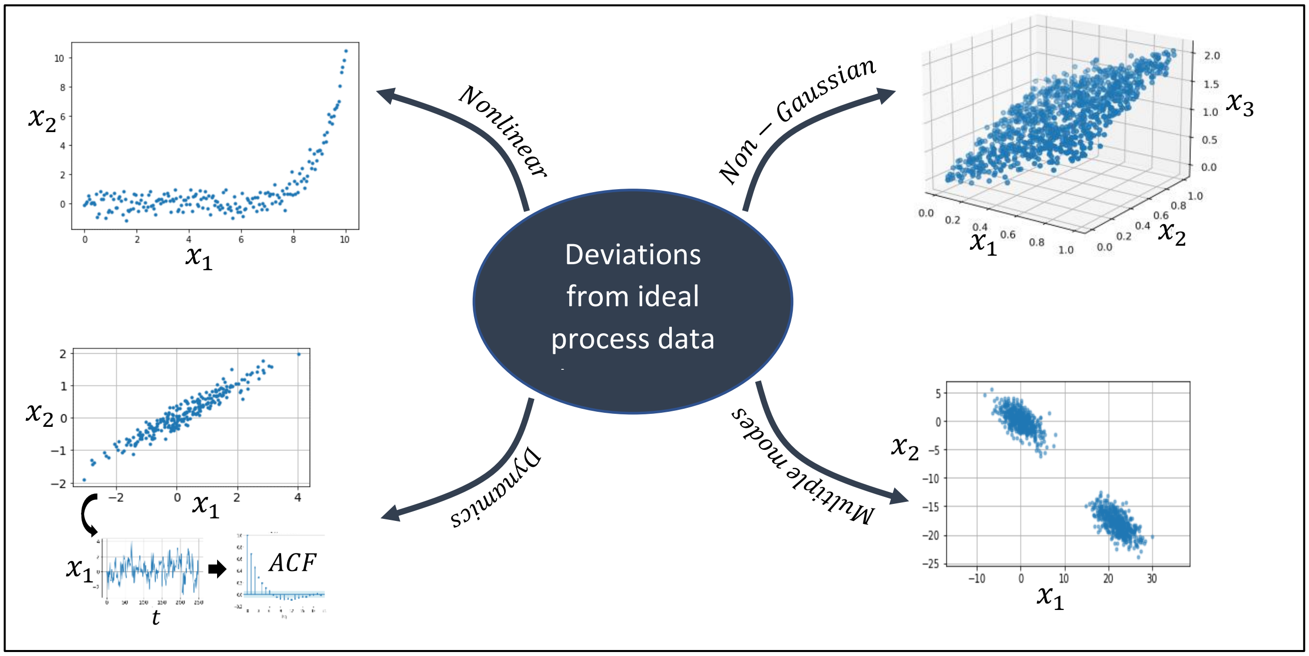


Figure 3.1: Illustration of deviations from ideal process data characteristics

To further motivate the discussions in the rest of the chapter, let's take a quick look at the impacts the non-ideal data characteristics can have on PCA performance.

Effect of nonlinearity

In an ideal PCA-compatible dataset, the variables are linearly related which allows the standard PCA method to find the lower-dimensional manifold along which the data is distributed. However, as can be seen below, PCA fails to transform the original 2D dataset to a 1D feature space even though it is apparent that the original data points lie along a curved manifold. This severely limits the ability of standard PCA to detect faulty samples.

Rest of the Chapter 3 not shown in this preview

Chapter 4

Machine Learning for Plant Health Management: Workflow and Best Practices

Whether you are building a ML solution for fault detection, fault classification, or fault prognosis, model development is the most critical task. Inarguably, obtaining a good ML model is not a trivial task. You cannot obtain a good model by just dumping all the available raw data in an off-the-shelf machine learning module. Incorrectly specify one hyperparameter and your model will return garbage results; provide insufficiently ‘rich’ training dataset and even the most carefully chosen ML model will prove incapable of providing meaningful insights. Unfortunately, an automated procedure for ML model development that works for all types of problems does not exist. Nonetheless, there is no cause for despair. The trick to successful model development lies in being actively involved in the several model development stages and making use of several useful guidelines that the ML community has come up with over the years. We already saw in the previous chapter the importance of acquiring a good understanding of data for correct model selection. In this chapter, we will learn several other guidelines and the best practices.

We will not cover the best practices associated with generic machine learning workflow. Concepts like feature extraction, feature engineering, cross-validation, regularization, etc., have already been covered in detail in our first book of the series. Albeit we will touch upon topics that are specific to plant health management applications. In this chapter, our focus will be on aspects that you should not ignore to ensure that you are not unknowingly setting your model up for failure. Specifically, we will cover these topics

- ML model development workflow
- Data selection and pre-processing to obtain good training dataset
- Assessment of monitoring performance
- Best practices for model selection and tuning

4.1 ML Model Development Workflow

The prime objective of the ML modeling task for building process modeling solutions is to obtain a model that provide high fault sensitivity (i.e., the model is able to detect process faults in incipient stages) and low false alarm rate (i.e., the model does not report a process fault when process is operating normally). Balancing the trade-off between these two requirements is not easy and requires careful attention to varied aspects during model development. It definitely takes more than just executing a ‘model = <some ML_model>.fit()’ command on the available data. In Chapter 1, we saw an overview of the typical steps involved in a ML model development exercise. In this chapter, we will look at the different components of the workflow in more details. Figure 4.1 lists the subtasks that we will touch upon. While separate books can be written on each of these subtasks, we will focus on the aspects that may get overlooked by an inexperienced process data scientist.

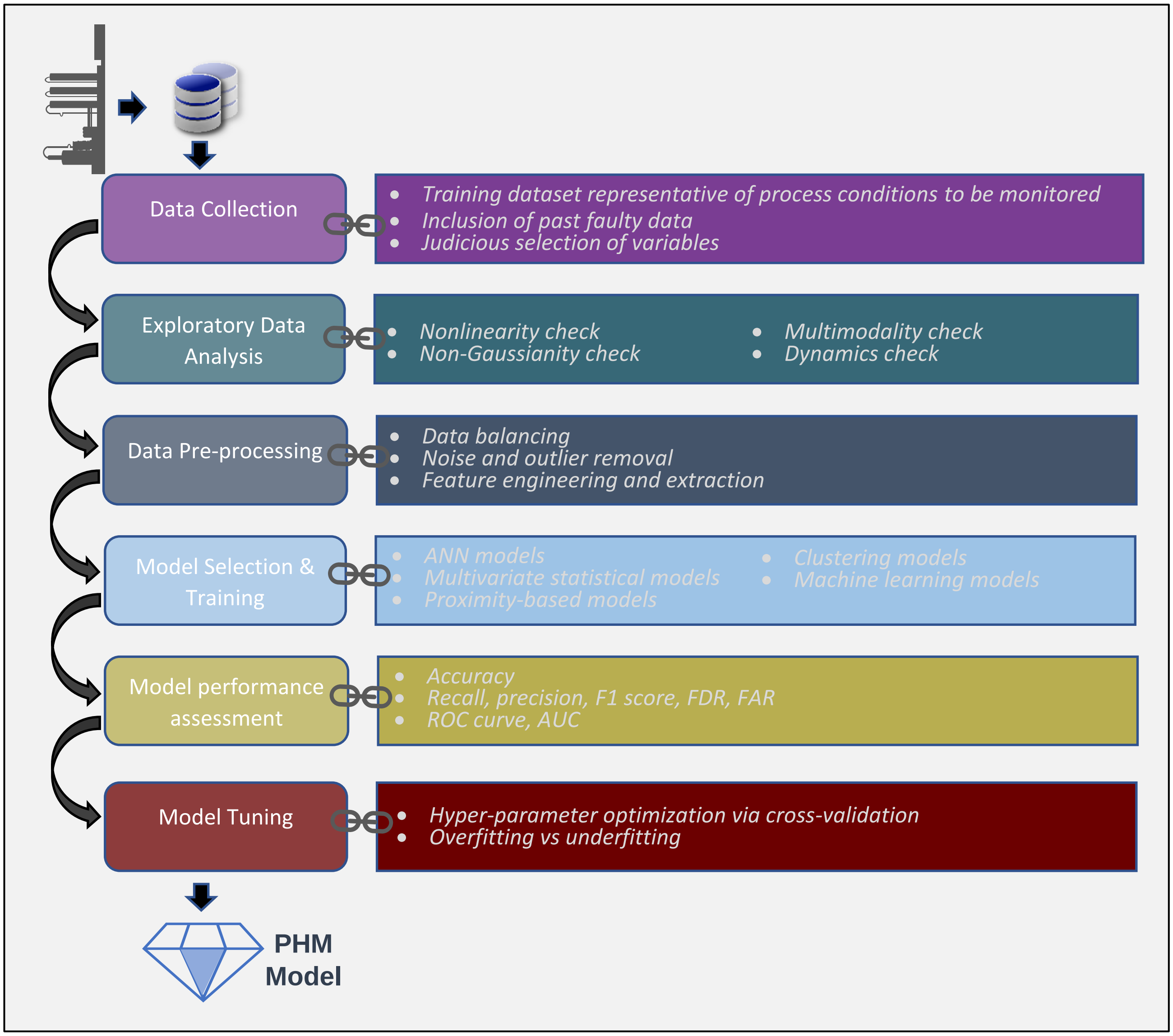

Data Collection
Training dataset representative of process conditions to be monitored
Inclusion of past faulty data
Judicious selection of variables
Exploratory Data Analysis
Nonlinearity check
Non-Gaussianity check
Multimodality check
Dynamics check
Data Pre-processing
Data balancing
Noise and outlier removal
Feature engineering and extraction
Model Selection & Training
ANN models
Multivariate statistical models
Proximity-based models
Clustering models
Machine learning models
Model performance assessment
Accuracy
Recall, precision, F1 score, FDR, FAR
ROC curve, AUC
Model Tuning
Hyper-parameter optimization via cross-validation
Overfitting vs underfitting
PHM Model

Figure 4.1: ML model development workflow

Rest of the Chapter 4 not shown in this preview

Part 2

Univariate Signal Monitoring

Chapter 5

Control Charts for Statistical Process Control

Before machine learning engulfed the process industry, simple plotting of key plant variables with statistically chosen upper and lower thresholds used to be the norm for detecting process abnormalities. These plots, called control charts, formed the major component of statistical process control or statistical quality control. Although control charts have lost some of their shine due to the advent of advanced multivariate process monitoring tools, they are still widely employed by plant management to monitor crucial KPIs, for example, product quality, process efficiency, etc. Simple concept, easy interpretation, and quick implementation are some of the reasons for their continued popularity.

Shewhart charts (which includes the popular 3-sigma charts) are the earliest, simplest, and most commonly used control charts. These, however, show poor performance for detection of faults that cause small deviations. Therefore, alternatives such as CUSUM charts and EWMA charts have been devised. In this chapter, we will learn these techniques and become familiar with how to implement them in practice. We will conclude with some discussion on the ways to overcome the shortcomings of univariate control charts. Specifically, the following topics are covered.

- Introduction to Shewhart control charts
- Introduction to CUSUM control charts
- Introduction to EWMA control charts
- Statistical process control of aeration tank via CUSUM chart
- Strategies for overcoming limitations of univariate statistical process control

5.1 Control Charts: Simple and Time-tested Process Monitoring Tools

Control charts are one of the seven[7] pillars of statistical process control that are traditionally used to monitor key production or product quality metrics in order to detect unexpected deviations. When the process is 'in-control', the monitored variables are expected to exhibit only natural cause variations around some target or mean values. As shown in Figure 5.1, a control chart is simply a display of measurements of a single process variable plotted against time or sample number. Additionally, these charts include a centerline or the expected mean value, and a couple of limit lines called UCL (upper control limit) and LCL (lower control limit). Most industrial process variables show natural variations due to random disturbances affecting the process. The control limits are statistically designed in such a way that under natural cause variations, the monitored variable remains within the control limits with certain desired probability. The breach of the control limits indicates potential process fault or an 'out-of-control' situation. Proper specification of the limits is therefore essential to ensure minimal false alarms and rapid detection of faults.

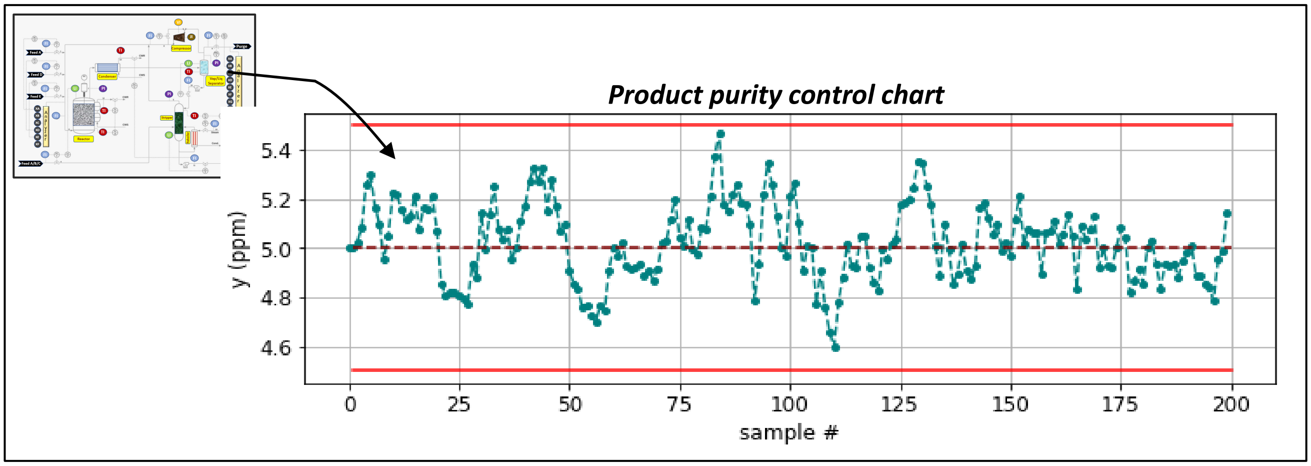


Figure 5.1: Representative control chart for product purity

The traditional control charts take three different forms, viz, Shewhart charts, CUSUM charts, and EWMA charts. While a Shewhart chart plots only the current measurements on the control chart, the other two plot some combination of current and past measurements. You will soon learn how usage of past measurements allow detection of incipient or low magnitude faults which may not get detected by Shewhart charts. Control charts are not limited to tracking process measurements only; any metric that is expected to exhibit only random fluctuations around some mean or target can be monitored via control charts. Correspondingly, control charts are also employed for monitoring model residuals, latent variables (for example, in PCA), etc. Let's first get started with Shewhart charts.

[7] https://asq.org/quality-resources/statistical-process-control

Rest of the Chapter 5 not shown in this preview

Chapter 6

Process Fault Detection via Time Series Pattern Matching

Imagine you are a plant operator newly put in charge of running a plant and you observe an interesting pattern in one of the process variable: occasional spikey fluctuations without the signal violating the DCS alarm limits. A natural line of investigation would be to find if such patterns have occurred in the past and are a leading indicators of underlying process faults. However, how do you quickly sift through years of historical data to find similar patterns? Consider another scenario where you are responsible for quality control of a batch process. To check if the latest batch went smoothly, you may want to compare it with known reference/golden batch. However, batches may show normal variations due to different batch durations or abnormal deviations due to process fault. How do you train an algorithm to smartly call out a faulty batch? One thing common in both these scenarios is that we are not looking at abnormality of a single measurement; instead, abnormality of a sequence of successive values (also called collective anomalies) is of interest.

Time series pattern matching is a mature field in the area of time series classification and recent algorithmic advances now allow very fast sequence comparisons to find similar or abnormal patterns in historical data. Unsurprisingly, pattern matching is being offered as prime feature in commercial process data analytic software (such as Aspen's Process Explorer, SEEQ, etc.). In this chapter, we will work through some use-cases of pattern-matching-based process monitoring. Specifically, the following topics are covered

- Introduction to time series anomalies
- Pattern matching-based fault detection: use-cases in process industry
- Fault detection via historical pattern search for steam generator process
- Fault detection via discord discovery

6.1 Time Series Anomalies and Pattern Matching

In anomaly detection literature, anomalies in univariate time series or dynamic signals are categorized into three categories: point anomalies, contextual anomalies, and collective anomalies. Figure 6.1 illustrates these anomalies for a valve (%) opening signal. As depicted in Figure 6.1b, if a single measurement deviates significantly from the rest of the sensor readings, then a point anomaly is said to have occurred. Contextual anomaly occurs when a measurement is not anomalous in an 'overall sense' but only in a specific context. For example, in Figure 6.1c, point '*B*' is abnormal when taken in the context of operation mode 1 only; valve opening goes close to 80% under normal operation but not when the process is in mode 1. The last category of collective anomaly occurs when a group/sequence of successive measurements jointly show abnormal behavior, although the individual measurements may not violate NOC range. While control charts can be built to detect point and contextual anomalies, more specialized approaches are needed to detect collective anomalies. Therefore, this chapter is devoted to study of approaches for collective anomaly detection.

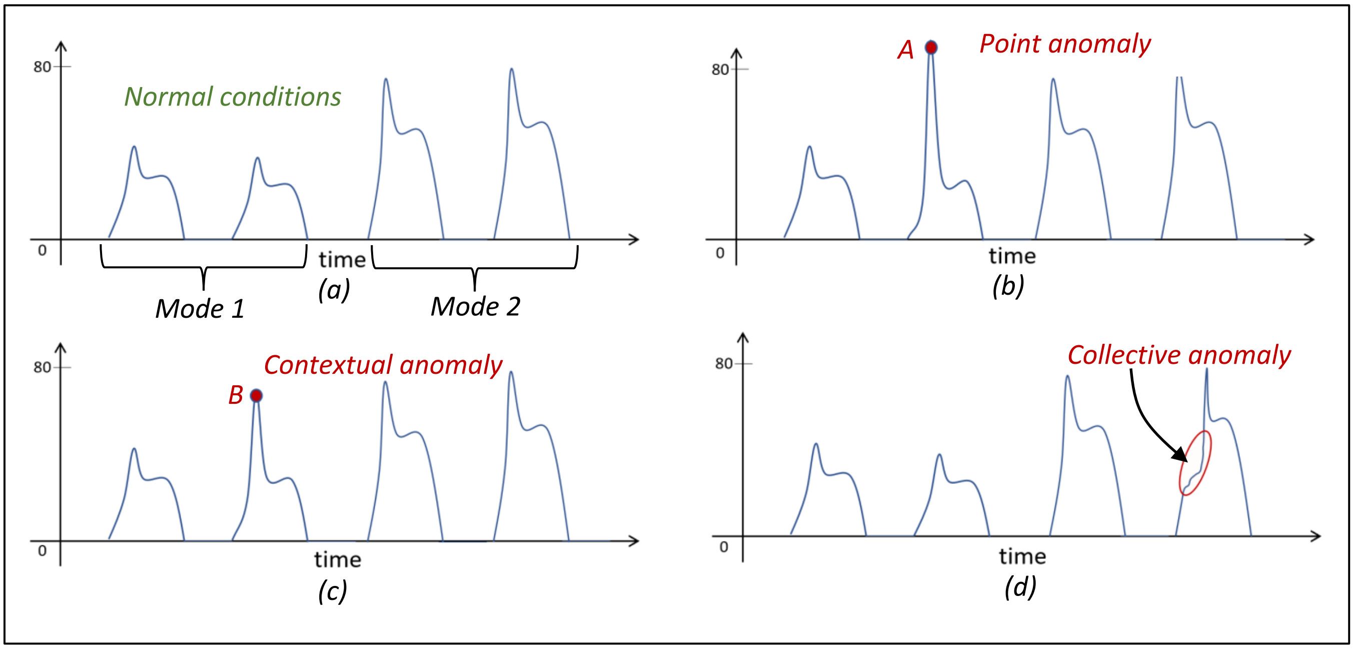


Figure 6.1: Time series anomalies: representative illustrations

The need for (sub) sequence-based pattern matching for FDD show up in different forms in process industry; Figure 6.2 illustrates some of the use-case scenarios. Let's work through some of these use-cases to understand the underlying techniques and available resources.

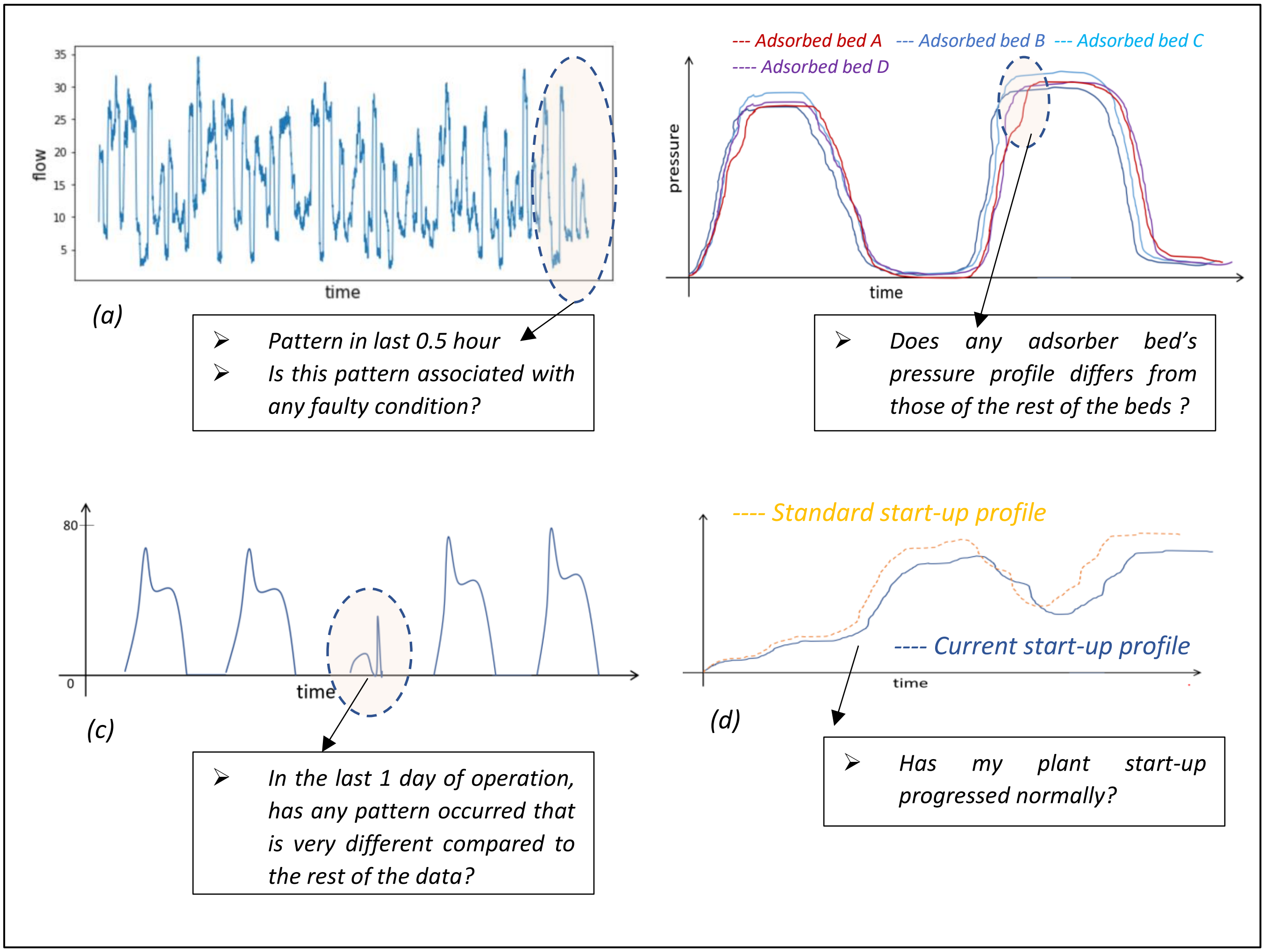


Figure 6.2: Sample use-case scenarios of time series pattern matching-based fault detection

We will work through the case scenarios *(a)* and *(c)*. Both of these use-cases involve finding similarity of a 'query' subsequence with several other subsequences taken from the same time series or another time series. In order to accomplish this in a time-efficient manner, a library called STUMPY[8] will be utilized. Let's learn how to utilize STUMPY for our time series data mining tasks.

[8] https://stumpy.readthedocs.io/en/latest/index.html.
S.M. Law, STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. Journal of Open Source Software, 2019.

Rest of the Chapter 6 not shown in this preview

Part 3

Multivariate Statistical Process Monitoring

Chapter 7

Multivariate Statistical Process Monitoring for Linear and Steady-State Processes: Part 1

It is not uncommon to have hundreds of process relevant variables being measured at manufacturing facilities. However, conservation laws such as mass balances, thermodynamics constraints, enforced product specifications, and other operational restrictions induce correlations among the process variables and make it appear as if the measured variables are all derived from a small number of hidden (un-measured) variables. Several smart techniques have been derived to find these hidden latent variables. Latent variable-based techniques allow characterization of 'normal' process noise affecting the process during NOC. Process monitoring methods based on latent space monitor the values of latent variables and process noise in real-time to infer the presence of process faults. Sounds complicated? Don't worry! This chapter will show you how this is accomplished while retaining focus on conceptual understanding and practical implementation.

PCA and PLS are among the most popular latent variable-based process monitoring tools and have been reported in several successful industrial process monitoring applications. This chapter provides a comprehensive exposition of the PCA and PLS techniques and teaches you how to apply them for fault detection. Furthermore, we will learn how to identify the faulty process variable using the popular contribution analysis methodology. Specifically, the following topics are covered

- Introduction to PCA and PLS
- Process fault detection via PCA and PLS
- Fault isolation in PCA- and PLS-based process monitoring applications
- Process monitoring of polymer manufacturing process via PCA
- Process monitoring of polyethylene manufacturing process via PLS

7.1 PCA: An Introduction

Principal component analysis (PCA), in essence, is a multivariate technique that transforms a high-dimensional set of correlated variables into a low-dimensional set of uncorrelated (latent) variables with minimum loss of information. Consider the 3-dimensional data in Figure 7.1. It is apparent that although the data is three dimensional, the data-points mostly lie along a 2-D plane; and even in this plane, the spread is much higher along a particular direction. PCA converts the original *(x,y,z)* space into a 2-D principal component (PC) space where the 1st PC (PC1) corresponds to the direction of maximum spread/variance in data and the 2nd PC (PC2) corresponds to the direction with highest variance among all directions orthogonal to 1st PC. Depending upon modeling requirements, even the 2nd PC may be discarded, essentially obtaining a 1-D data while losing out some information. Also, as we will see soon, it is straightforward to recover original data from data in PC space.

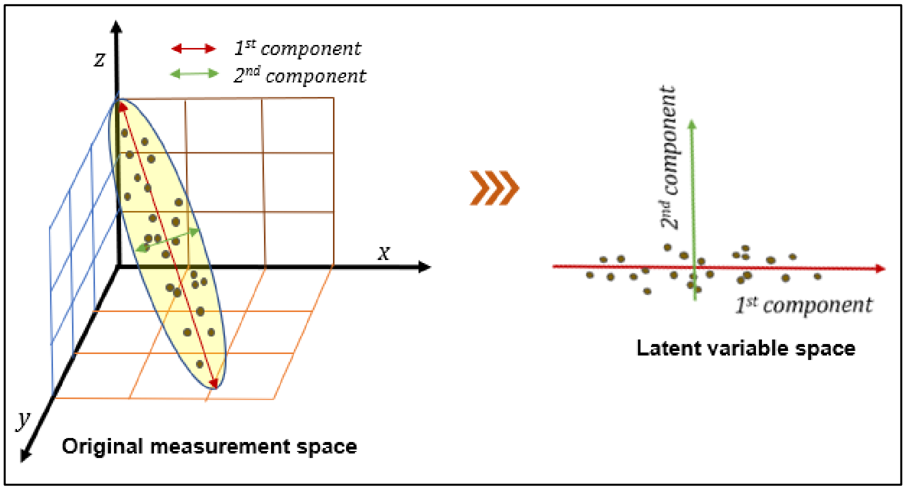


Figure 7.1: PCA illustration

In ML world, it is common to find applications of classification and clustering techniques in the PC space. In process industry, process modeling (via principal component regression (PCR)) and monitoring are common application of PCA[9]. PCA is also frequently utilized for process visualization. For many applications, two or three PCs are adequate for capturing most of the variability in process data and therefore, the compressed process data can be visualized within a single plot. Plant operators and engineers use this single plot to find past and current patterns in process data. PCA-based fault detection goes further and compresses all the information in the PC space and the process noise into a couple of control charts. You will soon learn how to generate and use these control charts.

[9] The popularity of latent-variable techniques for process control and monitoring arose from the pioneering work by John McGregor at McMaster University.

Rest of the Chapter 7 not shown in this preview

Chapter 8

Multivariate Statistical Process Monitoring for Linear and Steady-State Processes: Part 2

By now you must be very impressed with the powerful capabilities of PCA and PLS techniques. These methods allowed us to extract latent variables and monitor systematic variations in latent space and process noise separately. However, you may ask, "Are these the best latent variable-based techniques to use for all problems?". We are glad that you asked! Other powerful methods do exist which may be better suited in certain scenarios. For example, independent component analysis (ICA) is preferable over PCA when process data is not Gaussian distributed. It can provide latent variables with stricter property of statistical independence rather than only uncorrelatedness. Independent components may be able to characterize the process data better than principal components and thus may result in better monitoring performance.

In another scenario, if your end goal is to classify process faults into different categories for fault diagnosis, then, maximal separation between data from different classes of faults would be your primary concern rather than maximal capture of data variance. Fisher discriminant analysis (FDA) is preferred for such tasks.

In this chapter, we will learn in detail the properties of ICA and FDA. We will apply these methods for process monitoring and fault classification for a large-scale chemical plant. Specifically, the following topics are covered

- Introduction to ICA
- Process monitoring of non-Gaussian processes
- Introduction to FDA
- Fault classification for large scale processes.

8.1 ICA: An Introduction

Independent Component Analysis (ICA) is a multivariate technique for transforming measured variables into statistically independent latent variables in a lower-dimensional space. Statistical independence is a stricter condition than uncorrelatedness and in some situations, working with independent components (ICs) can give better results than working with uncorrelated PCs from PCA. While ICA and PCA are related (in the sense that latent variables are linear projections of measured variables), they differ in the way the latent variables are extracted. Figure 8.1 highlights the difference between them using a simple illustration where two independent signals are linearly combined to generate correlated signals and then PCA/ICA are used to extract latent signals. It is apparent that simply decorrelating the signals via PCA did not recover the original signals. On the other hand, ICA reconstructs the original signals accurately[10].

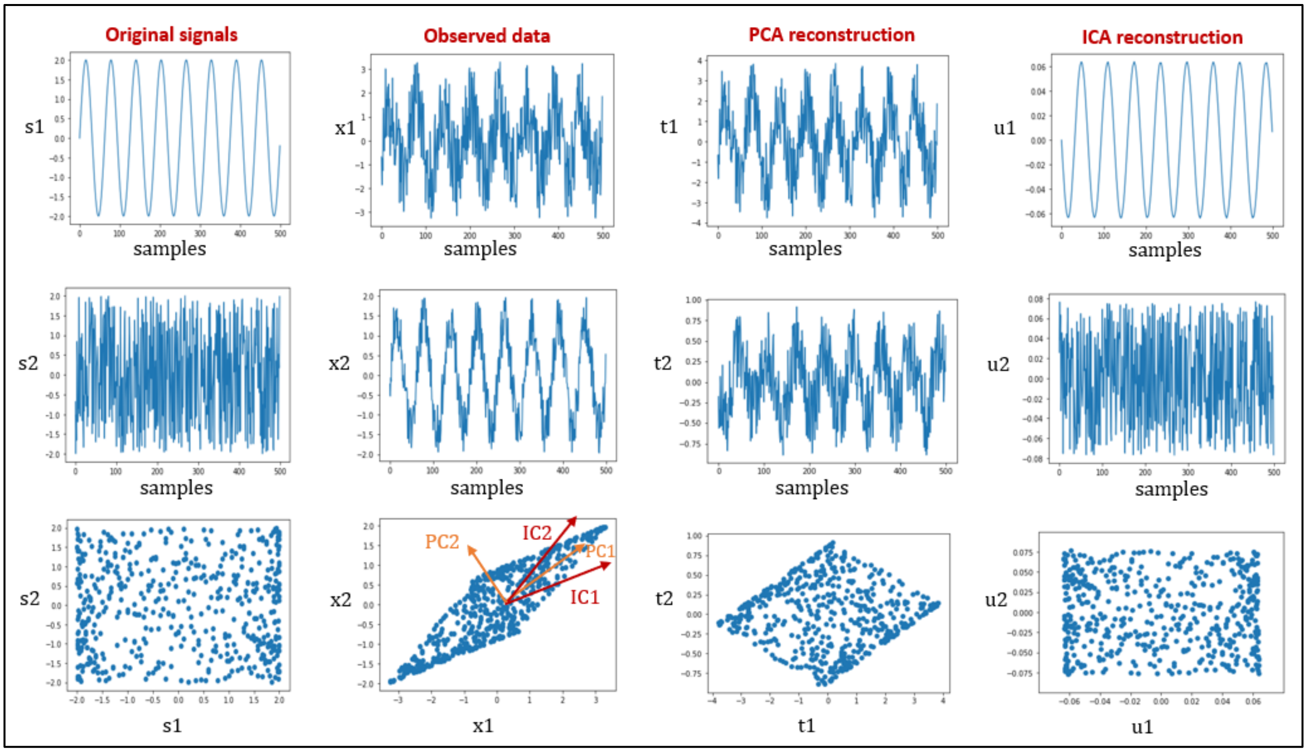


Figure 8.1: Simple illustration of ICA vs PCA. The arrows in the x1 vs x2 plot show the direction vectors of corresponding components. Note that the signals t1 and t2 are not independent as value of one variable influences the range of values of the other variable.

ICA uses higher-order statistics for latent variable extractions, instead of only second order statistics (mean, variance/covariance) as done by PCA. Therefore, for non-Gaussian

[10] If you observe closely, you will find that ICA latent signals (u1 and u2) do differ from s1 and s2 signals in terms of sign and magnitude; we will soon learn why this happens and why this is not a cause of worry.

Rest of the Chapter 8 not shown in this preview

Chapter 9

Multivariate Statistical Process Monitoring for Linear and Dynamic Processes

In the previous chapters, we saw how beautifully latent variable-based MSPM techniques can extract hidden steady-state relationships from data. However, we imposed a major restriction of absence of dynamics in the dataset. Unfortunately, it is common to have to deal with industrial datasets that exhibit significant dynamics and the standard MSPM techniques fail in extracting dynamic relationships among process variables. Nonetheless, the MSPM research community came up with a simple but ingenious modification to the standard MSPM techniques that made working with dynamic dataset very easy. The trick entails including the past measurements as additional process variables. That's it! The standard techniques can then be employed on the augmented dataset. The dynamic variants of the standard MSPM techniques are dynamic PCA (DPCA), dynamic PLS (DPLS), dynamic ICA (DICA), etc.

Dynamic PCA and dynamic PLS are among the most popular techniques for monitoring linear and dynamic processes; accordingly, these are covered in detail in this chapter. Additionally, this chapter also introduces another very popular and powerful technique that is specially designed to extract dynamic relationships from process data – canonical variate analysis (CVA). Using numerical and industrial-scale case studies, we will see how to use these three techniques to build fault detection tools. Specifically, the following topics are covered

- Introduction to dynamic PCA
- Fault detection using DPCA
- Introduction to dynamic PLS
- Introduction to CVA
- Fault detection using CVA for Tennessee Eastman process

9.1 Dynamic PCA: An Introduction

Dynamic PCA is the dynamic extension of conventional PCA designed to handle process data that exhibit significant dynamics. DPCA simply entails application of conventional PCA to augmented data matrix which, as shown in Figure 9.1, is generated by using past measurements as additional process variables. Note that each 'variable' of the augmented matrix is normalized to zero mean and unit variance as is done in conventional PCA. It may seem surprising, but such a simple approach has achieved great success in process industry and has been readily adopted due to ease of implementation.

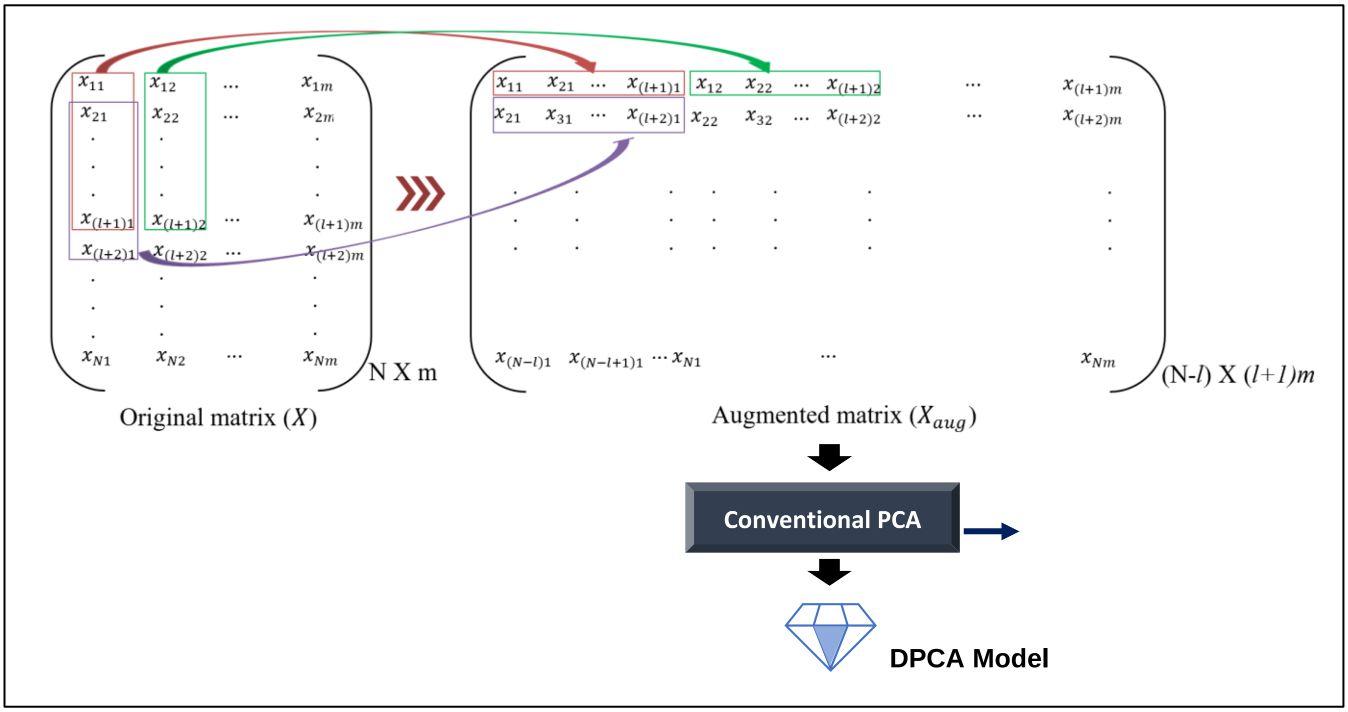


Figure 9.1: Dynamic PCA procedure [l denotes the number of lags used]

All the mathematical expressions for the computations of the score matrix[11], residual matrix, Hotelling's T^2, and SPE remain the same (Eq. 1 to Eq. 8) as that shown in Chapter 7, except that now you will be using scaled X_{aug} instead of X, i.e., $T = X_{aug}P$; $E = X_{aug} - \hat{X}_{aug}$. The procedure for determination of number of retained latent variables also remains the same. You may, amongst other approaches, look for a 'knee' in the scree plot of the explained variance or use the cumulative percent variance approach. If you choose 'l' correctly, then both the static and dynamic relationships among process variables are captured and correspondingly, the residuals and the Q statistic will not exhibit autocorrelations[12]. For test dataset, you would again simply perform augmentation with time-lagged measurements. Before we get into the nitty-gritties, let's see a quick motivating example on why DPCA is superior to PCA in the presence of dynamics.

[11] The number of retained principal components in DPCA could be greater than m.

[12] The DPCA scores can show autocorrelations.

Example 9.1:

To illustrate how DPCA can extract dynamic relationships, let's consider the following noise-free dynamic system.

$$x_1(k) = 0.8x_1(k-1) + x_2(k-1); \quad k \text{ is sampling instant}$$

The number of zero singular (eigen) values extracted during PCA indicates the number of linear relationships that exist among the process variables. Let's see if we can extract out the above dynamic relationship using only data (1000 samples of x_1 and x_2).

```
# import required packages
import numpy as np, matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# generate data for the system: x1(k) = 0.8*x1(k-1) + x2(k-1)
x2 = np.random.normal(loc=0, scale=1, size=(1000,1))
x1 = np.zeros((1000,1))
for k in range(1,1000):
   x1[k] = 0.8*x1[k-1] + x2[k-1]
X = np.hstack((x1, x2))

# function to generate augmented matrix
def augment(X, n_lags):
        N, m = X.shape
        X_aug = np.zeros((N-n_lags, (n_lags+1)*m))
        for sample in range(n_lags, N):
                XBlock = X[sample-n_lags:sample+1,:]
                X_aug[sample-n_lags,:] = np.reshape(XBlock, (1,-1), order = 'F')
        return X_aug

# fit DPCA model
X_aug = augment(X, 1) # augment data
X_aug_centered = X_aug - np.mean(X_aug, axis=0) # center data
dpca = PCA().fit(X_aug_centered) # fit PCA model
print('DPCA singular values:', dpca.singular_values_) # get singular values

>>> DPCA singular values: [6.664e+01 3.731e+01 3.094e+01 1.1661e-14]
```

As expected, only one singular value is very close to zero. All we now need to do is fetch the singular vector corresponding to this singular value and check if it represents our dynamic system.

```
# get 4th singular vector
print('4th singular vector: ', dpca.components_[3,:])

>>> 4th singular vector:  [ 4.923e-01 -6.154e-01  6.154e-01  1.7348e-17]
```

The 4th singular vector represents the following relation

$$0.4923x_1(k-1) - 0.6154\,x_1(k) + 0.6154x_2(k) = 0$$
$$\Rightarrow \boxed{x_1(k) = 0.8x_1(k-1) + x_2(k-1)}$$

Voila! DPCA has successfully extracted the underlying process dynamics. PCA on the other hand does not reveal any relationship between the variables.

Rest of the Chapter 9 not shown in this preview

Chapter 10

Multivariate Statistical Process Monitoring for Nonlinear Processes

In the previous chapters, we saw how a simple trick of using time-lagged variables enabled application of conventional MSPM techniques to dynamic processes. You may wonder if anything similar exists for nonlinear processes. Fortunately, it does! The underlying principle is to project the original variables onto a high-dimensional feature space where features are linearly related. The challenging part is the determination of the nonlinear mapping from the original measurement space to the feature space This is where a 'kernel' trick comes into picture wherein data gets projected without the need to explicitly define the nonlinear mapping. Conventional MSPM is then employed in the feature space. Sounds complicated? Once you are done with this chapter, you will realize that it's much easier than it may seem to you right now.

The main advantage of kernel-based MSPM techniques (KPCA, KPLS, KICA, KFDA, etc.) is that they do not require nonlinear optimization and only linear algebra is involved. Unsurprisingly, kernelized methods have become very attractive for dealing with nonlinear datasets while retaining the simplicity of their linear counterparts. Among the kernel MSPM techniques, kernel PCA and kernel PLS are the most widely adopted, have found considerable successes in process monitoring applications, and therefore will be the focus of our study in this chapter. Specifically, the following topics are covered

- Introduction to kernel PCA
- Fault detection using kernel PCA
- Introduction to kernel PLS
- Fault detection using kernel PLS

10.1 Kernel PCA: An Introduction

Kernel PCA is the nonlinear extension of conventional PCA suitable for handling processes that exhibit significant nonlinearity. To understand the motivation behind KPCA, consider the simple scenarios illustrated in Figure 10.1. In Figure 10.1a, we can see that the data lie along a line which can be obtained from the first eigenvector of linear PCA. In Figure 10.1b, data lie along a curve; conventional PCA cannot help to find this nonlinear curve. Correspondingly, PCA fails to detect the obvious outlier. However, all is not lost for the latter scenario. Instead of working in the (x_1, x_2) measurement space, if we work in the $(z_1, z_2) = (x_1^4,\ x_2)$ feature space, then we end up with linearly related features and the abnormal data point can be flagged as such. Unfortunately, the task of finding such (nonlinear) mapping that maps raw data to feature variables is not trivial. Thankfully, there is something called 'kernel trick' that allows you to work in the feature space without having to define the nonlinear mapping. We will learn how this is accomplished in the next section.

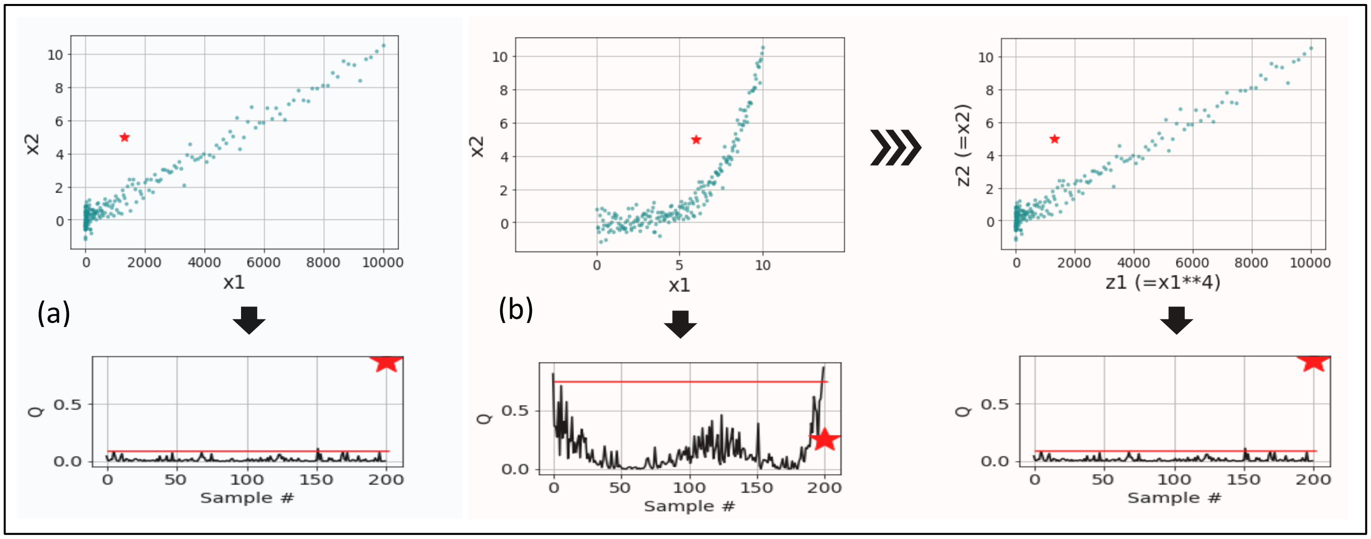
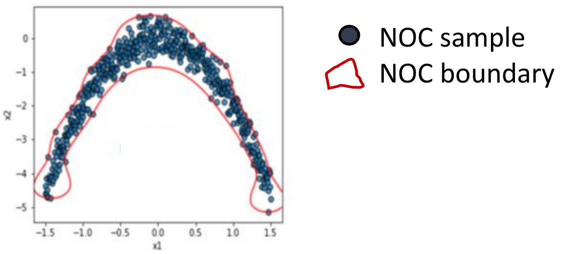


Figure 10.1: Nonlinearity impact on PCA-fault detection. Faulty sample shown in red. [One principal component chosen in all simulations]

KPCA can work with arbitrary data distributions. As far as process monitoring applications are concerned, KPCA can help you create an abnormality boundary around your NOC data as shown below.



To understand how KPCA works, let's revisit the mathematical underpinnings of PCA.

Kernel functions and kernel trick

Usage of kernel trick is not limited to KPCA and KPLS. Other ML techniques such as SVM, CVA, etc., also use kernel functions to model nonlinear processes. So, what are these kernel functions? Let's try to understand them.

We alluded to before that a popular approach to handling nonlinearity is to map observation sample *x* to x_F in feature space where conventional linear ML technique can be applied.

$$\varphi(x): x \rightarrow x_F$$

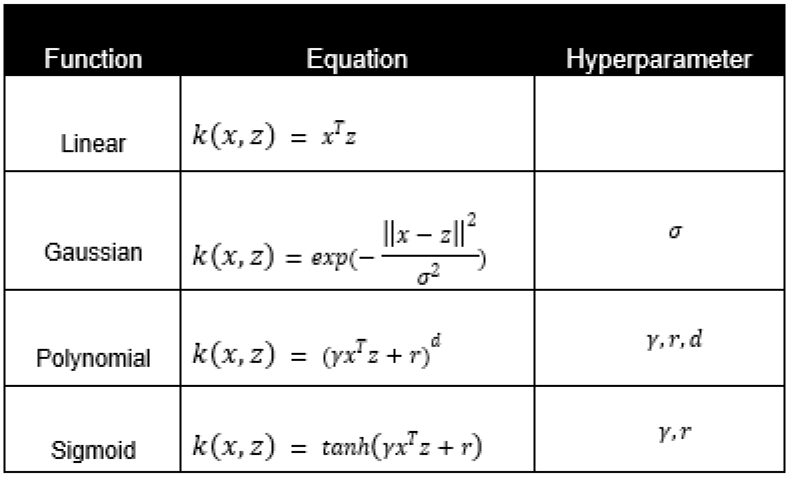
However, the map $\varphi(.)$ is unknown. Thankfully, in the mathematical formulation of many ML algorithms, the inner (or dot) product of feature vectors, $\varphi(x)^T\varphi(x)$, is frequently encountered. This inner product is denoted as

$$k(x_i,\ x_j) = <\varphi(x_i),\ \varphi(x_j)> = \ \varphi(x_i)^T\varphi(x_j)$$

where *k(.,.)* is called the kernel function. Several forms of *k(.,.)* are available and the most common form is Gaussian or radial basis function defined as

$$k(x_i,\ x_j) = \ exp\left[-\frac{(x_i - x_j)^T(x_i - x_j)}{\sigma^2}\right]$$

where, δ (a hyperparameter), is called kernel width. Usage of kernel functions allow application of linear ML techniques in feature space without explicitly knowing the feature vectors and this trick is called the '*kernel trick*'. Another term you will encounter in kernelized algorithms is kernel matrix (often denoted as *K*). The *(i, j)*th element of *K* is simply $k(x_i,\ x_j)$. The table below lists the commonly used kernel functions



Function	Equation	Hyperparameter
Linear	$k(x,z) = x^Tz$	
Gaussian	$k(x,z) = exp(-\frac{\lVert x-z\rVert^2}{\sigma^2})$	σ
Polynomial	$k(x,z) = (\gamma x^Tz + r)^d$	γ, r, d
Sigmoid	$k(x,z) = tanh(\gamma x^Tz + r)$	γ, r

Let’s use the polynomial kernel to illustrate how using kernel functions amounts to higher dimensional mapping. Assume that we use the following kernel

$$k(x,z) = (x^T z + 1)^2$$

where $x = [x_1, x_2]^T$ and $z = [z_1, z_2]^T$ are two vectors in the original 2D space. We claim that the above kernel is equivalent to the following mapping

$$\varphi(x) = [x_1, x_2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1]$$

To see how, just compute $\varphi(x)^T\varphi(z)$

$$\begin{aligned}\varphi(x)^T\varphi(z) &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1z_1 + 2x_2z_2 + 2x_1x_2z_1z_2 + 1\\ &= (x_1z_1 + x_2z_2 + 1)^2\\ &= (x^Tz + 1)^2\\ &= k(x,z)\end{aligned}$$

Therefore, if you use the above polynomial kernel, you are implicitly projecting your data onto a 6th dimensional feature space! If you were amazed by this illustration, you will find it more interesting to know that Gaussian kernels map original space into an infinite dimensional feature space! Luckily, we don’t need to know the form of this feature space.

Rest of the Chapter 10 not shown in this preview

Chapter 11

Process Monitoring of Multi-Mode Processes

In the previous chapters, we witnessed the benefits of customizing the conventional MSPM techniques for non-Gaussian, dynamic, and nonlinear processes. In this chapter, we will remove the last remaining restriction of unimodal operation. In your career, you will frequently encounter industrial datasets that exhibit multiple operating modes due to variations in production levels, feedstock compositions, ambient temperature, product grades, etc., and data-points from different modes tend to group into different clusters. The mean and covariance of process variables may be different under different operation models and therefore, when you are building a monitoring tool, judicious incorporation of the knowledge of these data clusters into process models will lead to better performance and, alternatively, failure to do so will often lead to unsatisfactory monitoring performance.

In absence of specific process knowledge or when the number of variables is large, it is not trivial to find the number of clusters or to characterize the clusters. Fortunately, several methodologies are available which you can choose from for your specific solution. In this chapter, we will learn different ways of working with multimodal data, some of the popular clustering algorithms, and understand their strengths and weaknesses. We will conclude by building a monitoring tool for a multimode semiconductor process. Specifically, the following topics are covered

- Different methodologies for modeling multimodal process data
- Introduction to clustering
- Finding groups using classical k-means clustering
- Probabilistic clustering via Gaussian mixture modeling
- Process monitoring of multimode semiconductor manufacturing operation

11.1 Need and Methods for Specialized Handling of Multimode Processes

In process systems, multimode operations occur naturally due to varied reasons. For example, in a power generation plant, production level changes according to the demand leading to significantly different values of plant variables with potentially different inter-variable correlations at different production levels. The multimode nature of data distribution causes problems with traditional ML techniques. To understand this, consider the illustrations in Figure 11.1. In subfigure (a), data indicates 2 distinct modes of operation. From process monitoring perspective, it would make sense to draw separate monitoring boundaries around the two clusters; doing so would clearly identify the red-colored data-point as an outlier or a fault. The Conventional PCA-based monitoring, on the other hand, would fail to identify the outlier. In subfigure (b), the correlation between the variables is different in the two clusters. It would make sense to build separate models for the two clusters to capture the different correlation structure. The Conventional PLS model would give inaccurate results.

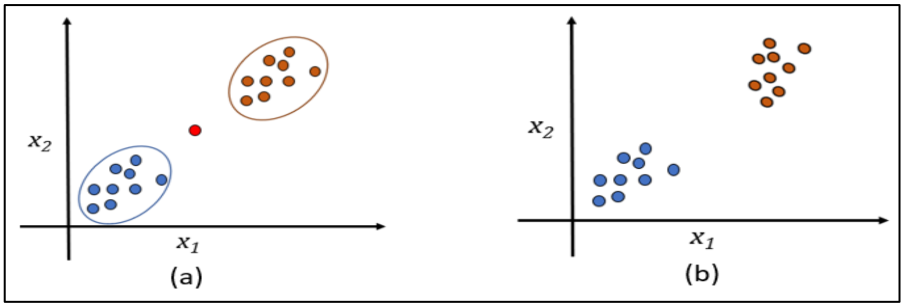


Figure 11.1: Illustrative scenarios for which conventional ML techniques are ill-suited

A few different methodologies have been adopted by the PSE community to monitor multimode process operations. Let’s familiarize ourselves quickly with these.

Multiple model approach

Here, separate models are built for each of the clusters corresponding to the different operation modes as shown in the figure below. Once the clusters have been characterized in the training data and cluster-wise models have been built, prediction for a new sample can be obtained by either only considering the cluster-model most suitable for the new sample or combining the predictions from all the models as shown in Figure 11.2. The decision fusion module can also take various forms. For example, for a process monitoring application, a simple fusion strategy could be to consider a new sample as a normal sample if atleast one of the cluster-models predict so. A different strategy could be to combine the abnormality metrics from all the models and make prediction based on this fused metric. Similarly, for soft sensing application, response variable prediction from individual models can be weighted and combined to provide final prediction.

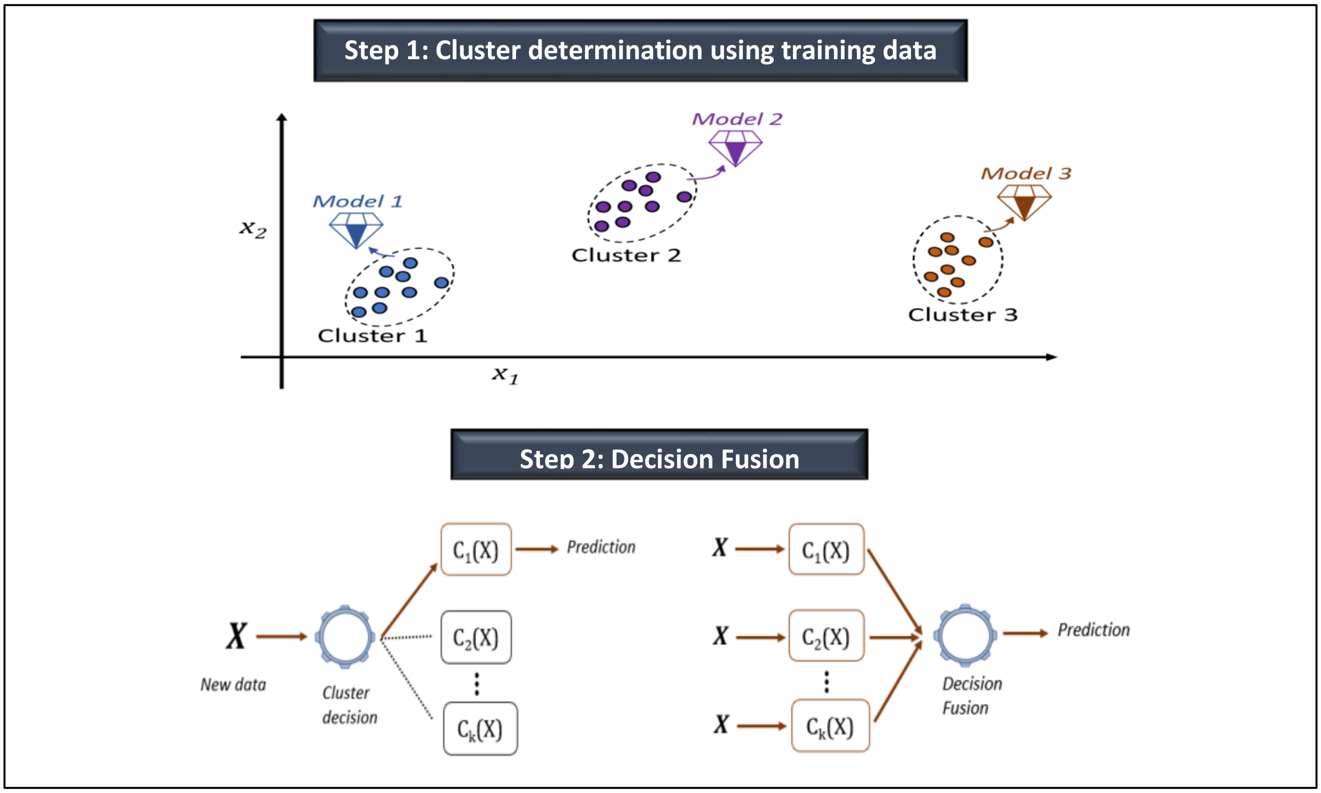


Figure 11.2: Multiple model approach for multimode processes

Lazy or just-in-time learning

In this approach, the model building exercise is carried out online. When new process data come in, relevant data are fetched from the historical dataset that are similar to the incoming samples based on some nearest neighborhood criterion. A local model is built using the fetched relevant data. The obtained model processes the incoming samples and is then discarded. A new local model is built when the next samples come in.

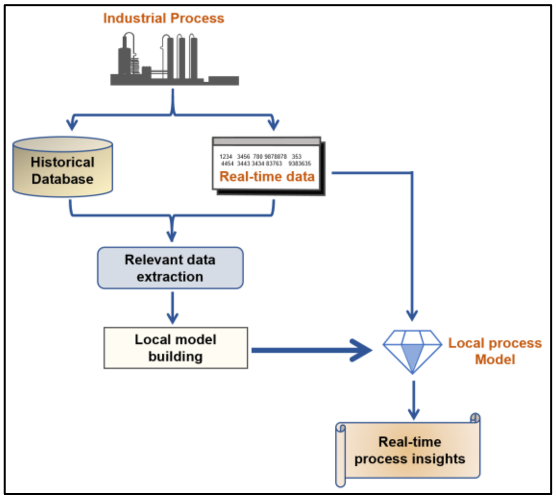


Figure 11.3: Steps involved in a just-in-time learning methodology

External analysis

In this strategy, the influence of process variables (called external variables) such as product grade, feed flow, etc., that lead to multimode operation is removed from the other 'main' process variables and then the conventional MSPM techniques are employed on the ensuing residuals as shown in the figure below.

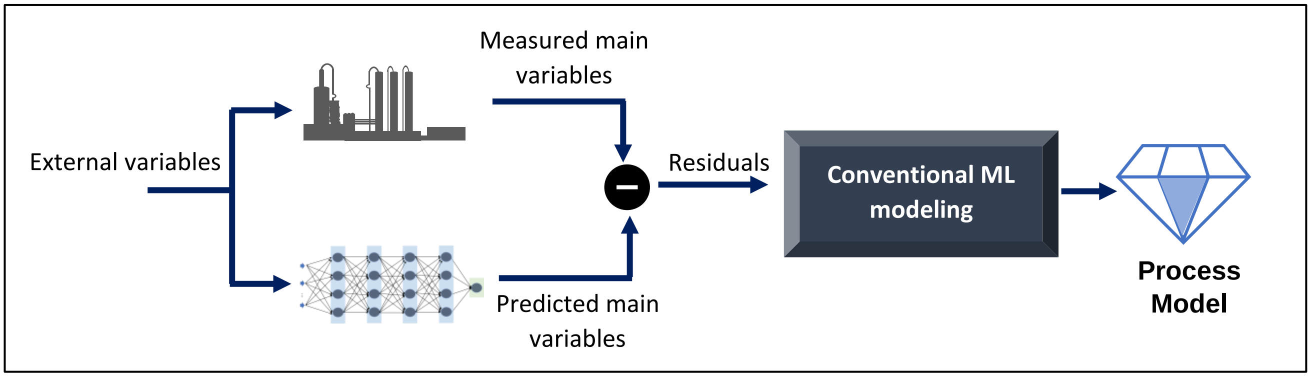


Figure 11.4: External analysis approach for multimode data

Rest of the Chapter 11 not shown in this preview

Part 4

Classical Machine Learning Methods for Process Monitoring

Chapter 12

Support Vector Machines for Fault Detection

In the previous chapters, we focused on multivariate statistical process monitoring methods that modelled process data through extraction of latent variables. In this part of the book, we will cover several classical ML techniques that come in handy in building process monitoring applications. These techniques do not attempt to build any statistical model of the underlying data distribution. Rather, the measurement space itself may be segregated into favorable/unfavorable regions, high-density/low-density regions, or pair-wise distances maybe computed to generate monitoring metrics, etc. SVM (support vector machine) is one such algorithm which excels in dealing with high-dimensional, nonlinear, and small or medium-sized data.

SVMs are extremely versatile and can be employed for classification and regression tasks in both supervised and unsupervised settings. SVMs, by design, minimize overfitting to provide excellent generalization performance. Infact, before ANNs became the craze in ML community, SVMs were the toast of the town. Even today, SVM is a must-have tool in every ML practitioner's toolkit. You will find more about SVMs as you work through this chapter. In terms of uses in process industry, SVMs have been employed for fault classification, fault detection, outlier detection, soft sensing, etc. We will focus on process monitoring-related usage in this chapter.

To understand different aspects of SVMs, we will cover the following topics

- Fundamentals of SVMs
- Kernel SVMs
- SVDD (support vector data description) for unsupervised classification
- Fault detection via SVDD for semiconductor manufacturing process

12.1 SVMs: An Introduction

The classical SVM is a supervised linear technique for solving binary classification problems. For illustration, consider Figure 12.1a. Here, in a 2D system, the training data-points belong to two distinct (positive and negative) classes. The task is to find a line/linear boundary that separates these 2 classes. Two sample candidate lines are also shown. While these lines clearly do the stated job, something seems amiss. Each of them passes very close to some of the training data-points. This can cause poor generalization: for example, the shown test observation 'A' lies closer to the positive samples but will get classified as negative class by boundary L2. This clearly is undesirable.

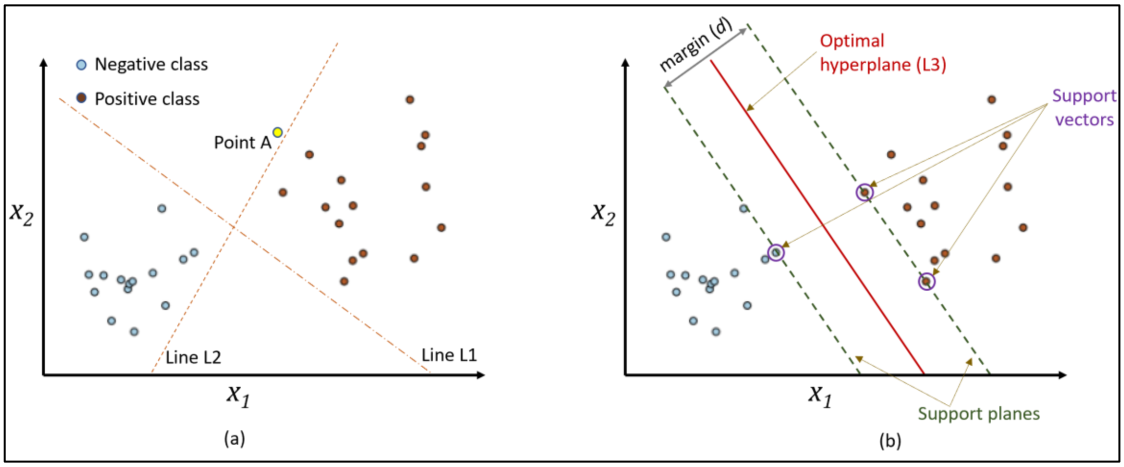


Figure 12.1: (a) Training data with test sample A (b) Optimal separating boundary

The optimal separating line/decision boundary, line L3 in Figure 12.1b, lies as far away as possible from either class of data. L3, as shown, lies midway of the support planes (planes that pass-through training points closest to the separating boundary). During model fitting, SVM simply finds this optimal boundary that corresponds to the maximum margin (distance between the support planes). In Figure 12.1, any other orientation or position of L3 will reduce the margin and will make L3 closer to one class than to the other. Large margins make model predictions robust to small perturbations in the training samples.

Points that lie on the support planes are called support vectors[13] and completely determine the optimal boundary, and hence the name, support vector machines. In Figure 12.1, if support vectors are moved, line L3 may change. However, if any non-support vectors are removed, L3 won't get affected at all. We will see later how the sole dependency on the support vectors imparts computational advantage to the SVMs.

[13] Calling data-points as vectors may seem weird. While this terminology is commonly used in general SVM literature, support vectors refer to the vectors originating from origin with the data-points on support planes as their tips.

Rest of the Chapter 12 not shown in this preview

Chapter 13

Decision Trees and Ensemble Learning for Fault Detection

Imagine that you are in a situation where even after your best attempts your model could not provide satisfactory performance. What if we tell you that there exists a class of algorithms where you can combine several 'versions' of your 'weak' performing models and generate a 'strong' performer that can provide more accurate and robust predictions compared to its constituent 'weak' models? Sounds too good to be true? It's true and these algorithms are called ensemble methods.

Ensemble methods are often a crucial component of winning entries in online ML competitions such as those on Kaggle. Ensemble learning is based on a simple philosophy that committee wisdom can be better than an individual's wisdom! In this chapter, we will look into how this works and what makes ensembles so powerful. We will study popular ensemble methods like random forests and XGBoost.

The base constituent models in forests and XGBoost are decision trees which are simple yet versatile ML algorithms suitable for both regression and classification tasks. Decision trees can fit complex and nonlinear datasets, and yet enjoy the enviable quality of providing interpretable results. We will look at all these features in detail. Specifically, we will cover the following topics

- Introduction to decision trees and random forests
- Introduction to ensemble learning techniques (bagging, Adaboost, gradient boosting)
- Fault detection and classification for gas boilers using decision trees and XGBoost

13.1 Decision Trees: An Introduction

Decision trees (DTs) are inductive learning methods which derive explicit rules from data to make predictions. They partition the feature space into several (hyper) rectangles and then fit a simple model (usually a constant) in each one. As shown in Figure 13.1 for a binary classification problem in 2D feature space, the partition is achieved via a series of if-else statements. As shown, the model is represented using branches and leaves which lead to a tree-like structure and hence the name decision tree model. The questions asked at each node make it very clear how the model predictions (class A or class B) are being generated. Consequently, DTs become the model of choice for applications where ease of rationalization of model results is very important.

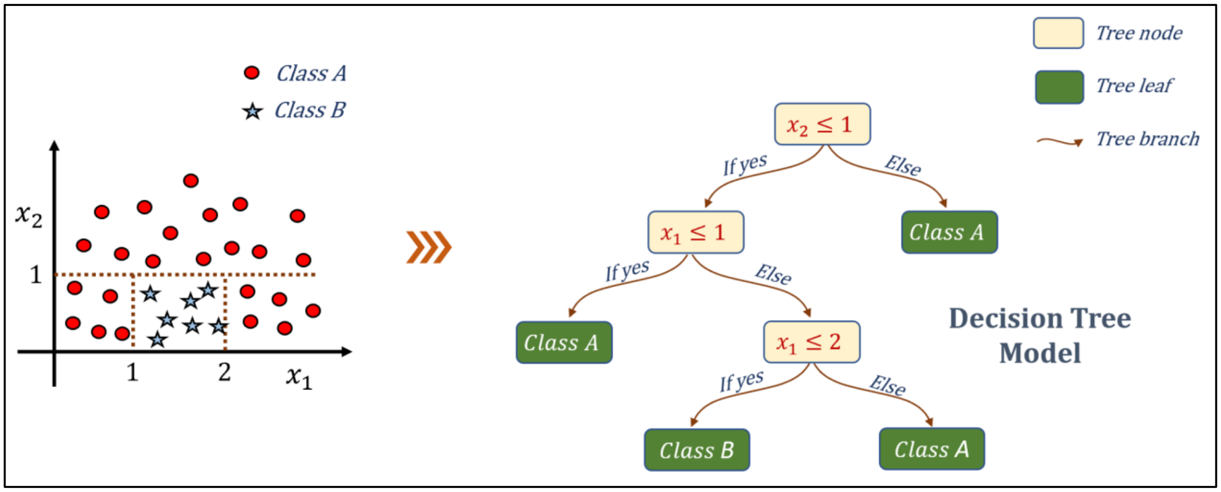


Figure 13.1: A decision tree with constant model used for binary classification in a 2D space

The trick in DT model fitting lies in deciding which questions to ask in the if-else statements at each node of the tree. During fitting, these questions split the feature space into smaller and smaller subregions such that the training observations falling in a subregion are similar to each-other. The splitting process stops when no further gains can be made or stopping criteria have been met. Improper choices of splits will generate a model that does not generalize well. In the next subsection, we will study a popular DT training algorithm called CART (classification and regression trees) which judiciously determines the splits.

Mathematical background

CART algorithm creates a binary tree, i.e., at each node two branches are created that split the dataset in such a way that overall data ‘impurity’ reduces. To understand this, consider

Rest of the Chapter 13 not shown in this preview

Chapter 14

Proximity-based Techniques for Fault Detection

Most of the anomaly detection techniques that we have studied so far work by finding some structure in training dataset, such as the low-dimensional manifold in PCA, NOC boundary in SVDD, optimal separating hyperplane in SVM, etc. However, another popular class of methods exists that utilizes a very straightforward and natural notion of anomalies as data points that are far away or isolated from the NOC data samples; logically, these methods are classified as proximity-based methods.

Proximity of a data point can simply be defined as its distance (as done in *k-NN* method) from its neighbors. An abnormal data point lies far away from other NOC data and therefore, its nearest neighbors' distances will be large compared to those for NOC samples. Another related but slightly different notion of proximity is the density or number of other data points in a local region around a test sample. Local outlier factor (LOF) is a popular method in this category wherein samples not lying in dense region are classified as anomalies. The third technique, isolation forest (IF), that we will study in this chapter uses the similar notion that anomalies are 'far and between'. Here, the data space is split until each data point gets 'isolated'. Anomalies can be isolated easily and require very few splits compared to NOC samples that lie close to each other.

You may have realized that these techniques generate interpretable results and are easy to understand. Correspondingly, they come in pretty handy to analyze complex system whose characteristics may not be well-known *a priori*. Let's now get down to business. We will cover the following topics

- Introduction to k-NN technique
- Introduction to LOF technique
- Introduction to isolation forest technique
- Applications of k-NN, LOF, and IF for fault detection in semiconductor manufacturing process

14.1 KNN: An Introduction

The k-nearest neighbors (k-NN or KNN) algorithm is a versatile technique based on a simple intuitive idea that the label/value for a new sample can be obtained from the labels/values of closest neighboring samples (in the feature space) from the training dataset. The parameter *k* denotes the number of neighboring samples utilized by the algorithm. As shown in Figure 14.1, k-NN can be used for both classification and regression. For classification, k-NN assigns test sample to the class that appears the most amongst the *k* neighbors. For regression, the predicted output is the average of the value of the *k* neighbors. Due to its simplicity, k-NN is widely used for pattern classification and was included in the list of top 10 algorithms in data mining.[14]

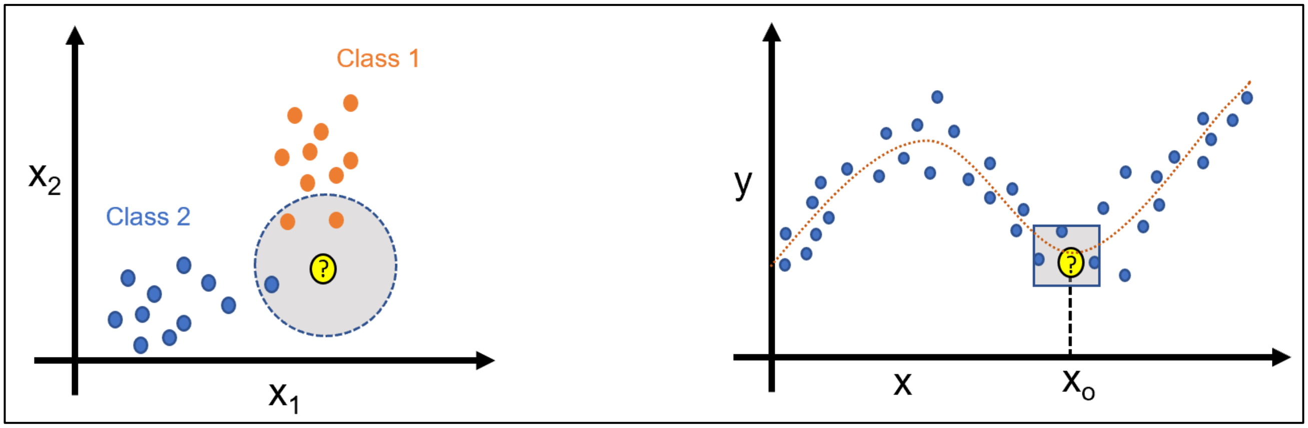


Figure 14.1: k-NN illustration for classification (left) and regression (right). Yellow data point denotes unknown test sample. The grey-shaded region represents the neighborhood with 3 nearest samples.

k-NN belongs to the class of lazy learners where models are not built explicitly until test samples are received. At the other end of the spectrum, eager learners (like, SVM, decision trees, ANN) 'learn' explicit models from training samples. Unsurprisingly, training is slower, and testing is faster for eager learners. KNN requires computing the distance of the test sample from all the training samples, therefore, k-NN also falls under the classification of instance-based learning. Instance-based learners make predictions by comparing the test sample with training instances stored in memory. On the other hand, model-based learners do not need to store the training instances for making predictions.

[14] Wu et al., Top 10 algorithms in data mining. Knowledge and Information systems, 2008.

Rest of the Chapter 14 not shown in this preview

Part 5

Artificial Neural Networks for Process Monitoring

Chapter 15

Fault Detection & Diagnosis via Supervised Artificial Neural Networks Modeling

It won’t be an exaggeration to say that artificial neural networks (ANNs) are currently the most powerful modeling construct for describing generic nonlinear processes. ANNs can capture any kind of complex nonlinearities, don’t impose any specific process characteristics, and don’t demand specification of process insights prior to model fitting. Furthermore, several recent technical breakthroughs and computational advancements have enabled (deep) ANNs to provide remarkable results for a wide range of problems. Correspondingly, ANNs have re(caught) the fascination of data scientists and the process industry is witnessing a surge in successful applications of ML-based process control, predictive maintenance, inferential modeling, and process monitoring.

ANNs can be used in both supervised and unsupervised learning settings. While we will cover the supervised learning-based FDD applications of ANNs in this chapter, unsupervised learning is covered in the next chapter. Supervised fitting of ANN models are applicable if you have adequate number of historical faulty samples (so that you can fit a fault classification model) or your signals are categorizable into predictors and response variables (so that you can fit a regression model and monitor residuals). Different forms of ANN architectures have been devised (such as FFNNs, RNNs, CNNs) to deal with datasets with different characteristics. CNNS are mostly used with image data and therefore, we will study FFNN and RNN in this chapter.

There is no doubt that ANNs have proven to be monstrously powerful. However, it is not easy to tame this monster. If the model hyperparameters are not set judiciously, it is very easy to end up with disappointing results. The reader is referred to Part 3 of Book 1 of this series for a detailed exposition on ANN training strategies and different facets of ANN models. In this chapter, the focus is on exposing the user to how ANNs can be used to build process monitoring applications. Specifically, the following topics are covered

- Introduction to ANNs
- Introduction to RNNs
- Process monitoring using ANNs via external analysis

15.1 ANN: An Introduction

Artificial neural networks (ANNs) are nonlinear empirical models which can capture complex relationships between input-output variables via supervised learning or recognize data patterns via unsupervised learning. Architecturally, ANNs were inspired by human brain and are a complex network of interconnected neurons as shown in Figure 15.1. An ANN consists of an input layer, a series of hidden layers, and an output layer. The basic unit of the network, neuron, accepts a vector of inputs from the source input layer or the previous layer of the network, takes a weighted sum of the inputs, and then performs a nonlinear transformation to produce a single real-valued output. Each hidden layer can contain any number of neurons.

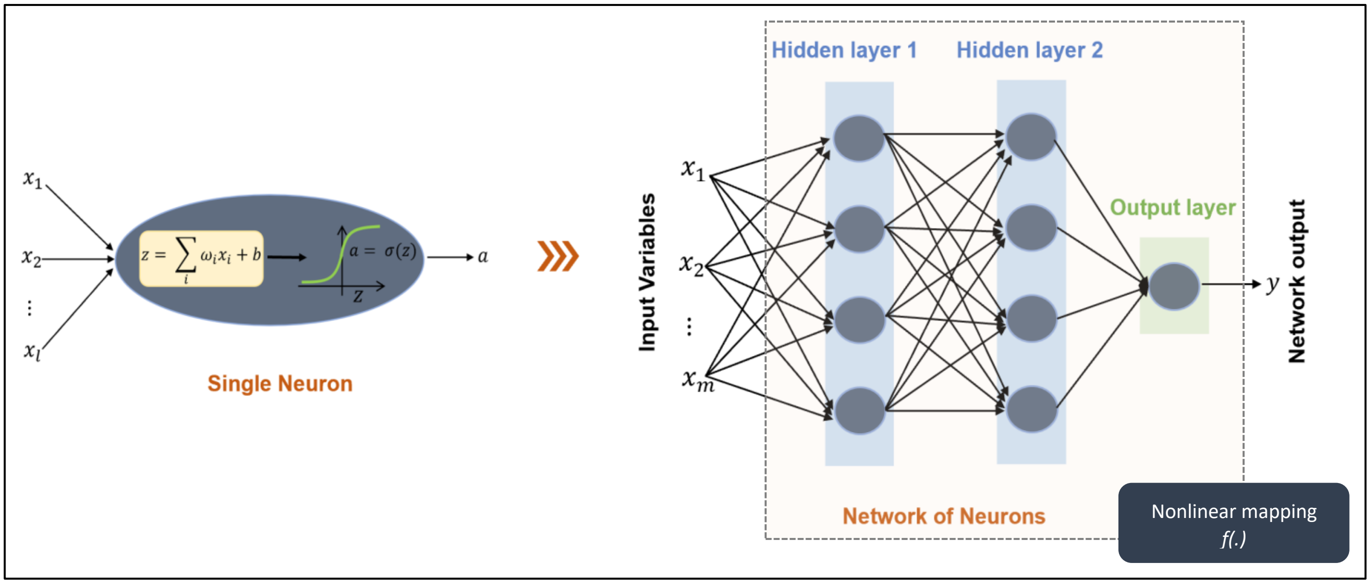


Figure 15.1: Architecture of a single neuron and feedforward neural network with 2 hidden layers

The network shown in Figure 15.1 is an example of a fully-connected feed-forward neural network (FFNN), the most common type of ANN. In FFNN, signals flow in only one direction, from the input layer to the output layer via hidden layers. Neurons between consecutive layers are connected fully pairwise and neurons within a layer are not connected.

What is deep learning

In a nutshell, using an ANN with a large number of hidden layers to find relationship/pattern in data is deep learning (technically, ≥ 2 hidden layers implies a deep neural network (DNN)). Several recent algorithmic innovations have overcome the model training issues for DNNs which have resulted in the DNN-led AI revolution we are witnessing today.

Rest of the Chapter 15 not shown in this preview

Chapter 16

Fault Detection & Diagnosis via Unsupervised Artificial Neural Networks Modeling

In the previous chapter, we looked at supervised fitting of artificial neural networks where either the faults labels were available for historical samples or the process variables were divided into predictors and response variable sets. However, you are very likely to encounter situations where you only have NOC samples in your training dataset without any predictor/response division. In Part 3 of this book, we studied a powerful technique suitable for such datasets, called PCA; PCA, however, is limited to linear processes. Nonetheless, the underlying mechanism of extracting the most representative features of training dataset and compressing it into a feature space with reduced dimensionality need not be limited to linear systems. ANNs excel at handling nonlinear systems and extracting hidden patterns in high-dimensional datasets. Unsurprisingly, clever neural network-based architectures have been devised to enable unsupervised fitting of nonlinear datasets. Two popular models in this category are autoencoders (AEs) and self-organizing maps (SOMs)

Autoencoders are ANN-based counterparts of PCA for nonlinear processes. Here, low-dimensional latent feature space is derived via nonlinear transformation and, just like we did for PCA, the systematic variations in the feature space and the reconstruction errors are handled separately to provide the monitoring statistics. Autoencoders are very popular for building FDD solutions for nonlinear processes. They are also commonly used to provide intermediate low-dimensional features which are then used for subsequent modeling (clustering, fault classification, etc.). SOM is another variant of neural network-based architecture that project a high-dimensional dataset onto a 2D grid (yes, you read that right!). Here, latent variables are not derived, albeit the focus is on ensuring that the topology of the projected data is similar to that in the original measurement space. This feature renders SOMs very useful for data visualization, clustering, and fault detection applications.

We will undertake in depth study of both these powerful techniques in this chapter. Specifically, the following topics are covered

- Introduction to autoencoders and self-organizing maps
- Fault detection and diagnosis using autoencoders: application to FCCU process
- Fault detection and diagnosis using SOMs: application to semiconductor dataset

16.1 Autoencoders: An Introduction

An autoencoder (AE) in its basic form is a 3-layered ANN consisting of an input layer, a hidden layer, and an output layer as shown in Figure 16.1. An AE takes an input $x \in \mathbb{R}^n$ and predicts a reconstructed $\hat{x} \in \mathbb{R}^n$ as an output. To prevent the network from trivially copying x to $\hat{x}$, the hidden layer is constrained to be much smaller than *n* (the number of neurons in the hidden layer, say *m*, gives the dimension of the latent/feature space). This forces the network to capture only the systematic variations in input data and learn only the most representative features as the latent variables. The nonlinear activation function of the neurons in the hidden layer enables the latent variables to be nonlinearly related to the input variables. During model fitting, the gap between x and $\hat{x}$ (termed reconstruction error) is minimized to find network parameters. The basic AE network can be made deeper by adding more hidden layers resulting in deep (or stacked) autoencoders.

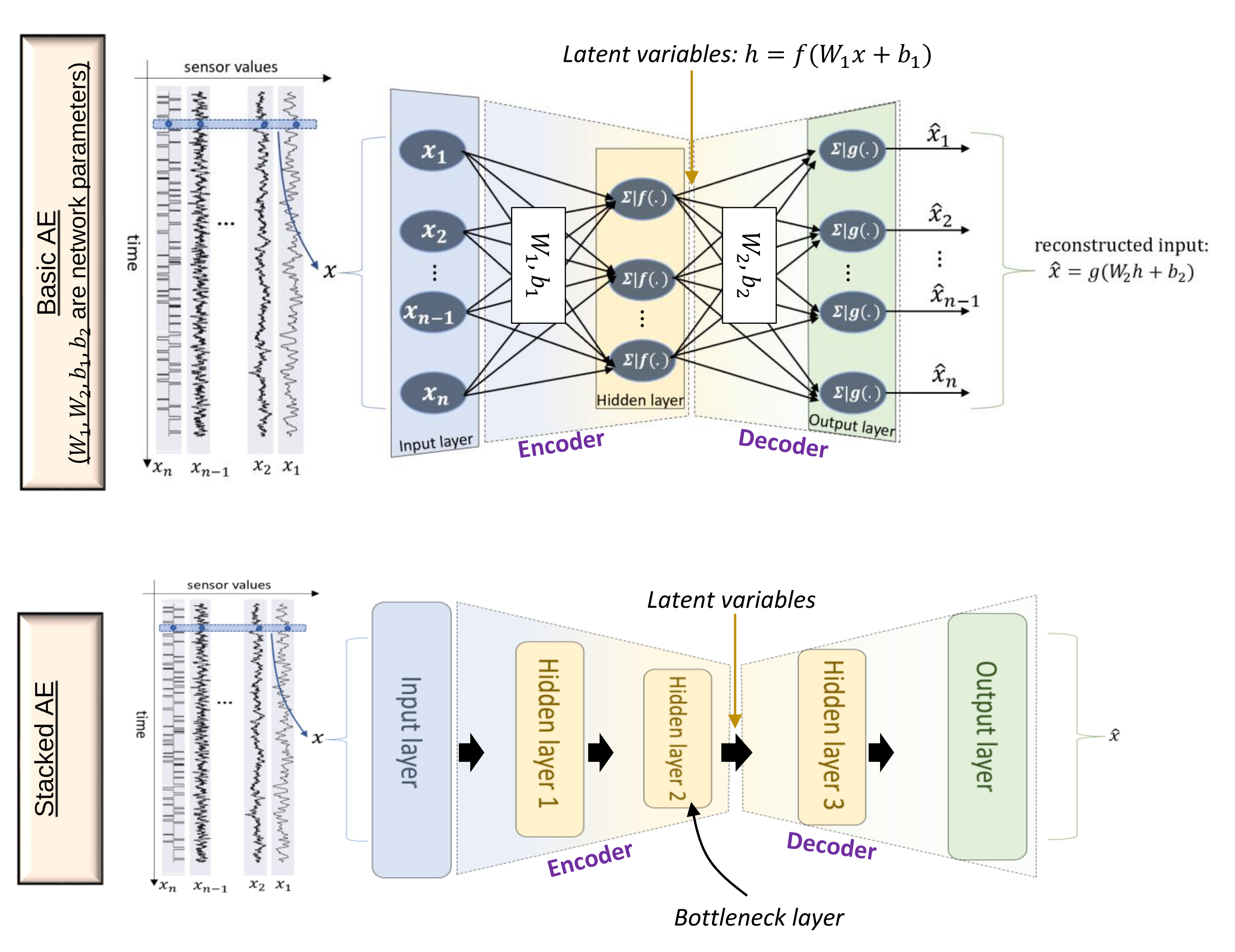
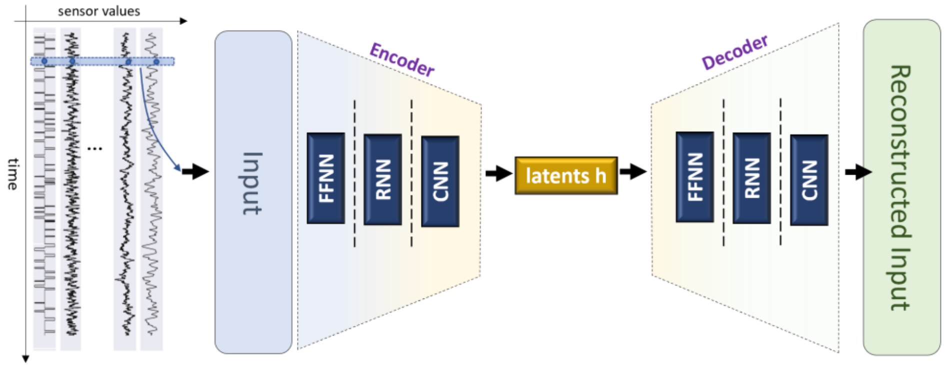


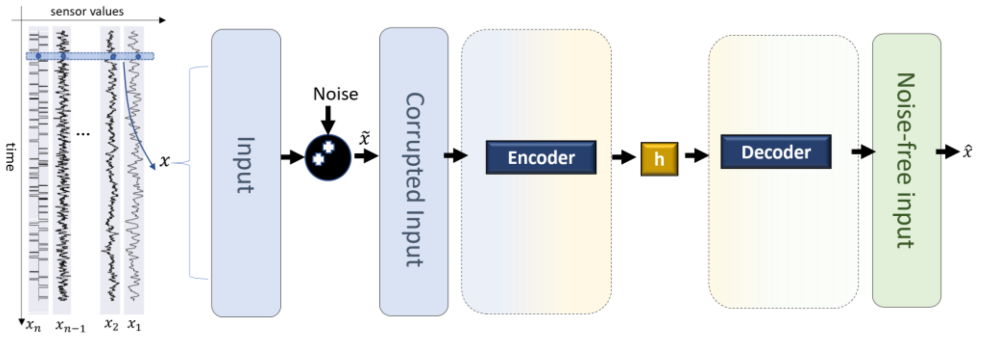
Figure 16.1: Autoencoder architecture

The symmetrical and sandwich nature of the deep AE architecture should be apparent wherein the sizes of the layers first decrease and then increase. Care must be taken though to not use too many hidden layers; otherwise, the network will overfit and may simply learn the identity mapping from x to $\hat{x}$! Moreover, in the previous figure, you will notice that AE architecture is divide into an encoder part and a decoder part. An encoder projects or codify an input sample *x* to lower dimensional feature *h*. The decoder maps the feature vector back to the input space. The encoder-decoder form makes the AR architecture very flexible. Once an AE has been trained, one can use the encoder as a standalone network to obtain the latent variables. Moreover, you are not limited to using only FFNN in the encoders and decoders. RNNs and CNNs are also frequently employed. RNN-based AE is used as a nonlinear counterpart of dynamic PCA.



Vanilla AE vs Denoising AE

The form of autoencoder we saw in Figure 16.1 is the conventional or vanilla form wherein the network is forced to find patterns in data by constraining the size of coding/latent variable (*m*) to be less than the size of input variable (*n*). This is also called an undercomplete autoencoder. An alternative way of forcing an autoencoder to learn only the systematic variation in data is by corrupting input data by adding synthetic noise and then training the network to reconstruct the uncorrupted input. Such autoencoders are called denoising autoencoders and its representative architecture is shown below. Note that we did not explicitly represent encoder having number of neurons in hidden layer less than the number of input variables. Denoising AE allow having $m \geq n$.



Dimensionality reduction via autoencoders

To see autoencoders in action, let's apply it for the dimensionality reduction of a simulated dataset from a fluid catalytic cracking unit (FCCU[15]) shown in Figure 16.2. FCCUs are critical units in modern oil refineries and convert heavy hydrocarbons into lighter and valuable products such as LPG, gasoline, etc. As shown, the FCCU operation involves catalytic reaction, catalyst regeneration, and distillation. A total of 46 signals are made available as outputs (recorded every minute). Data has been provided in 7 CSV files. Each file contains data from one simulation. One of the CSV files contain NOC data over a period of 7 days with varying feed flow. Five faults have been simulated one at a time in 5 separate simulations. We will work with the 7 days of NOC data.

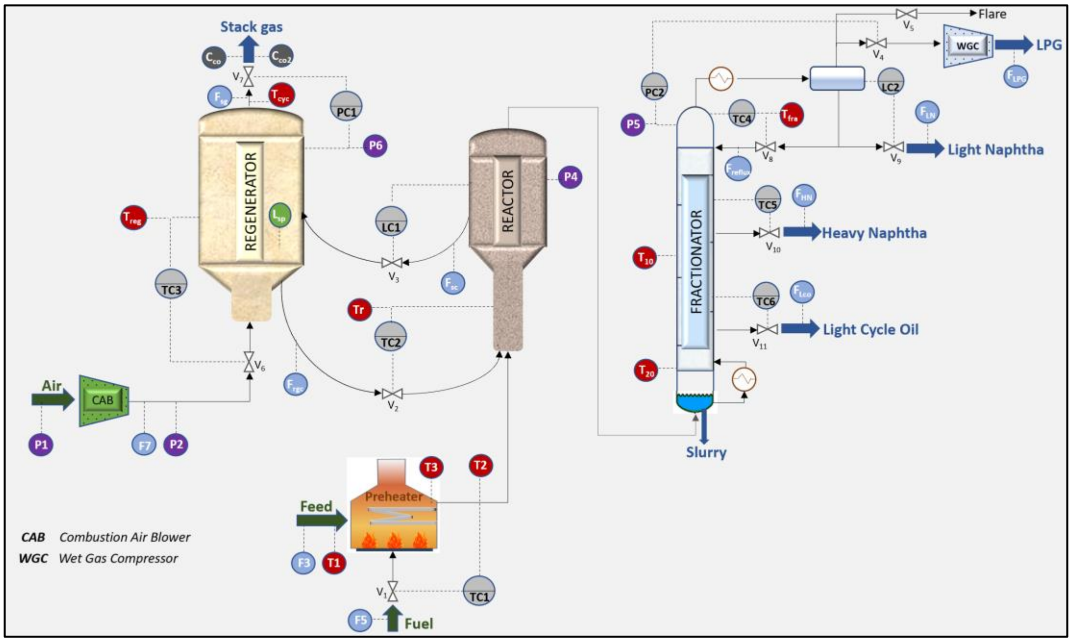


Figure 16.2: Fluid catalytic cracking unit with available measurements

We know that most of the variability in the data is driven from the variations in the feed flow and therefore, we will attempt to generate a 1D latent space (*m=1*).

```
# import required packages
import numpy as np, pandas as pd, matplotlib.pyplot as plt
import tensorflow
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

[15] Details on the system and datasets available are provided in detail at https://mlforpse.com/fccu-dataset/.

Rest of the Chapter 16 not shown in this preview

Part 6

Vibration-based Condition Monitoring

Chapter 17

Vibration-based Condition Monitoring: Signal Processing and Feature Extraction

Rotating machinery, which includes motors, compressors, pumps, turbines, fans, etc., form the backbone of industrial operations. Unsurprisingly, a large fraction of operation downtime can be attributed to the failures of these machines. Over the last decade, the process industry has adopted predictive maintenance as the means to proactively handle these failures and the technique that has largely become synonymous with predictive maintenance is vibration-based condition monitoring (VCM). All rotating machines exhibit vibratory motions and different kind of faults produce characteristic vibratory signatures. This makes VCM a reliable and effective tool for health management of rotating equipment. Considering the importance of VCM in process industry, its different aspects are covered in this part of the book.

Vibrations are usually measured at very high frequency and the large volume of data makes analysis of raw data difficult. Correspondingly, processing vibration data and extracting meaningful features that can provide early signs of failures become very crucial. Traditionally, these features have been analyzed by vibration experts. However, in recent times, several successful applications of ML-based VCM have been reported. All the techniques that we have studied in the previous parts of the book can be used for VCM. While we will look at ML-based VCM in the next chapter, this chapter sets the foundations for VCM and covers vibration data processing and feature extraction.

Over the years, VCM practitioners and researchers have fine-tuned the art of vibration monitoring and have come up with several specialized and advanced techniques. Arguably, it is easy for a beginner to feel 'lost' in the world of VCM. The current and the following chapters will help provide some order to this seemingly chaotic world. Specifically, the following topics are covered

- Basics of vibrations
- VCM workflow
- Spectral analysis of vibration signal
- Time domain, frequency domain, and time-frequency domain feature extraction

17.1 Vibration: A Gentle Introduction

Vibrations are simply back and forth motion of machines around their position of rest. All rotating machines (motors, blowers, chillers, compressors, turbines, etc.) exhibit vibratory motion under normal and faulty conditions. Figure 17.1 shows a representative setup for vibration sensing of an industrial machine. The sensors (transducers) convert vibratory motion (of displacement, velocity, or acceleration) into analogue electrical signals which are digitized and stored. The figure below shows how the recorded signal looks like on a time-axis for a machine with gradually degrading condition. The increasing vibration levels indicate underlying machine issues.

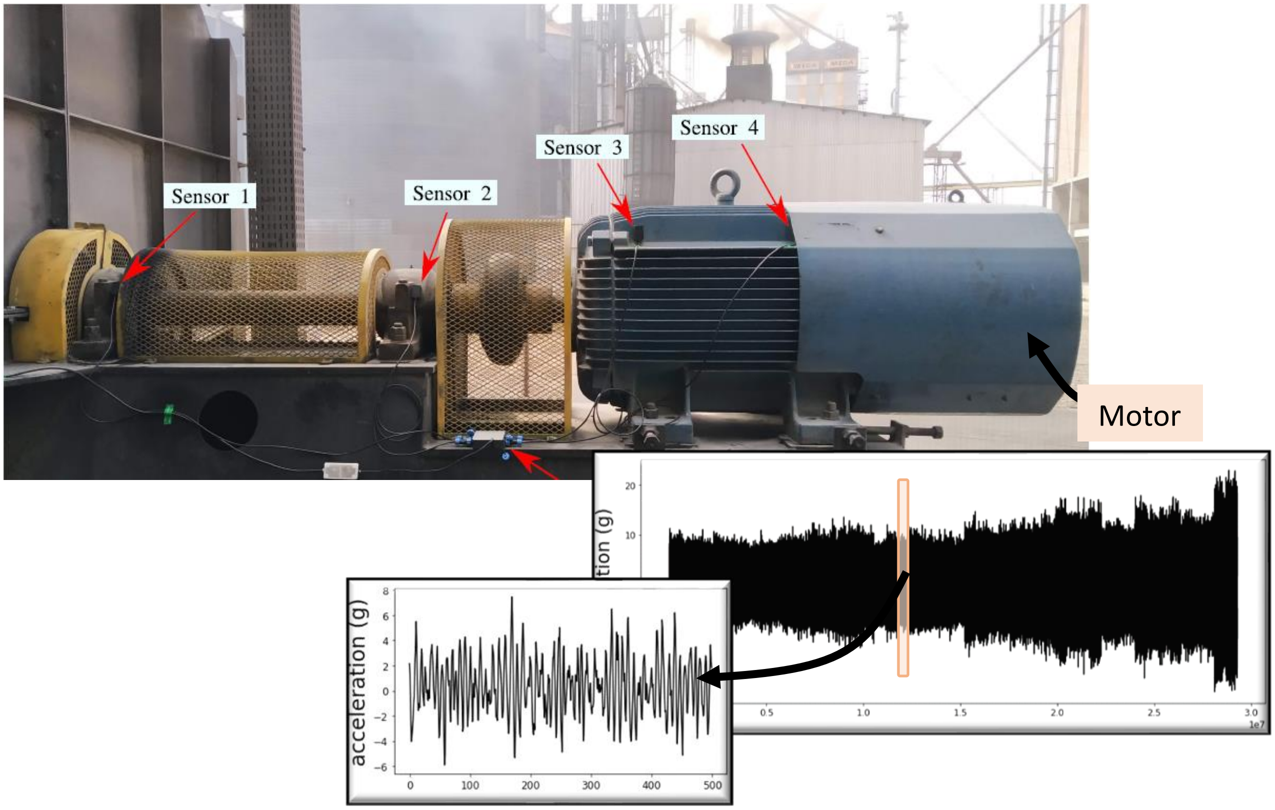


Figure 17.1: Representative vibration monitoring system[16]

The components of rotating machines (rotors, bearings, gears) undergo different types of failures due to well-studies causes such as mechanical looseness, misalignment, cracks, etc.

[16] Romanssini et al., A Review on Vibration Monitoring Techniques for Predictive Maintenance of Rotating Machinery. *Eng*, 2023. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

Rest of the Chapter 17 not shown in this preview

Chapter 18

Vibration-based Condition Monitoring: Fault Detection & Diagnosis

Vibration-based condition monitoring was already a widely adopted technique in process industry long before ML craze took over the manufacturing world. The International Organization for Standardization (ISO) has come up with alarm limits for vibration RMS for different classes of rotating machines. Additionally, VCM researchers have worked diligently to discover the characteristics signatures of failures in different components of a rotating machines. Correspondingly, several rules of thumb and heuristics have been devised to pinpoint root causes of faults using vibration features. However, these heuristics do not cover all possible fault scenarios and a vibration expert is still required to conduct analysis and interpretation of vibration signal features. Fortunately, the advent of machine learning has made VCM more accessible to generic process data scientists.

Several different types of ML models have been reported in VCM literature for fault classification, fault detection, and fault diagnosis. For example, fault detection applications have been built by using the whole spectrum (or waveform) as input to an autoencoder or spectrogram image as input to a CNN (convolutional neural network) model. ML models don the cap of a vibration expert to find the patterns in vibration signal, distinguish between NOC and abnormal vibrations, and discriminate between different fault conditions. In this chapter, we will look at one such implementation of ML-based VCM. Specifically, the following topics are covered

- VCM workflow
- Classical approaches for VCM
- SVM-based fault classification of motors

18.1 VCM Workflow: Revisited

Vibration signals contain indicators of machine faults. Previously, we saw the steps commonly taken to 'amplify' these indicators through judicious extraction of features. In this chapter, we will focus on how these features are used to make inferences regarding health of rotating machinery. Figure 18.1 shows some of the approaches commonly employed. The classical approaches include, amongst others, simply looking for the presence of harmonics in the spectrum and comparing individual features against ISO-recommended thresholds. In recent times, ML-based VCM is gradually becoming more popular. Any of the ML techniques that we have seen in the previous parts of the book can be employed.

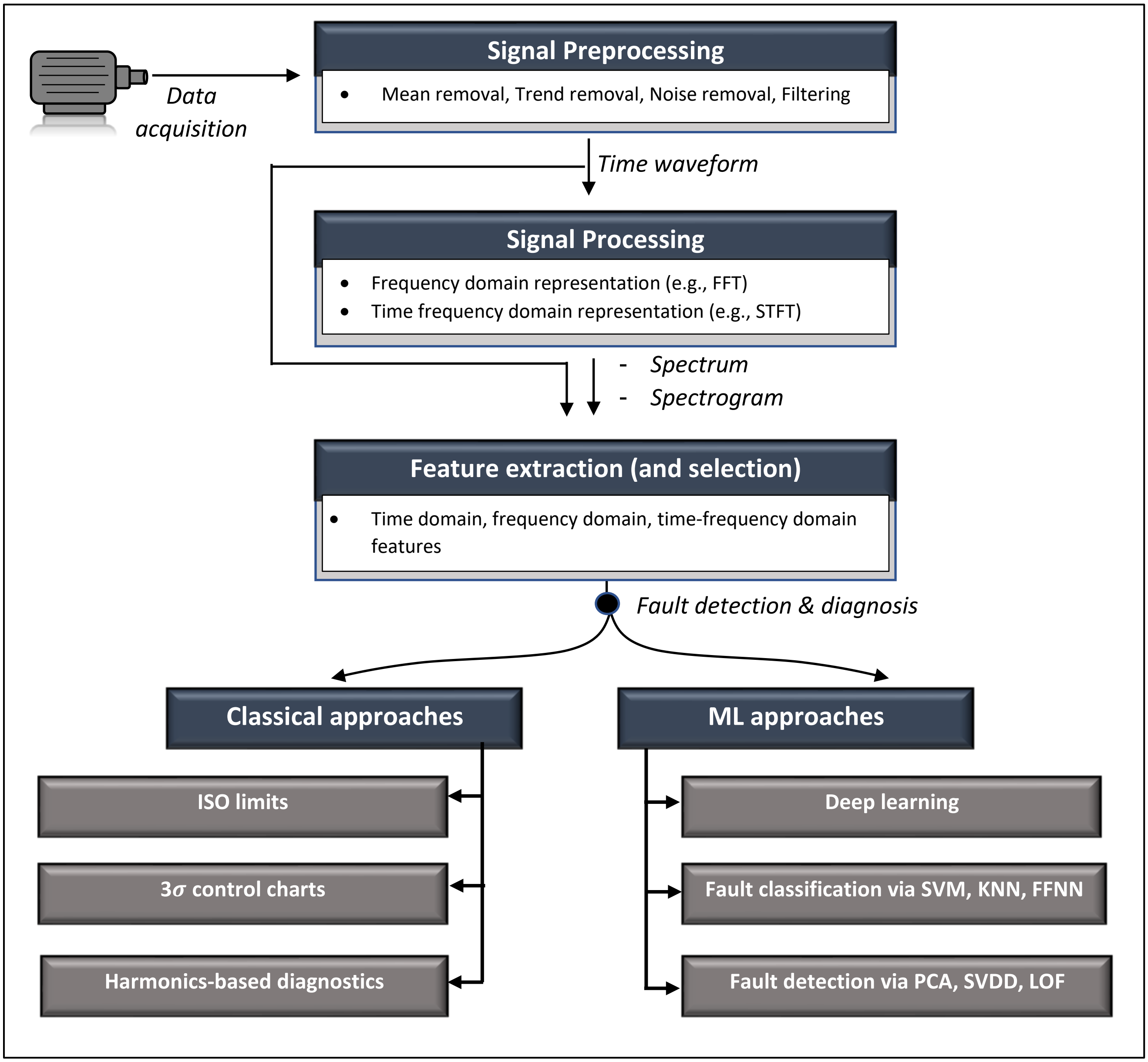

Data acquisition
Signal Preprocessing
Mean removal, Trend removal, Noise removal, Filtering
Time waveform
Signal Processing
Frequency domain representation (e.g., FFT)
Time frequency domain representation (e.g., STFT)
- Spectrum
- Spectrogram
Feature extraction (and selection)
Time domain, frequency domain, time-frequency domain features
Fault detection & diagnosis
Classical approaches
ML approaches
ISO limits
3σ control charts
Harmonics-based diagnostics
Deep learning
Fault classification via SVM, KNN, FFNN
Fault detection via PCA, SVDD, LOF

Figure 18.1: Vibration condition monitoring workflow - revisited

Rest of the Chapter 18 not shown in this preview

Part 7

Predictive Maintenance

Chapter 19

Fault Prognosis: Concepts & Methodologies

All machines eventually break and plant operators have traditionally relied upon regular time-based (preventive) maintenance to avoid costly downtimes due to machinery failures. Although economically inefficient, preventive maintenance remained the default approach in process industry for a long time. Only in recent times, condition-based maintenance approach has gained widespread acceptance wherein a machine's real-time data is used to assess the machine's health, detect failures, and trigger (on-demand) maintenance. However, the recent advancement in data mining has brought another step change in the mindset of plant reliability personnel: mere detection of machine faults is no longer good enough; accurate forecast of the fault's progression leading to predictive maintenance (PdM) is the new vogue. The lure of PdM is obvious – it facilitates advance planning of maintenance, better management of spare part's inventory, etc. Correspondingly, PdM is the holy grail that industrial executives are striving for to remain competitive.

PdM, in essence, involves fault prognosis or the prediction of a machine's health degradation over time after detection of incipient faults. Different PdM methodologies are employed depending on the availability of fundamental knowledge of fault's mechanism, historical run-to-failure data, etc. The dominant PdM approach involves computation of a health indicator (HI) that summarizes the state of a machine health and shows a clear degradation trend as an incipient fault progresses from incipience to high severity. HI allows computation of RUL (remaining useful life) which is the remaining time until fault severity crosses failure threshold necessitating the machine being taken out of service.

Several different strategies have been devised for computation of HIs and the subsequent RUL estimation. While the RUL estimation strategies are covered in detail in the next chapter, this chapter focusses on the data-driven methods for HI computations. Specifically, the following topics are covered

- Concepts and methodologies for PdM
- Fault prognosis: introduction and workflow
- Approaches for health indictor computation
- Fault prognosis case study for wind turbines

19.1 Fault Prognosis: Introduction & Workflow

Fault prognosis simply refers to the task of estimating the progression of health degradation of a machine[17]. Fault prognosis kick in after a fault has been detected. The end objective of fault prognosis is to estimate the time remaining until fault severity hits failure threshold. A machine or an operation unit may be kept in operation (even with faults) until it reaches failure conditions. Therefore, estimation of the time remaining or RUL can help plant operators maximize an equipment lifetime and plan maintenance judiciously. Figure 19.1 presents the different prognostic methodologies that can be employed depending on the level of available information about fault mechanism and past fault data.

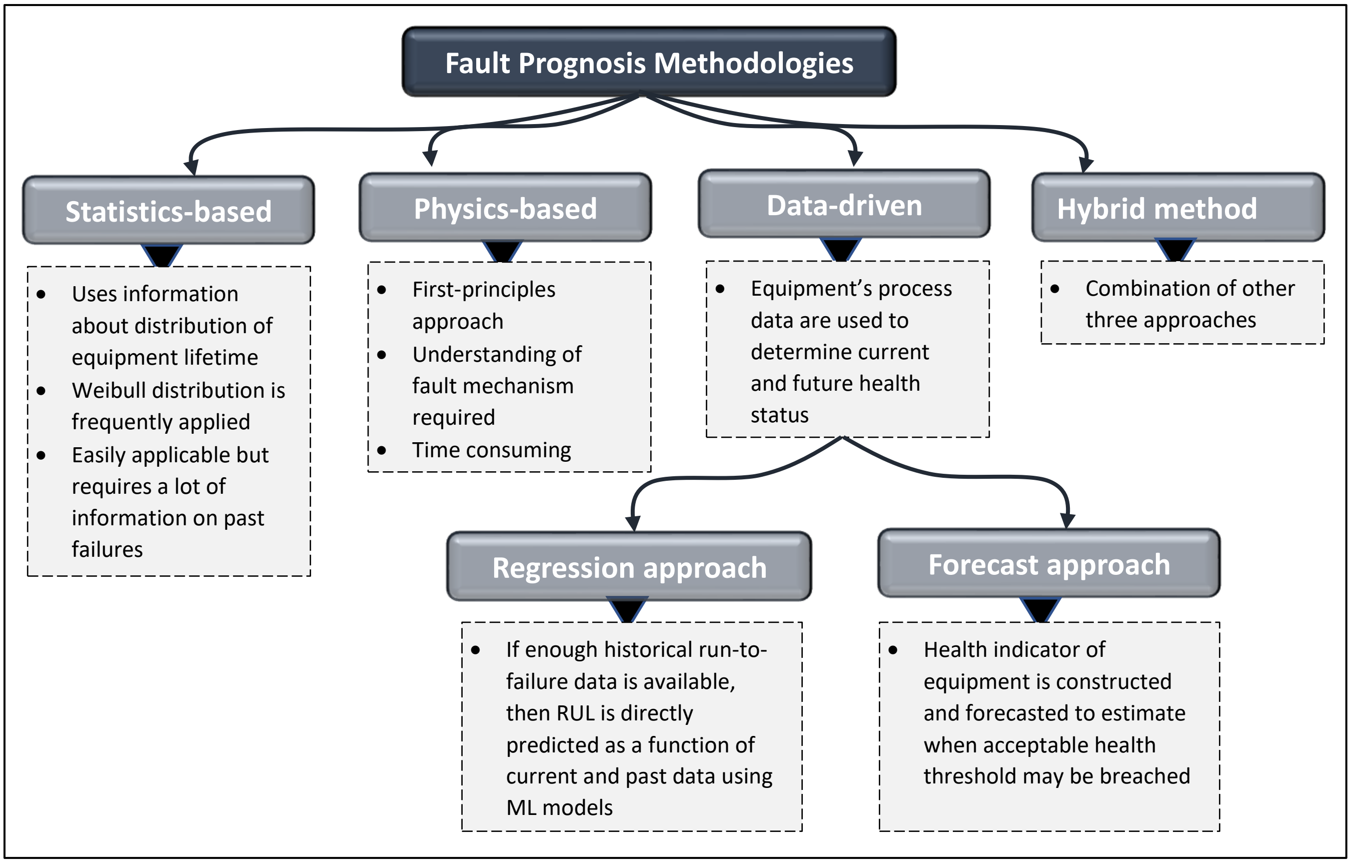

Fault Prognosis Methodologies
Statistics-based
- Uses information about distribution of equipment lifetime
- Weibull distribution is frequently applied
- Easily applicable but requires a lot of information on past failures
Physics-based
- First-principles approach
- Understanding of fault mechanism required
- Time consuming
Data-driven
- Equipment's process data are used to determine current and future health status
Hybrid method
- Combination of other three approaches
Regression approach
- If enough historical run-to-failure data is available, then RUL is directly predicted as a function of current and past data using ML models
Forecast approach
- Health indicator of equipment is constructed and forecasted to estimate when acceptable health threshold may be breached

Figure 19.1: Prognostics Methodologies

Among the shown approaches, HI-based approach is very popular. A shown in Figure 19.2, a curve showing the current trend of fault severity or health condition is computed. Thereafter, the future progression of the curve is predicted to estimate the RUL. In this chapter, we will look at how such curves can be generated in a data-driven way. The strategy for HI forecast is covered in detail in the next chapter.

[17] Fault prognosis is not limited to health prediction of machines only. It is applicable to a subprocess of a plant and the whole plant as well.

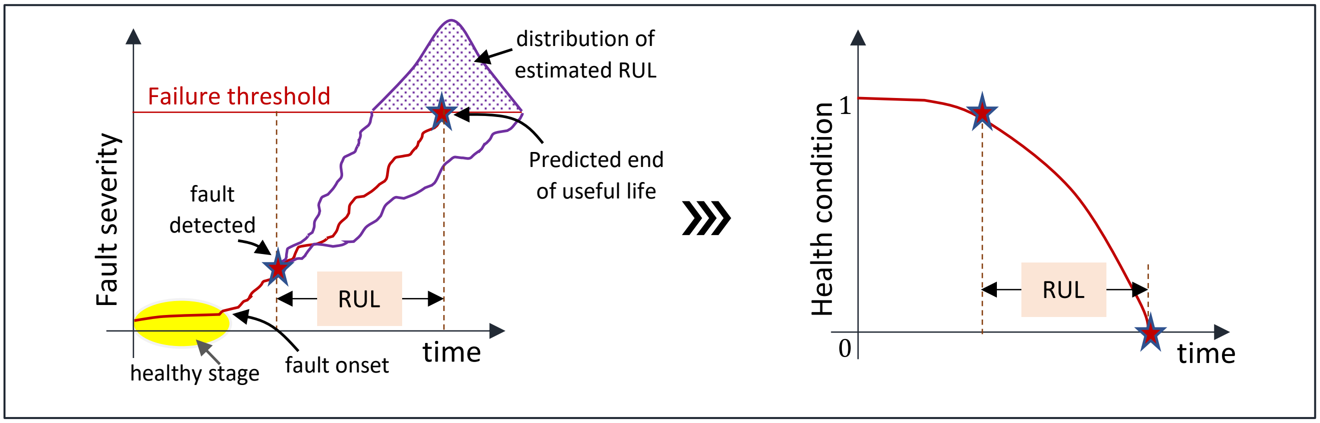


Figure 19.2: Fault severity and health condition progression with time

Rest of the Chapter 19 not shown in this preview

Chapter 20

Fault Prognosis: RUL Estimation

In the previous chapter, we introduced the concept of remaining useful life which is simply the time remaining until failure of an equipment. Three broad data-based techniques were mentioned that are: 1) reliability data-based approach wherein lifespan distribution of similar equipment is utilized to find the expected RUL 2) direct computation of RUL via regression-based ML modeling 3) computation of health indicator as an intermediate step. The first two approaches require information about the past lifespan of equipment and complete run-to-failure histories. However, it is difficult to get these data in process industry as very often machines get repaired before they reach failure stages (remember preventive maintenance!). This makes HI-based approach more suitable and, unsurprisingly, more popular. In the previous chapter, we saw how to compute HI for a wind turbine. We will take this case study to completion and show how to estimate the RUL.

Within the HI-based approach, two strategies are widely adopted. If decent amount of past run-to-failure data are available, then one can simply pick up the historical HI trend that matches the most with the current equipment's HI trajectory and use the historical lifespan to compute the required RUL. This is called similarity-based approach. A popular alternative is to simply use the existing HI values of current equipment and fit a curve to it to extrapolate it in the future and find when the failure threshold is breached. This is called degradation-based approach. We will go into more details into these two strategies in this chapter. Overall, the following topics are covered

- Introduction to RUL
- Health indicator-based RUL estimation strategies
- Health indicator degradation modeling for RUL estimation of a wind turbine
- Deep learning-based direct RUL estimation for a gas turbine

20.1 RUL: Revisited

In the previous chapter, we looked at some broad classes of strategies for RUL estimation. We also looked at how a health indicator can be calculated. Figure 20.1 reproduces Figure 19.1 and adds more details regarding HI-based approaches for RUL computation. The figure also highlights the four commonly employed strategies. As alluded to earlier, the choice of model depends on the type and amount of information available on past failures. If large amount of past run-to-failure data are available, then one can build a deep learning model to directly predict the RUL. We will see one such application in this chapter.

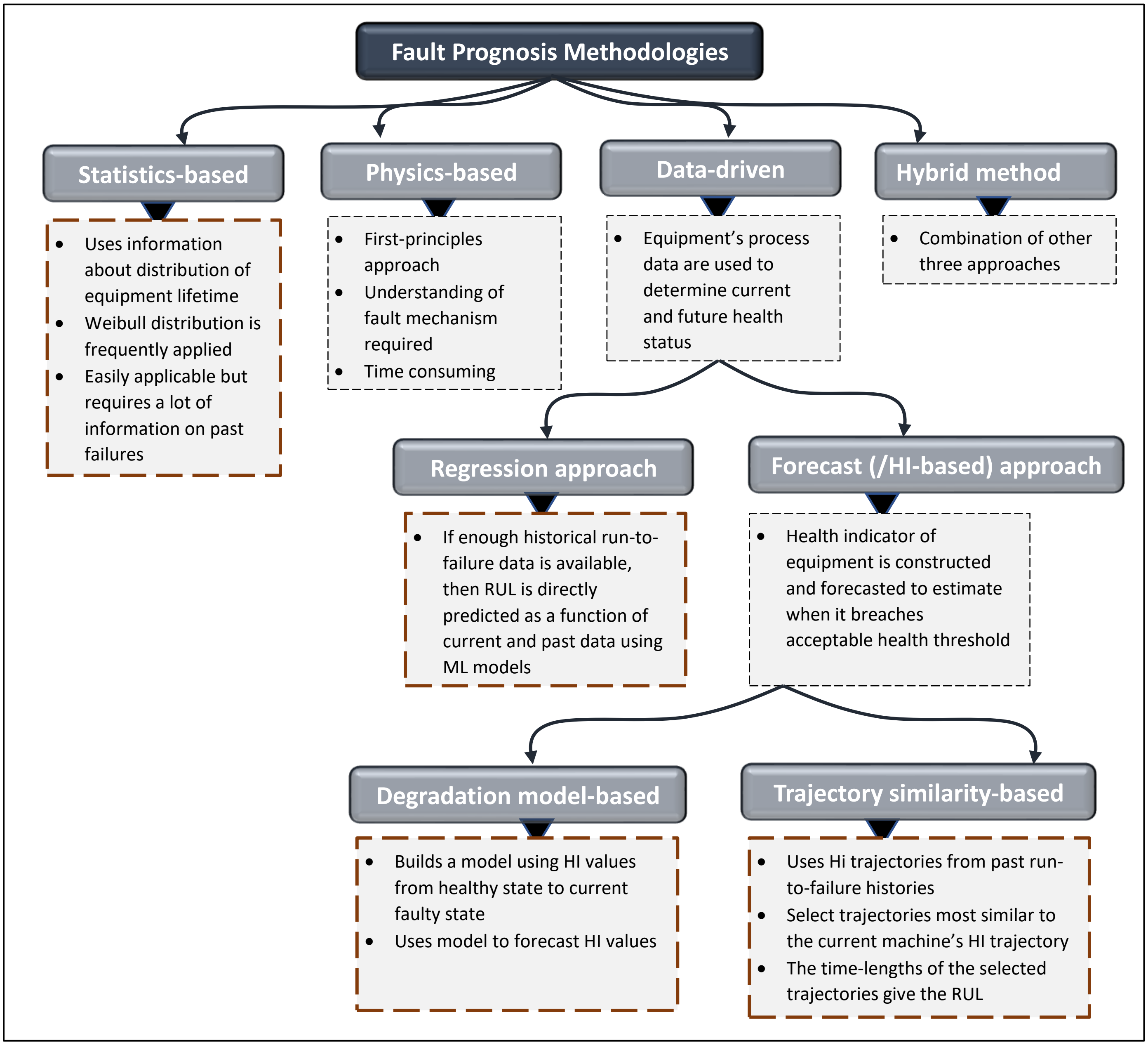

Fault Prognosis Methodologies
Statistics-based
Uses information about distribution of equipment lifetime
Weibull distribution is frequently applied
Easily applicable but requires a lot of information on past failures
Physics-based
First-principles approach
Understanding of fault mechanism required
Time consuming
Data-driven
Equipment's process data are used to determine current and future health status
Hybrid method
Combination of other three approaches
Regression approach
If enough historical run-to-failure data is available, then RUL is directly predicted as a function of current and past data using ML models
Forecast (/HI-based) approach
Health indicator of equipment is constructed and forecasted to estimate when it breaches acceptable health threshold
Degradation model-based
Builds a model using HI values from healthy state to current faulty state
Uses model to forecast HI values
Trajectory similarity-based
Uses Hi trajectories from past run-to-failure histories
Select trajectories most similar to the current machine's HI trajectory
The time-lengths of the selected trajectories give the RUL

Figure 20.1: Prognostics Methodologies

Rest of the Chapter 20 not shown in this preview

End of the book

Of process data science, By process data scientists, For process data scientists

Machine Learning in Python for Process and Equipment Condition Monitoring, and Predictive Maintenance

This book is designed to help readers quickly gain a working-level knowledge of machine learning-based techniques that are widely employed for building equipment condition monitoring, plantwide monitoring, and predictive maintenance solutions in process industry. The book covers a broad spectrum of techniques ranging from univariate control charts to deep-learning-based prediction of remaining useful life. Consequently, the readers can leverage the concepts learned to build advanced solutions for fault detection, fault diagnosis, and fault prognostics. The application-focused approach of the book is reader friendly and easily digestible to the practicing and aspiring process engineers, and data scientists. Upon completion, readers will be able to confidently navigate the Prognostics and Health Management literature and make judicious selection of modeling approaches suitable for their problems.

The following topics are broadly covered:

- *Exploratory analysis of process data*
- *Best practices for process monitoring and predictive maintenance solutions*
- *Univariate monitoring via control charts and time-series data mining*
- *Multivariate statistical process monitoring techniques (PCA, PLS, FDA, etc.)*
- *Machine learning and deep learning techniques to handle dynamic, nonlinear, and multimodal processes*
- *Fault detection and diagnosis of rotating machinery using vibration data*
- *Remaining useful life predictions for predictive maintenance*

www.MLforPSE.com