UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Data driven methods for updating fault detection and diagnosis system in chemical processes

Mohammad Hamed Ardakani

CEPIMA

Ph.D. Thesis:

Data Driven Methods for Updating Fault Detection and Diagnosis Systems in Chemical Processes

Mohammad Hamed Ardakani

Directed By:

Moisès Graells

Antonio Espuña

To my lovely parents.

Acknowledgment

My gratitude to CEPIMA group, especially its esteemed founder Professor Luis Puigjaner for all supports and considerations that I have received during my PhD.

I wish to thank Professor Antonio Espuña; a proficient scientist that has deep and genuine understanding of chemical engineering. Working with him and learning from him would be a great chance for any researcher.

I am thankful to Professor Moises Graells; an admirable professional professor. I believe that chemical engineering puzzles are not challenging enough for his amazing problem solving abilities. Certainly, the skills that I have learned from him will be with me forever.

I am grateful to Professor Rosa Mari; a reliable, kind and sympathetic friend in addition to be a great professor. The supports that I have received from her are unprecedented and unforgettable.

I would like to thank doctor Ahmed Shokry; a decent fiend and helpful cooperator. I have had his collaboration for doing my thesis, especially for data processing tasks.

Acknowledgment

I sincerely thank to Professor Gerard Escudero; without his unconditional helps doing this thesis was impossible. I have received his great supports for understanding, developing, programming and solving machine learning problems during all years of my PhD. Expert, smart and honest professor with the awesome personality.

Abstract

Modern industrial processes are becoming more complex, and consequently monitoring them has become a challenging task. Fault Detection and Diagnosis (FDD) as a key element of process monitoring, needs to be investigated because of its essential role in decision making processes. Among available FDD methods, data driven approaches are currently receiving increasing attention because of their relative simplicity in implementation. Regardless of FDD types, one of the main traits of reliable FDD systems is their ability to be updated while new conditions that were not considered at their initial training appear in the process. These new conditions would emerge either gradually or abruptly, but they have the same level of importance as in both cases they lead to FDD poor performance.

For addressing updating tasks, some methods have been proposed, but mainly not in research area of chemical engineering. They could be categorized to those that are dedicated to managing Concept Drift (CD) (that appear gradually), and those that deal with novel classes (that appear abruptly). The available methods, mainly, in addition to the lack of clear strategies for updating, suffer from performance weaknesses and inefficient required time of training, as reported.

Accordingly, this thesis is mainly dedicated to data driven FDD updating in chemical processes. The proposed schemes for handling novel classes of faults are based on unsupervised methods, while for coping with CD both supervised and unsupervised updating frameworks have been investigated. Furthermore, for enhancing the functionality of FDD systems, some major methods of data processing, including imputation of missing values, feature selection, and feature extension have been investigated.

The suggested algorithms and frameworks for FDD updating have been evaluated through different benchmarks and scenarios. As a part of the results, the suggested algorithms for supervised handling CD surpass the performance of the traditional incremental learning in regard to MGM score (defined dimensionless score based on weighted F1 score and training time) even up to 50% improvement. This improvement is achieved by proposed algorithms that detect and forget redundant information as well as properly adjusting the data window for timely updating and retraining the fault detection system. Moreover, the proposed unsupervised FDD updating framework for dealing with novel faults in static and dynamic process conditions achieves up to

90% in terms of the NPP score (defined dimensionless score based on number of the correct predicted class of samples). This result relies on an innovative framework that is able to assign samples either to new classes or to available classes by exploiting one class classification techniques and clustering approaches.

Resumen

Los procesos industriales modernos son cada vez más complejos y, en consecuencia, su control se ha convertido en una tarea desafiante. La detección y el diagnóstico de fallos (FDD), como un elemento clave de la supervisión del proceso, deben ser investigados debido a su papel esencial en los procesos de toma de decisiones. Entre los métodos disponibles de FDD, los enfoques basados en datos están recibiendo una atención creciente debido a su relativa simplicidad en la implementación. Independientemente de los tipos de FDD, una de las principales características de los sistemas FDD confiables es su capacidad de actualización, mientras que las nuevas condiciones que no fueron consideradas en su entrenamiento inicial, ahora aparecen en el proceso. Estas nuevas condiciones pueden surgir de forma gradual o abrupta, pero tienen el mismo nivel de importancia ya que en ambos casos conducen al bajo rendimiento de FDD.

Para abordar las tareas de actualización, se han propuesto algunos métodos, pero no mayoritariamente en el área de investigación de la ingeniería química. Podrían ser categorizados en los que están dedicados a manejar Concept Drift (CD) (que aparecen gradualmente), y a los que tratan con clases nuevas (que aparecen abruptamente). Los métodos disponibles, además de la falta de estrategias claras para la actualización, sufren debilidades en su funcionamiento y de un tiempo de capacitación ineficiente, como se ha referenciado.

En consecuencia, esta tesis está dedicada principalmente a la actualización de FDD impulsada por datos en procesos químicos. Los esquemas propuestos para manejar nuevas clases de fallos se basan en métodos no supervisados, mientras que para hacer frente a la CD se han investigado los marcos de actualización supervisados y no supervisados. Además, para mejorar la funcionalidad de los sistemas FDD, se han investigado algunos de los principales métodos de procesamiento de datos, incluida la imputación de valores perdidos, la selección de características y la extensión de características.

Los algoritmos y marcos sugeridos para la actualización de FDD han sido evaluados a través de diferentes puntos de referencia y escenarios. Como parte de los resultados, los algoritmos sugeridos para el CD de manejo supervisado superan el rendimiento del aprendizaje incremental tradicional con respecto al puntaje MGM (puntuación adimensional definida basada en el puntaje F1 ponderado y el tiempo de entrenamiento) hasta en un 50% de mejora. Esta mejora se logra mediante los algoritmos propuestos que detectan y olvidan la información redundante, así como ajustan correctamente la ventana de datos para la actualización oportuna y el reciclaje del sistema de detección de fallas. Además, el marco de actualización FDD no supervisado propuesto para tratar fallas nuevas en condiciones de proceso estáticas y dinámicas logra hasta 90% en términos de la puntuación de NPP (puntuación adimensional definida basada en el número de la clase de muestras correcta predicha). Este resultado se basa en un marco innovador que puede asignar muestras a clases nuevas o a clases disponibles explotando una clase de técnicas de clasificación y enfoques de agrupamiento.

Table of Contents

List of Tables

List of Figures

Nomenclature

Acronyms	
AC	Auxiliary Classifier
ACr	Adjusted Cardinality
ANN	Artificial Neural Networks
AV	Adjusted Value
CA	Clustering Accuracy
Card	Cardinality
CAS	Classification Assessment Scores
CD	Concept Drift
cda	Concept Drift Amount
CS	Validity index presented by Chou and Su
CSTR	Continuous Stirred Tank Reactor
DB	Validity index presented by Davies and Bouldin
DT	Decision Trees
DW	Dynamic Window
F	False
FD	Fault Detection
FDD	Fault Detection and Diagnosis
FN	False Negative
FP	False Positive
F1	Fault 1
F2	Fault 2
F3	Fault 3
FSM	Feature Selection Method
FW	Fixed sliding Window
IL	Incremental Learning
ILDW	Incremental Learning Dynamic Window

IPP	Individual Prediction Performance
GNB	Gaussian Naïve Bayes
GA	Genetic Algorithm
KKT	Karush-Kuhn-Tucker Conditions
MAR	Missing At Random
MC	Main Classifier
MCAR	Missing Completely At Random
NIL	Non-Incremental Learning
NMAR	Not Missing At Random
ND	Novelty Detection
NPP	Net Prediction Performance
Nr	Normal
NSV	Number of Support Vectors
NUC	Non-Updated Classifier
OCC	One Class Classifiers
OCS	One Class Support Vector Machines
OCS_Nr	One Class Support vector machine trained with normal class
OCS_F1	One Class Support vector machine trained with F1 class
OCS_F2	One Class Support vector machine trained with F2 class
OCS_F3	One Class Support vector machine trained with F3 class
OK	Ordinary Kriging
OM	Online Monitoring module
OMU	Offline Model Updating
OVPP	Overall Validation Prediction Performance
QP	Quadratic Problem
RBF	Radial Basis Function
RMSE	Root Mean Square Error
RP	Removing Percentage
RV	Retraining Value
SV	Support Vector

SVM	Support Vector Machines
T	True
TE	Tennessee Eastman
Tec	Technique
TI	Time Interval
TP	True Positive
Tr	Threshold
TN	True Negative
UT	Updating Threshold
UTP	Updating Threshold Parameter
VPP	Validation Prediction Performance
WMC	Wrapper Method Classifiers
Indices, Parameters and Values	
b	bias
b^*	Optimum value of bias
c=0,1,2,….C	Classes (c=0: normal class, c=1: Fault 1 (F1), etc.)
$\overline{CAS}$	Mean of the CAS
$\overline{CPU}$	Mean of the CPU time of training
cda_{kc}	Concept drift amount in kth dataset for class c
d_j	Binary decision for selecting feature $\boldsymbol{x}_j$
er_{kc}	Calculated error for class c in the kth dataset
$\overline{F1}$	Mean of the F1 score
h_1, h_2, h_3	Tank levels in three tanks case study
i=1,2,…….I	Index of sample
j=1,2,…….J	Index of feature
k=1,2,…...K	Index of dataset
k*	Index of reference dataset
m, n	Arbitrary values
$\overline{NSV}$	Mean of the number of the support vectors
P_{kc}	Precision of class c in kth dataset

$rmse_{Tec,j}$	RMSE between $\widehat{m}_{Tec,j}$ and $\mathring{m}_j$
w^*	Optimum value of weight vector
x_{kij}	Value of the jth feature of the ith observation in the X_k
x_{k^*ij}	Value of the jth feature of the ith observation in the X_{k^*}
$\hat{x}_{Tec,ij}$	Estimated value for x_{ij} with specific technique
y_i	Label of ith sample
λ_i	Lagrangian multipliers
ω	Weight coefficient
β	Cluster threshold parameter
Matrices, Vectors and Sets	
A	Union of the labeled subsets
A_c	Set of cluster c
A_c^u	Updated set of cluster c
A_k	Set of all the provided augmented datasets with the kth dataset and reference dataset
A_c^V	Validation subset of cluster c
A_c^T	Training subset of cluster c
AG_{kc}	Augmented dataset of class c in the kth dataset
$AG1_{kc}$	Subset of AG_{kc} made by NACF
$AG2_{kc}$	Subset of AG_{kc} made by NACF
B^*	Unlabeled dataset in online monitoring module
B	Unlabeled dataset
B^{*c}	Updated unlabeled dataset in online monitoring module
CDA_k	Set of cda_{kc} for all the classes in the kth dataset
$\boldsymbol{D}$	Vector of all d_j
E	Error dataset
E_k	Set of all $er_{kc}$for all the classes in the kth dataset
$\boldsymbol{e}_j$	jth error feature
WI_k	Window of Dataset at kth time interval

X^*	New dataset
X_{IJ}	Dataset with I rows and J features
X_k	kth dataset
X_{k^*}	Reference dataset
$\widehat{X}_{Tec}$	Recuperated dataset
X'_k	Filtered X_k
$\boldsymbol{x}_i$	Vector of the ith sample
$\boldsymbol{x}_i^*$	Vector of the new sample
$\boldsymbol{xe}_j$	Vector of the jth feature of XE
$\boldsymbol{x}_{ki}$	Vector of the ith sample in the X_k
$\boldsymbol{x}_{k^*i}$	Vector of the ith sample in the X_{k^*}
XE	Union of the X and E
$\boldsymbol{x}_r$ and $\boldsymbol{x}_s$	Random support vectors
Γ_{kc}	Subset of the X_k containing all the samples of the class c
Γ_{k^*c}	Subset of the X_{k^*} containing all the samples of the class c
Γ'_{kc}	Filtered Γ_{kc}
Functions and Models	
$\bar{C}$	Labeling function for clustering
Cl	Clustering Function
H	Net prediction function
h_c	Prediction function of OCC trained with cluster c
L	Label function for providing true class of samples
$\mathring{m}_j$	Function for ideal values for jth feature
$\widehat{m}_{Tec,j}$	Provided model for feature j with specific technique
$NACF$	Non-automatic clustering function
P	Class prediction function

Chapter 1: Introduction

1.1 Outline

Chemical processes must be monitored continuously for producing final chemical products with the desirable quality of the markets as well as safety purposes. Process monitoring systems assist plant operators for observing faults in the early time rather than uncover poor quality of the chemical products. Without process monitoring a manufacturing process cannot be successful. The process monitoring could be on any measurable feature in order to have timely reactions for fixing and removing the abnormality causes. This could prevent not only economic and environmental damages, but also provide safe work environment for the operators. Weighing up the importance and complexity of the process that is aimed to be controlled, type of the process monitoring approaches would be altered. Process monitoring could range from relatively simple statistical methods to complex sophisticated systems of advanced chemical plants. Considering the importance of the process monitoring systems, they have been studied profoundly. It is an interesting research topic in chemical, mechanical and electrical engineering as well as statistics, mathematics and data science; consequently, various algorithms and approaches have been investigated and presented. Process monitoring systems have crucial elements that interact with each other, and they may work simultaneously or sequentially. These main elements are Fault Detection and Diagnosis (FDD) systems, optimization system, and control system. Each of these elements is essential, and without them complete process monitoring task cannot be expected.

Based on the FDD attributes, some main categories of them can be found, including data driven or model based approaches, and supervised or unsupervised methods; each of them has their relative strengths and weaknesses. Regardless of the FDD type, there are some desirable characteristics that FDD systems should ideally possess to be infallible [1]; early detection and diagnosis; isolability that is ability of the diagnostic system to discriminate between different failures; robustness to various noise and uncertainties; novelty identifiability; multiple fault identifiability; explanation facility that is providing explanations on how the fault originated and propagated to the current situation; adaptability; and reasonable computational requirement.

None of the FDD methods individually has all the desirable features that one complete method must hold. In order to cover the weaknesses of individual methods and for having a robust FDD system, combination of the methods would be a suitable avenue. In [2], advantages and disadvantages of different FDD methods are discussed, and it is concluded that hybrid systems could be a practical way for improving FDD performance. Even the hybrid systems may have some weaknesses in several aspects, but at least the optimum methods could be designated.

FDD data driven approaches could be divided into two main categories, including supervised, and unsupervised methods [3] [4]. The combination and exploitation of both approaches in semi supervised methods have been investigated [5] [6]. The selection of each approach depends on the prior knowledge about the possible faults and their patterns in the process measurements; assumption of the supervised learning approaches is that the process data could be correctly labeled [7] [8]. However, labeling may be a quite difficult, costly, or even impossible task in many practical situations. The most popular supervised data driven methods are based on classification techniques that detect and classify normal or faulty conditions based on pattern recognition principles. On the other hand, unsupervised approaches are usually applied whereas there is no prior information about the fault types in the process output data (or it is difficult to obtain). In unsupervised approaches, such as clustering methods, required knowledge is provided from the available/historical data.

FDD systems must possess an ability to handle a massive volume of data, and two main strategies could be taken to fit FDD to this attribute. First, applying several parallel FDD algorithms that work at the same time either doing the same tasks or different. Second, applying the incremental (gradual) learning algorithms that learn from data step by step [9]. It is worth mentioning the combination of these two strategies could be a solution, too.

Because of the possible time constraints, and samples availability the learning algorithms may be divided to four general groups, including batch mode learning, Incremental Learning (IL), online learning, and any time learning [9]. In batch mode learning, all the samples in training dataset are available, and they are executed for model learning at once. In IL, receiving samples are integrated for training without the need for performing learning step from the scratch. In most of the IL algorithms, samples are read only once, and this assists reducing required computational time. In online learning, samples arrive continuously, and integration of them and learning step must be done with very low latency and computational time [10] [11]. The first and most important advantages of working online could be early detection in order to prevent possible malfunctions that could be averted by doing timely appropriate reactions. In anytime learning, algorithm maximizes or minimizes evaluation criterion to enhance the quality of the model until an interruption. The conditions on which the FDD data driven methods are selected could have a wide range, but they basically depend on the state of the process that is batch or continuous, and scale of the plants [9].

Drifting of samples in the machine learning terminology is summarized in terms of Concept Drift (CD) [12]. CD is a kind of drift that is gradual, in [13], CD is considered as a change in data distribution that can cause predictive performance of the classifiers to degrade over time. For handling CD several approaches such as incremental learning, decremental learning, adaptive algorithm etc. have been suggested [14] [15]. On the other hand, there are conditions/states that appear abruptly. Novelty Detection (ND) methods have been proposed [16] [17] for dealing with new process conditions that appear abruptly.

As it mentioned, adaptability or ability of updating is a vital characteristic of reliable FDD systems, however, it is hardly addressed in academic research [7]. From industrial perspective, adaptability is quite important because process plants usually do not remain invariant. The process changes could have several causes such as adjusting operating policy, appearance of new/novel conditions/classes, etc. In the process that novel classes appear, adaptability is addressed by ability to use information of novel samples for retraining [16]. The FDD systems must be well adapted to these changes with minimal effort [7] [8]; otherwise, they will be invalided after a period in operation [18]. Therefore, in this thesis, FDD updating in supervised and unsupervised styles for dealing with CD and those classes that appear abruptly are investigated and discussed.

In chapter 1, exploited methods and tools of FDD as well as applied benchmarks are reviewed. Chapter 2 is devoted to data processing, which has proved advantageous for enhancing classification and FDD performance. Data processing has an important role in FDD performance because datasets with noise, outlier, missing values and redundant information hinder efficient process monitoring [19]. Thus, application of regression approaches using machine learning techniques to regress the missing, noisy and outlier values have been studied. In addition, feature selection and feature extension methods for removing redundant features and providing new features, respectively, have been investigated.

Chapter 3 is about supervised updating of Fault Detection (FD) systems. In the first section, Dynamic Window (DW) algorithm for handling CD in the process is presented. DW exploits the last available samples to detect and forget redundant samples by using Auxiliary Classifier (AC). Moreover, in order to exploit advantageous of DW and IL, Incremental Learning Dynamic Window (ILDW) algorithm is proposed. In the second section, a framework is presented that implicitly traces changes in CD amount with clustering approaches and devised index, which is in accordance with the precision definition.

Chapter 4 is dedicated to unsupervised FDD updating. In the first section, a framework for FDD updating while new conditions/faults abruptly appear in the process is presented. The hybrid FDD updating framework consists of automatic clustering and One Class Classifiers (OCC) while FDD performance is enhanced by an observer. In the second section, for unsupervised handling CD in the process, unsupervised ILDW algorithm is proposed; within this ILDW scheme, the needed labels for FD updating are predicted and then filtered.

Finally, Chapter 5 is about the main conclusions and contributions of the thesis as well as future works and published contributions.

1.2 Fault Detection and Diagnosis

The term of fault is defined as a deviation from an acceptable range of an observed variable or a calculated parameter associated with a process [20]. Thus, a fault is defined as a process abnormality or symptom, such as high temperature in a reactor or low product quality and so on. The underlying cause(s) of this abnormality, such as a failure in a coolant pump or a controller, is (are) called the basic event(s) or the root cause(s) [20]. Four main procedures of process monitoring could be considered as following: Fault Detection (FD), fault identification, fault diagnosis, and process recovery [17].

FD is the determination of whether a fault has occurred, and data processing could efficiently assist FD system for detecting faults. Hotellings T^2 statistic and Q statistic (also called squared prediction errors) are common data processing approaches that are calculated from the principal component analysis. These two methods do FD based on monitoring variables, and not with classification task. FD of largescale processes because of the availability of enormous amount of features and samples would be a challenging task. This problem is addressed in [21] for conducting dynamic largescale process, and also similar studies that deal with big data could be found in [22] [23].

Fault diagnosis is to determine which fault occurred while fault identification is to identify the features most relevant to the fault. In [24], more specifically fault diagnosis is defined as determination of type, location, magnitude, and time of the fault. Fault isolation and fault diagnosis could be employed interchangeably; If the classification task is done with historical data, the method is fault diagnosis, otherwise, it is fault isolation [25].

1.3 Methods and Tools

In following sections and subsections, applied techniques and tools that are exploited in the next chapters are described, including classification, clustering and regression methods. All the calculation and reported CPU times are done with MSI-notebook Intel (R) Core (TM) i7-4710HQ CPU @ 2.5GHz 2.5 GHz Ram-16 GB.

1.3.1 Classification Methods

Diagnosis can be considered as classification problem [2]. The assignment of a sample to one of the available categories or classes is the problem addressed by pattern classification theory [26]. Commonly, the most popular supervised data driven FDD approaches are based on classification methods. Artificial Neural Networks (ANN), Support Vector Machines (SVM), Decision Trees (DT), and Gaussian Naïve Bayes (GNB), are among most common classification methods.

Without requiring any explicit mathematical models, classifiers could be trained based on pattern recognition principles by historical data, including information about normal and different faulty situations [27]. The learning process, by optimization or adjustment of the parameters, enables these classifiers to extract knowledge from data. Then, the trained classifiers can be used for process supervision in order to detect and diagnose possible faults from the process outputs measurements [28]. Classifiers would be categorized into two groups: multiclass classifiers and

One Class Classifiers (OCC). Multiclass classifiers for training need at least two labeled classes of samples while OCCs need only one class for training.

1.3.1.1 Multiclass Classifiers

Classifiers such as SVM, basically, are designed for classifying two classes (binary classification). With the multiple classes two main strategies for training the binary classifiers could be hired; one-versus-one and one-versus-all. With the C classes in one-versus-one strategy, $C\,(C-1)/2$ classifiers must be trained whereas in one-versus-all C classifiers are required. Classification with one-versus-one strategy will be difficult, while number of the classes increase. This is mainly because of the number of the parameters that must be tuned [29]. In one-versus-all strategy, one of the classes is considered as positive and the rest of the classes are counted as negative. Binary classifiers with one-versus-all strategy, sometimes in the literature, are known as One Class Classifiers (OCC) [30]. Instead of creating several binary classifiers, a more natural way is to distinguish all the classes in one single optimization processing. Many algorithms are suggested for handling multiple classes in one step and only by one multiclass classifier [31] [32] [33]. Depending on the FDD purposes, any combination of the classifiers may be applied.

1.3.1.1.1 Support Vector Machines (SVM)

SVM is a method developed by Vapnik and co-workers [26], while Cauwenberghs et al. [34] proposed an algorithm for implementing SVM in online way for the first time. SVM have been applied for classification in different research areas [23] [27] [28]. SVM works based on maximizing the margin between the training patterns and the decision boundary [6]. Considering set of training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ in which $\boldsymbol{x}_i$ is the vector and $y_i \in \{+1, -1\}$, the classifier is based on two principal elements: a weight vector (function) "w" and bias "b" which is the distance of the hyperplane to the origin. Figure 1. 1 shows the simplest form of "b" and "w".

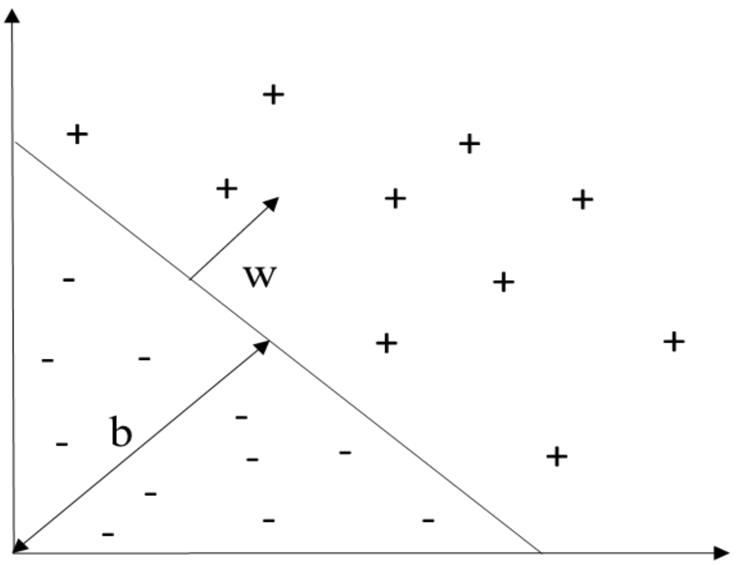

Figure 1. 1. Parameters of the SVM.

The result of the classification task for new samples, $\boldsymbol{x}_i^*$, will be $f(\boldsymbol{x}_i^*) = +1$, when $f\ (\boldsymbol{x}_i^*) = (\boldsymbol{x}_i^*, w) + b \geq 0$, otherwise $f(\boldsymbol{x}_i^*) = -1$. Based on the definition positive and negative samples close to the hyperplane are called Support Vectors (SV). Finding out (w, b) could be solved as a convex Quadratic Problem (QP) [35] with the unique solution, Equation (1. 1):

$$Minimize\ ||w||\ \ s.t.\ y_i(\langle w, \boldsymbol{x}_i \rangle + b) \geq 1 \tag{1. 1}$$

The optimum amount of "w" and "b" are "w^*", "b^*" respectively, and are calculated based on Equation (1. 2), and Equation (1. 3):

$$w^* = \sum_{i=1}^{I} \lambda_{\ i} y_i \boldsymbol{x}_i \tag{1. 2}$$

$$b^* = -\frac{1}{2} \langle w^*, \boldsymbol{x}_r, \boldsymbol{x}_s \rangle \tag{1. 3}$$

Considering $\boldsymbol{x}_r$ and $\boldsymbol{x}_s$ are random SVs and $\lambda_{\ i}$ is Lagrangian multipliers. Accordingly, $f(x)$ is calculated with Equation (1. 4):

$$f(\boldsymbol{x}_i^*) = sgn(\langle w^*, \boldsymbol{x}_i^* \rangle + b^*) \tag{1. 4}$$

Karush-Kuhn-Tucker (KKT) conditions, Equation (1. 5), are necessary and sufficient conditions for an optimal point of a QP.

$$\begin{cases} \lambda_i = 0 \Leftrightarrow y_i f(\boldsymbol{x}_i) \geq 1 \\ 0 < \lambda_i < Ci \Leftrightarrow y_i f(\boldsymbol{x}_i) = 1 \\ \lambda_i = Ci \Leftrightarrow y_i f(\boldsymbol{x}_i) \leq 1 \end{cases} \tag{1. 5}$$

In which Ci is a parameter that trades off between wide margins with a small number of margin failures.

SVM could be applied both in online and in offline styles. The offline application is common, however for online application recently promising algorithms are proposed that have proved advantageous in accuracy and computational time [36] [37]. For applying SVM two issues must be considered; selecting kernel function and finding its best parameters, and selecting optimal features of training dataset [38]. The kernel function converts a nonlinear classification problem into a linear one in a high dimensional feature space [39] [40]. Some applicable kernel functions are polynomial, Radial Basis Function (RBF) and sigmoid kernel. The kernel parameters should be properly tuned in order to improve the classification performance. RBF is among the most applicable kernel functions, and for adjusting its parameter a common approach in the literature is the Grid algorithm [38].

1.3.1.1.2 Decision Trees (DT)

Ruled based methods consist of an antecedent part and a consequence part [5]. DT is a recursive approach for extracting diverse classification rules [41]. Low latency for prediction makes them a suitable choice for processing large amounts of data. Standard DT structures are CART, ID3, and C4.5 [42]. The advantage of CART and C4.5 algorithms is that they can handle multiclass cases [43]. C4.5 does not make strong distributional assumptions about the data while CART does. The C4.5 makes decision trees that contain sets of ordered rules based on information entropy [8] [9].

1.3.1.1.3 Gaussian Naïve Bayes (GNB)

Naïve bayes classifier is based on strong independence assumptions between the features. It is highly competitive while few learning examples are available. For Gaussian Naïve Bayes (GNB) classifier the Gaussian distribution is assumed. In naïve bayes classifier because of the independency assumption, the classification process is very efficient in training and prediction time [44]. In some studies, naïve bayes classifiers because of its advantageous are implemented with other learning algorithms [45].

1.3.1.2 One Class Classifiers (OCC)

The OCC algorithms determine the smallest hypersphere enclosing the training samples [46], and thus all samples lying outside would be classified as abnormal/novel. One class classification techniques usually are applied for detecting novel classes [47]. A particular advantage of the OCC algorithms is that in training phase only positive samples are required. OCC techniques are useful while positive samples are rich, and negative samples are either rare or their structures are unclear [48]. One Class Support vector machines (OCS) is an applicable algorithm of OCC; there are several kernel methods, such as RBF, that could be applied to it. Figure 1. 2 presents a simplified two-dimensional feature space for OCS with RBF kernel.

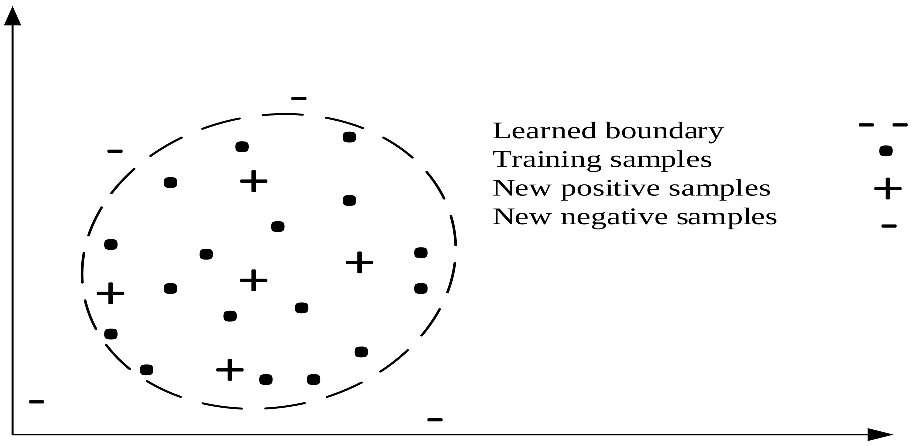

Figure 1. 2. Simplified OCS algorithm with RBF kernel for two-dimensional feature space.

1.3.2 Clustering Methods

Clustering is an unsupervised task that consists in partitioning an unlabeled dataset into groups of similar objects based on extracted knowledge from the data. Each created group is called a "cluster" [49] [50], and the notion of similarity depends on the purpose of the clustering [51]. Among different methods of clustering, K-means, hierarchical clustering, expectation maximization, and density based clustering are considered among the most practical methods because of their flexibility and applicability in different areas [52].

In the literature, clustering methods are divided into different groups. Based on the way of partitioning, they could be divided into hierarchical and partitional methods. Hierarchical methods rely on object similarity while partitional methods rely on the clustering criteria optimization [50]. From another perspective, clustering approaches could be divided into automatic and non-automatic techniques. While for non-automatic clustering number of the clusters must be defined, in automatic clustering optimal number of the clusters are determined without any prior knowledge about the data. In other words, automatic clustering algorithms are able to determine the optimal number of clusters, and simultaneously assigning the data to these clusters.

1.3.2.1 Non-Automatic Clustering

One of the most applicable non-automatic clustering methods is K-means. It is an unsupervised clustering technique that needs number of clusters for partitioning task. It is based on dividing a dataset into the specific number of clusters by minimizing some metrics relative to the cluster centroids. The cluster shapes are affected by the selected metric for minimization and distance definition [53]. For partitioning dataset X to k clusters, and in order to minimize the sum-of-squares criterion, k centroids must be determined. Centroids are randomly selected, and then samples are assigned to the nearest clusters. By repetition of selecting random centroids and assigning samples to the nearest cluster, center and samples of each cluster are determined [49] [53] [54].

1.3.2.2 Automatic Clustering

It refers to any approach for automatically determining the optimal number of clusters in a dataset. In automatic clustering methods, validity indices refer to devised statistical–mathematical functions; these indices are applied in order to judge the quality of partitioning task. Validity indices are based on two concepts: cohesion and separation. Cohesion is aimed at keeping the patterns/objects of one cluster similar, as much as possible, while separation is aimed at keeping patterns/objects of one cluster different, as much as possible, from the patterns/objects of other clusters [49]. Among various validity indices, two indices, CS and DB, have been implemented in many studies because of their proven capabilities. These two indices seek clusters that have minimum within-cluster scatter and maximum between-cluster separation [55]; the optimum values of these indices must be found with the optimization techniques.

1.3.2.2.1 DB

Minimizing parameter of the validity index presented by Davies and Bouldin (DB) leads to find out natural clusters of datasets [56], Equation (1. 6) to Equation (1. 9) :

For DB, $\bar{R}$ is defined as:

$$\bar{R} \equiv \frac{1}{N} = \sum_{i=1}^{N} R_i \tag{1. 6}$$

Where $R_i \equiv maximum\ of\ R_{ij}\ i \neq j$

R_{ij} is a function of S_i, $S_j$and M_{ij}:

$$R_{ij} \equiv \frac{S_i + S_j}{M_{ij}} \tag{1. 7}$$

S_i and M_{ij} are defined with the Equation (1. 8) and Equation (1. 9) :

$$S_i = \left\{ \frac{1}{T_i} \sum_{j=1}^{T_i} \left| X_j - A_i \right|^q \right\}^{\frac{1}{q}} \qquad 1.8$$

By definition, S_i is the q^{th} root of the q^{th} moment of the samples in cluster i about their mean. T_i is number of the samples in cluster i, and A_i is the centroid of cluster i.

$$M_{ij} = \left\{ \sum_{k=1}^{N} \left| a_{ki} - a_{kj} \right|^p \right\}^{\frac{1}{p}} \qquad 1.9$$

a_{ki} is the kth component of the n-dimensional vector a_i that is the centroid of cluster i. q and p could be selected independently [56]. In the present thesis, they are considered two.

1.3.2.2.2 CS

Chou and Su [55] proposed validity measure, CS; similar to DB index, by minimizing value of the CS index, optimal clusters would be founded.

Consider $Z = \{z_j;\ j = 1, 2, \ldots, N\}$ in which N is number of the clusters obtained by recalculating cluster centers, Equation (1. 10):

$$v_i = \frac{1}{|A_i|} \sum_{z_j \in A_i} x_j \qquad 1.10$$

Where A_i is the set of the i^{th} cluster and $|A_i|$ is the number of samples in A_i. Accordingly, CS is calculated with Equation (1. 11):

$$CS(c) = \frac{\sum_{i=1}^{c} \left\{ \frac{1}{|A_i|} \sum_{z_j \in A_i} \max_{x_k \in A_i} \{ d(z_j, z_k) \} \right\}}{\sum_{i=1}^{c} \left\{ \min_{j \in c, j \neq i} \{ d(v_i, v_j) \} \right\}} \qquad 1.11$$

d is a distance function that could be selected. In this thesis, d function is selected to be the Euclidean distance.

1.3.3 Regression Methods

The regression methods, in this thesis, are applied in order to approximate the feature behavior, however they could be applied as observers, too. The models are used to predict the missing values together with the real values of the noise and outliers measurements. Among regression models, ANN due to its universal approximation and ability to model nonlinear systems has been widely applied. Nevertheless, it shows some drawbacks as the curse of dimensionality, and the difficulty of configuring the network structure (the number of hidden layers and the number of neurons in each layer). Ordinary Kriging (OK) and Multivariate Dynamic Kriging (MDK) models have been used in many surrogate based optimization studies in the chemical process engineering area, showing high accuracy and capability to model complex highly nonlinear systems with a relatively small number of training data.

1.3.3.1 Ordinary Kriging (OK)

Kriging models are originated in the areas of mining and geostatistics that involve spatially and temporally correlated data. Their unique characteristic stems from their ability to combine global and local modeling. Having a set of input-output training data $(\boldsymbol{x}_i, y_i), i = 1,2, .. I$, the kriging assumes a general predictor $y(x) = P(x) + Q(x)$, which is composed of a polynomial function of interest $P(x)$, that provides the global behavior or the main trend of the system to be approximated, and in many cases $P(x)$ is taken as a constant value. The second term $Q(x) = \sigma^2 \psi$, is a realization of a stochastic process, with a mean of zero value, variance σ^2, and a correlation function $\psi(x_i, x_j) = exp\left(-\sum_{l=1}^{k} \omega_l \left|x_{i,l-} x_{j,l-}\right|^2\right) + \delta_{i,j}\, \xi$ between any two sample points of the training dataset. ω_l is the correlation parameter of the input variable, δ_{ij} is the Kronecker delta, and ξ is the regression constant that enables the kriging to smooth or regress noisy data. The final kriging predictor can be expressed as Equation (1. 12), where γ is the column vector of the correlations between the samples to be predicted x^P and the training data points.

$$y(x^p) = b + \gamma^T \psi^{-1}(Y - 1b) \qquad 1.\ 12$$

1.3.3.2 Multivariate Dynamic Kriging (MDK)

MDK is based on the construction and training of OK models that are trained to capture the incremental evolution of the system, i.e. the system future state/output variables over one time step. In more details, each OK model is trained to approximate the mapping between the future value of one state/output variable at the next time step as a function of the system previous state and control variable values $[\, X_t, X_{t-1}\, , \dots\, X_{t-l},\; U_t, U_{t-1}\, , \dots\, U_{t-l}]$ considering a specific time lag or delay L. This is given by Equation (1. 13):

$$\left.\begin{aligned} \hat{x}_1(t+1) &= f_1\big[\hat{X}(t), .. \,\hat{X}(t-L), U(t), .. U(t-L)\big] \\ \hat{x}_2(t+1) &= f_2\big[\hat{X}(t), .. \,\hat{X}(t-L),\, U(t), .. U(t-L)\big] \\ &\dots\dots \\ \hat{x}_i(t+1) &= f_i\big[\hat{X}(t), .. \,\hat{X}(t-L),\, U(t), .. U(t-L)\big] \\ &\dots\dots\dots.. \\ \hat{x}_{kx}(t+1) &= f_{kx}\big[\hat{X}(t), .. \,\hat{X}(t-L),\, U(t), .. U(t-L)\big] \end{aligned}\right\} \qquad 1.\,13$$

Where $U(t) \in R^{k_u}$ represents the control/input variables, and $X \in R^{k_x}$ corresponds to the state/output, which is recorded at discrete time instances of equal intervals Δt between them. k_u and k_x are the number of control and state variables, respectively. These sets of single step emulators are also considered as nonlinear autoregressive models with exogenous inputs, which are able to predict the system outputs over one time step ahead. Additionally, they can be also used via recursive interpolation to predict the outputs over several time steps. Thus, at each time step, the predicted values of the state variable are fed back to the model representing its input for the next time step estimation, together with the new value of the control variables. More details about the dynamic kriging models, multivariate dynamic prediction via recursive interpolation and their applications to other case studies could be found in [57] [58].

1.3.3.3 Artificial Neural Networks (ANN)

ANN is a well-known efficient method that is used widely for modeling nonlinear system. Feedforward ANN are frequently used in engineering applications for system modeling and identification. In this thesis, Matlab ANN toolbox and the function “feedforwardnet” have been used to create a feed forward ANN. The number of neurons and layers (two hidden layers of nine and three neurons), the training algorithm (“train”), and transfer functions (“default”) were selected to balance simplicity and accuracy.

1.3.3.4 Polynomial Regression (PR)

A widely used approximation method is the polynomial regression. Considering a set of input-output training data, the predictor/estimator is assumed as a polynomial function of a certain degree. The polynomial coefficients (model parameters) are estimated through the least square fitting in which the sum of square errors (between the data and the model predictions) are minimized in order to find the best parameters or coefficients of this polynomial/functional shape. In the thesis, the Matlab function “polyfit” is used for fitting a polynomial of the 25th degree to the data.

1.3.4 Optimization Methods

For industrial processes, optimization is one of the key action both for operation and process monitoring tasks [8]. Optimization problems could appear in different aspects and steps of FDD as single or multiple objectives. For optimization task with the multiple objectives, maximization of one or more objectives conflict with minimization of other objectives thus multiple solutions may possible, each is better than the rest in at least one of the objectives. For FDD task, the optimization with the multiple objectives could appear for selecting those algorithms that must be applied at the same time in a framework; some algorithms have advantages in the accuracy, and other have advantages in the computational time.

Optimization is the core of several supervised and unsupervised FDD algorithms therefore, regarding type of the exploited algorithms, FDD applications could be considered as an optimization problem [59] [29]. In SVM algorithm the optimization problem is to find out the optimum hyperplane that separates the classes [60]. In OCC algorithms separation of positive samples from negative samples is done with optimization task [61]. Furthermore, clustering task could be seen as a well-defined optimization problem [7].

For process monitoring a wide range of the optimization, techniques have been applied. In [62], particle swarm algorithm that is a heuristic optimization method is applied for improving performance of the independent components analysis. Derivative free optimization techniques have been widely used in many areas, mainly, due to their abilities to prevent challenges associated to the use of derivative based techniques [63].

In this thesis, in order to explore optimum amounts of the clustering validity indices and selecting the optimum number of features (for data processing) optimization techniques are applied. Among available techniques, the Genetic Algorithm (GA) is one of the best candidates. GA has shown very high capabilities in accuracy and robustness among a wide range of derivative free optimization techniques (such as; swarm optimization, direct search techniques etc.). GA has been applied to a wide range of the engineering applications involving different types of optimization problems [64], and including linear and nonlinear objectives. The GA is stochastic search procedure whose search method mimics the genetic evolution of a species. The GA search mechanism starts with an initial set of potential solutions that are randomly selected in most cases. In the present thesis, the GA of the MATLAB optimization toolbox is used with its default configurations.

1.3.5 Performance Indices

Performance indices are applied for quantitative assessment of the different FDD approaches. Types of the performance indices that must be applied depend on the application of FDD either online for each sample or offline for each batch of samples. Furthermore, they rely on whether FDD type is supervised, unsupervised or semi supervised [9] [65].

Classification Assessment Scores (CAS) are those that applied for validating supervised classifiers. In [66], the CAS for diagnosis is defined as the ability of correctly identifying the root causes of abnormal behaviors. In [67], CAS is defined as a fraction of correct predicted samples in the test dataset.

In the simplified condition with only two classes, the classifier could assign each sample to the set $\{p, n\}$ in which p stands for positive class and n stands for negative class, accordingly, four possibilities may happen. If the sample is truly positive and it is assigned to a positive class, then it is a True Positive (TP), but if it is assigned to a negative class, it is a False Negative (FN). If the sample is truly negative and it is classified as negative, it is a True Negative (TN), but if it is classified as positive, it is a False Positive (FP). Figure 1. 3 shows the confusion matrix of this classification [68].

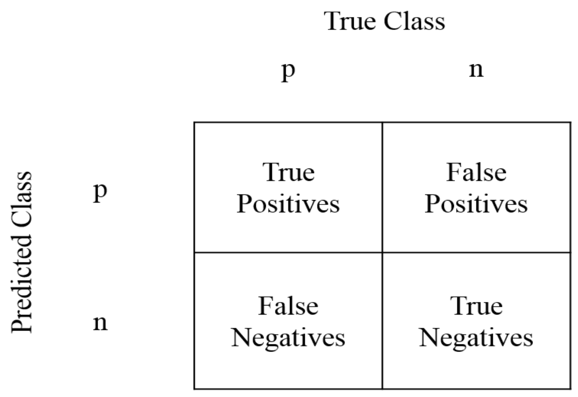

Figure 1. 3. Confusion matrix for classification of two classes.

Therefore, precision, recall, and accuracy could be defined with Equation (1. 14) to Equation (1. 16):

$$precision = \frac{TP}{TP + FP} \tag{1. 14}$$

$$recall = \frac{TP}{TP + FN} \tag{1. 15}$$

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{1. 16}$$

Precision measures the proportion of assigned samples to the positive class that are truly positives while recall measures the proportion of correctly classified samples of positive class [69]. Balanced accuracy index considers number of the samples in each class for calculating the accuracy. The advantage of this index is that it prevents ignorance of the classifier poor performance for those classes that has few samples [9] [65].

F1 score is a measure widely used to compare FDD systems. It is calculated as a weighted average of precision and recall as indicated in Equation (1. 17):

$$\text{F1 } score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1. 17}$$

The F1 score ranges from 0.0 (worst value) to 1.0 (best value), and facilitates the comparison between methods and summarizing concepts. It obviously implies loss of information which may be relevant in particular situations in which precision and recall need to be discriminated or weighted to model the actual consequences of the misdiagnosis [28]. Other similar performance indicators that have specific application could be found in [70] [71] [72].

In data stream problem, two ways for sampling methods are often applied: holdout and predictive sequential (prequential). In the holdout method, the samples are divided into training and holdout sets. The training set is used for adjusting the parameter of the classifier trained online, and the holdout set for testing the classifier at the regular time distance. Then by one of the performance indices, the accuracy of the classification task is reported. In the prequential method, each sample of the stream is tested, and then the error rate is computed by the accumulated sum of a loss function between the prediction and true values [73] [74].

For evaluating ND methods, various performance indices are applied. The effectiveness of ND techniques can be evaluated either with the number of novel samples that are correctly detected or with the number of non-novel samples that are wrongly classified as novel samples, which known as false alarm rate. Receiver operating characteristic curves are used to represent the trade-off between the detection rate of novel samples and the false alarm rate [68].

In the next chapters, some algorithms for FDD updating and data processing are proposed; those algorithms must be quantitatively compared. Thus, indices are proposed that give weight to the goal of the algorithms (that is having more accuracy for classifying different classes) versus the cost of the meeting the goal (that is usually CPU time). Additionally, in chapter four for justifying performance of the automatic clustering algorithms and OCCs, new indices are proposed and applied.

1.4 Benchmark Case Studies

The case studies that are employed in the next chapters are described in the following subsections:

1.4.1 Continuous Stirred Tank Reactor (CSTR)

The addressed case study is a Continuous Stirred Tank Reactor (CSTR) with a cooling jacket in which Cyclopentenol is produced from Cyclopentadiene by acid-catalysed electrophylic hydration in aqueous solution. This reactor was first described by Chen [75] and is reported in many research works [76] [77] [78]. The so-called Van der Vusse reactions are described by the following reactions, Equation (1. 18):

$$\begin{gathered} A \overset{k_1}{\rightarrow} B \overset{k_2}{\rightarrow} C \\ 2A \overset{k_3}{\rightarrow} D \end{gathered} \tag{1. 18}$$

In Equation (1. 18), the desired product of these reactions is cyclopentenol, B, while cyclopentanediol, C, is produced in an unwanted consecutive reaction, and dicyclopentadiene, D, is formed as a by-product of these reactions.

Each dataset that is produced by simulation of this CSTR tank has four features; first feature is the concentration of the reactant, C_A; second feature is the concentration of the product, C_B; third feature is temperature of the reactor, T_R; and fourth one is temperature of the cooling jacket, T_J. The nonlinear differential equations, Equation (1. 20) to Equation (1. 23), are derived from component balances for substances A and B, and from energy balances of the reactor and cooling jacket. In the equations, V is flow fed to the reactor that contains only reactant with the initial concentration of the C_{A0} and initial temperature of T_0. The T is temperature inside the reactor and the reaction velocities k_i are assumed to depend on the temperature via the Arrhenius law, on Equation (1. 19):

$$k_i = k_{i0} exp\left(\frac{E_i}{\frac{v}{°\mathrm{C}} + 273.15}\right) \quad i = 1,2,3,\ldots \tag{1. 19}$$

$$\frac{\partial C_A}{\partial t} = \frac{1}{V_R}\frac{\partial V}{\partial T}(C_{A0} - C_A) - k_1(T)C_A - k_3(T)C_A^2 \tag{1. 20}$$

$$\frac{\partial C_B}{\partial t} = \frac{-1}{V_R}\frac{\partial V}{\partial T}C_B + k_1(T)C_A - k_2(T)C_B \tag{1. 21}$$

$$\frac{\partial T}{\partial t} = \frac{1}{V_R}\frac{\partial V}{\partial T}(T_0 - T) - \frac{1}{\rho C_P}\left(k_1(T)C_A \Delta H_{R_{AB}} + k_2(T)C_B \Delta H_{R_{BC}} + k_3(T)C_A^2 \Delta H_{R_{AD}}\right) + \frac{k_w A_R}{\rho C_P V_R}(T_{K-T}) \tag{1. 22}$$

$$\frac{\partial T_K}{\partial t} = \frac{1}{m_k C_{PK}}\left(\frac{\partial Q_K}{\partial t} + k_w A_R (T - T_K)\right), \quad C_A \geq 0, C_B \geq 0 \tag{1. 23}$$

In Table 1. 1, applied terms in the Equation (1. 19) to Equation (1. 23) are described.

Table 1. 1. Names and values of the reaction parameters in CSTR.

Name of the parameter	Symbol	Value of the parameter
collision factor for reaction k_1	k_{10}	(1.287± .004).1012h^{-1}
collision factor for reaction k_2	k_{20}	(1.287± .04).1012h^{-1}
collision factor for reaction k_3	k_{30}	(9.043±).109 $1/molA.h$
activation energy for reaction k_1	E_1	-9758.3K
activation energy for reaction k_2	E_2	-9758.3K
activation energy for reaction k_3	E_3	$(4.2 \pm 2.36)(kJ/molA)$
enthalpies of reaction k_1	$\Delta H_{R_{AB}}$	$-(11.0 \pm 1.92)(kJ/molB)$
enthalpies of reaction k_2	$\Delta H_{R_{BC}}$	$-(41.85 \pm 1.41)(kJ/molA)$
enthalpies of reaction k_3	$\Delta H_{R_{AD}}$	$(93.42 \pm 4.10-4)(kJ/L)$
heat capacity	C_P	$(3.01 \pm .04)\ kJ/kg.K$
heat transfer coefficient for cooling jacket	kw	$(4032 \pm 120)\ (kJ/h.m^2\ K)$
surface of cooling jacket	A_R	$0.215\ m^2$
reactor volume	V_R	$0.01\ m^2$
coolant mass	m_K	$5.0\ kg$
heat capacity of coolant	C_{PK}	$(2.0 \pm .05)\ (kJ/kg.K)$

1.4.2 Three Tanks

The three tanks system, Figure 1. 4, could be described through a mathematical model and includes typical characteristics of tanks, pipelines, pumps networks, cooling water circuits of distillation columns and reactors [27]. The three tanks system has been widely used as a benchmark case study in monitoring, control and FDD studies [25] [79].

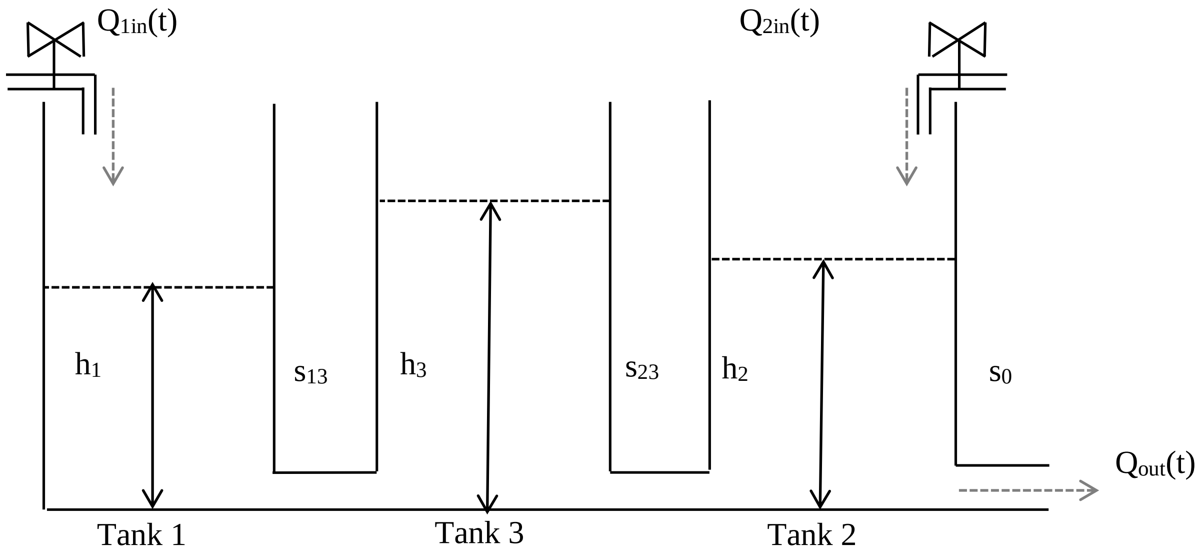

Figure 1. 4. Three tanks benchmark system.

The system consists of three identical cylindrical tanks of cross section area $A = 0.0154\, m^2$, which are serially interconnected by three cylindrical pipes of cross section area $s_{13} = s_{23} = s_0 = 0.005\, m^2$, and flow coefficients $a_{13} = 0.6836$, $a_{23} = 0.4819$, $a_0 = 0.4819$. Two pumps are delivering the liquid to the system with flowrates Q_1, Q_2, where the maximum allowed flowrates limit is $0.003\, m^3/s$. The process is described by the set of ordinary differential equations illustrated in Equation (1. 24).

Addition to Normal (Nr) condition, the process is subjected to three faults: Fault 1 (F1) is the leaking in tank 1 ($Q_{f1} = -0.0007\, m^3/s$), Fault 2 (F2) is the plugging in tank 2 ($Q_{f2} = +0.0007\, m^3/s$), and Fault 3 (F3) is the leaking in tank 3 ($Q_{f3} = -0.0007\, m^3/s$). These values have been selected to be between 10% and 25% of the inlet flow, based on the literature of this case study. Gaussian error, representing the noise, introduced by the different sensors, is added to the model output $\mathcal{N}(\mu = 0, \sigma = 0.010)$.

$$\left.\begin{aligned}\frac{dh_1}{dt} &= -a_1\, s_{13}\, sgn(h_1 - h_3)\sqrt{2g|h_1 - h_3|} + Q_1 + Q_{f1}\\ \frac{dh_2}{dt} &= a_3\, s_{23}\, sgn(h_3 - h_2)\sqrt{2g|h_3 - h_2|} - a_2 s_0\sqrt{2gh_2} + Q_2 + Q_{f2}\\ \frac{dh_3}{dt} &= a_1\, s_{13}\, sgn(h_1 - h_3)\sqrt{2g|h_1 - h_3|} - a_3 s_{23} sgn(h_3 - h_2)\sqrt{2gh_3 - h_2} + Q_{f3}\end{aligned}\right\} \quad 1.\ 24$$

1.4.3 Tennessee Eastman (TE)

The Tennessee Eastman (TE) process has been widely used as a benchmark to compare various monitoring solutions [80] [81] [82]. The process is open loop unstable due to the process exothermic reactions. Besides the reactor, the process has four main unit operations, as shown in Figure 1. 5, including condenser, compressor, separator and stripper. The process produces two liquid products (G and H) and one by-product (F) from four gaseous reactants (A, C, D, E) and an inert (B), Equation (1. 25) to Equation (1. 28):

$$A(g) + C(g) + D(g) \rightarrow G(liq) \qquad \text{Product 1} \qquad 1.\ 25$$

$$A(g) + C(g) + E(g) \rightarrow H(liq) \qquad \text{Product 2} \qquad 1.\ 26$$

$$A(g) + E(g) \rightarrow F(liq) \qquad \text{By-Product} \qquad 1.\ 27$$

$$3D(g) \rightarrow 2F(liq) \qquad \text{By-Product} \qquad 1.\ 28$$

The original open loop FORTRAN code was provided by Downs and Voge [83]. Different monitoring techniques have been tested and reported for the TE [62] [84]. These techniques have shown different capabilities in detecting the faults assumed for the process. The process has 52 process features and 20 faults or disturbances to be diagnosed. In regard to the process variables, 41 are measured (XMEAS in the original paper) and 11 are manipulated by valves (XMV in the original paper).

The 20 faults of the process are listed in Table 1. 2. Some faults in the TE were not fully described by the authors in the original paper, as it is the case of the faults 16 to 20, reported as unknown. Faults 3 and 9 only differ on the type of disturbance whereas the first one is due to a step fault and the second one due to a random variation. Faults 14 and 15 are the only faults generated due to a stuck valve. In the literature, faults 3, 9 and 15 are usually reported as undetectable faults [81] [84] [6]. In Table 1. 2, IDV based on original paper [83], stands for “Vector of disturbance flags” that could be found in Figure 1. 5.

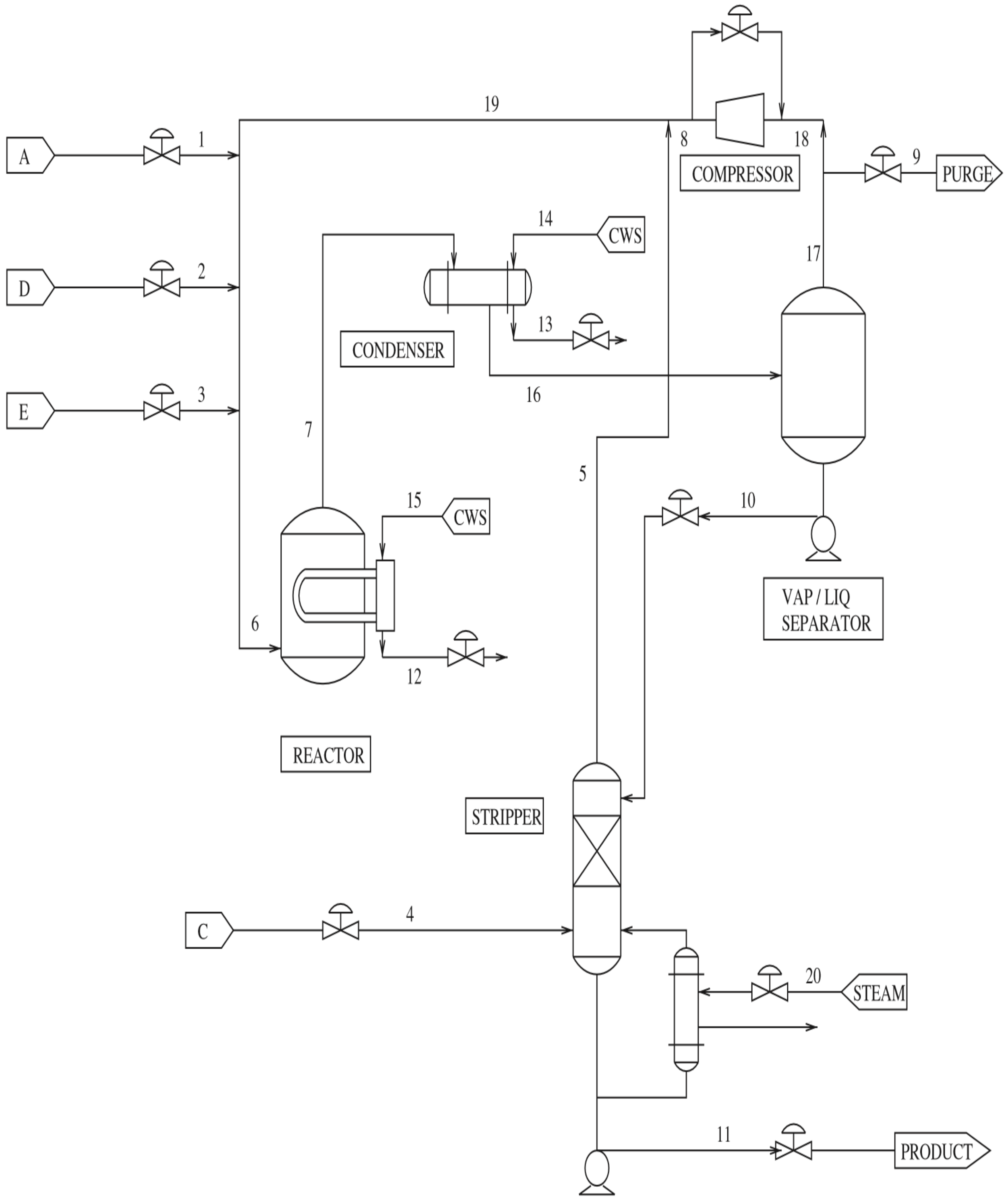

Figure 1. 5.Tennessee Eastman Flowsheet.

Table 1. 2.Faults of the TE Process.

Fault	Process variable	Type
IDV(1)	A/C feed ratio, B composition constant (stream 4)	Step
IDV(2)	B composition. A/C ratio constant stream 41	Step
IDV(3)	D feed temperature (stream 2)	Step
IDV(4)	Reactor Cooling Water Inlet Temperature	Step
IDV(5)	Condenser Cooling Water Inlet Temperature	Step
IDV(6)	A Feed Loss (Stream 1)	Step
IDV(7)	C Header Pres. Loss Reduced Availability (Stream 4)	Step
IDV(8)	A, B, C, Feed Composition (Stream 4)	Random Variation
IDV(9)	D Feed Temperature (Stream 2)	Random Variation
IDV(10)	C Feed Temperature (Stream 4)	Random Variation
IDV(11)	Reactor Cooling Water Inlet Temperature	Random Variation
IDV(12)	Condenser Cooling Water Inlet Temperature	Random Variation
IDV(13)	Reaction Kinetics	Slow Drift
IDV(14)	Reactor Cooling Water Valve	Sticking
IDV(15)	Condenser Cooling Water Valve	Sticking
IDV(16)	Unknown	Unknown
IDV(17)	Unknown	Unknown
IDV(18)	Unknown	Unknown
IDV(19)	Unknown	Unknown
IDV(20)	Unknown	Unknown

Chapter 2: Data Processing

In literature an expression data pre-processing is used for addressing series of steps to transform raw dataset into a clean and tidy dataset prior to statistical analysis [23]. In this chapter, data processing refers to the same concept of data pre-processing. Comparing with the two main steps of process monitoring, data measurement and modeling, data processing that aims to connect these two steps has received less attention [85]. Data processing leads to performance improvement and reducing training time of the FDD [7].

Main distinct steps of data processing could be grouped as data cleaning, data integration, data transformation, and data reduction [23]. Data cleaning is a step that deals with the missing data, noise and outliers; data integration step reorganizes various datasets into single dataset; data transformation unifies formats and unites of recorded data; and in data reduction redundant records and variables are removed [23]. The data processing is usually done in an iterative way. As an example, removing outliers and feature selection steps could be repeatedly applied until the dataset becomes appropriate for training and evaluation of the models [12] [86].

In this chapter, three methods and algorithms that are more practical for FDD are discussed. These methods may be implemented separately or integration of them, depending on FDD systems and plants that are monitored, could be applied. In the first section, imputation of missing values is disused; in the second section, the feature selection problem is investigated; and in the last section feature extension is studied.

.

2.1 Imputation of Missing Values

This section investigates the application of techniques for enhancing the quality of the data, through smoothing the noise, outliers and imputation of missing values that usually contaminate datasets. The information quality enhancements is aimed at improving the training datasets of data driven FDD. A simulation case study of CSTR is applied to produce datasets, and three techniques, including OK, ANN, and PR are applied.

2.1.1 Introduction

Data driven classification techniques have gained wide popularity for FDD due to their flexibility and robustness. The efficiency of these techniques mainly depends on the quality of data by which these classifiers are trained. Modern chemical manufacturing plants are often instrumented with a sophisticated network of sensors that provide enormous amounts of samples. The stored samples must be analyzed for monitoring status of the plant. On the whole, the quality of the process data is corrupted because of several potential problems that commonly happen in different parts of the plant [87]. These problems may appear as noise, outliers, and missing values.

In literature some techniques are proposed that could handle missing values without the need to data imputation [88] [89] [90]. Nevertheless, the main part of the data driven and statistical approaches cannot handle samples with missing values. Organizing missing values has become a fundamental requirement for classification approaches because unsuitable treatment of them may lead to misleading the classifiers [19]. In missing value problem, in some samples one or more features have values like $\pm\infty$, 0, ? or any other constants that do not reflect the real state of the physical measured quantity [12]. In order to deal with the missing values three possible ways may exist [23]. The samples that contain missing values could be ignored, that is not the effective way. The missing values could be determined and filled manually that is time consuming and prone to mistakes. As another option, missing values could be replaced by expected values. The expected values could be mean of the available data or could be obtained by prediction methods. Another family of approaches for dealing with the missing values are model based methods in which data distribution is modeled [19].

Regression methods and “hot and cold deck” approaches are two main groups of prediction methods for imputation of missing values. In regression methods missing values are filled in by the predicted values from a regression analysis [90]. One of the main advantages of the regression methods is that they keep the variance and covariance of the features with the missing values [19]. In hot and cold deck approaches, which has no parametric model, missing values are imputed with the values from a similar complete data vector [91]. These approaches have two steps for imputing the missing values. The first step is classification that the dataset is divided into disjoint clusters.

In the second step, for each incomplete sample the complete samples in its cluster are used for filling in missing values [92].

Another prediction approaches for dealing with the missing values are those based on machine learning. K-nearest neighbor, self-organizing map, multi-layer perceptron, recurrent neural network and auto-associative neural network are among most applicable ones [19].

Three standard mechanisms for missing values would be considered. First, Missing Completely At Random (MCAR) in which the probability of an observation being missing depends only on itself. Second, Missing At Random (MAR) that the probability of a value being missing is related only to the type of observed value. Third, Not Missing At Random (NMAR) that the probability of a value being missing is related to its amount. An example for NMAR is a sensor that does not detect temperatures below a certain threshold [87] [92].

Outlier detection and handling noise in datasets are part of the data cleaning which are very critical for data driven process modeling [86]. Even with a small portion of outliers great negative effects will appear on process model [93]. In [23], three ways of dealing with the noise and outliers are proposed that are binning methods, clustering and machine learning approaches. In binning methods, values around sample are applied for smoothing. In clustering methods, outliers are detected by grouping samples, and in machine learning methods data are smoothed by means of machine learning approaches.

In this section, the prediction accuracy and smoothing capability of the OK, ANN and PR techniques as regression methods are compared through their application to CSTR simulation case study. The CSTR simulation model is used to produce different sets of training; each includes different amount/percentage of noise, outliers and missing values.

2.1.2 Methodology

Consider a process history raw dataset, X_{IJ} , where I is the number of samples, and J is number of the features, Equation (2. 1):

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & x_{ij} & \vdots \\ x_{I1} & \cdots & x_{IJ} \end{pmatrix} \quad \text{i=1,2,3,}\cdots I, \qquad \text{j=1,2,3,} \cdots J \tag{2. 1}$$

Assume missing values in the dataset, as well as a certain proportion of Gaussian noise and outliers. With each regression technique and for each feature a model, $\widehat{m}_{Tec,j}$, with the available data (containing missing values, noise and outliers) for approximating the feature behavior is trained. The models are applied to predict all the samples and making recuperated dataset $\hat{X}_{Tec}$, Equation (2. 2):

$$Model_{Tec} = \{\widehat{m}_{Tec,1}, \cdots \widehat{m}_{Tec,j}. \cdots, \widehat{m}_{Tec,J}\}, \quad \mathrm{j} = 1,2, \ldots \mathrm{J} \quad \mathrm{Tec} \in \{OK, ANN, PR\}$$

$$\hat{x}_{Tec,ij} = \widehat{m}_{Tec,j}(x_{ij}) \ \ \forall\, i, j$$

$$\hat{X}_{Tec} = \{\hat{x}_{Tec,ij}\} = \{\widehat{m}_{Tec,j}(x_{ij})\} = \begin{pmatrix} \widehat{m}_{Tec,1}(x_{11}) & \cdots & \widehat{m}_{Tec,J}(x_{1J}) \\ \vdots & \widehat{m}_{Tec,j}(x_{ij}) & \vdots \\ \widehat{m}_{Tec,1}(x_{I1}) & \cdots & \widehat{m}_{Tec,J}(x_{IJ}) \end{pmatrix} \tag{2. 2}$$

The accuracy of each model $\widehat{m}_{Tec,j}$, is assessed by comparing predicted values of the model with provided values by related ideal model, $\overset{\circ}{m}_j$. The $\overset{\circ}{m}_j$ returns samples free from noise, outliers and missing values. Root Mean Square Error (RMSE) is calculated as an index for comparing performance, Equation (2. 3):

$$rmse_{Tec,j} = RMSE\left(\overset{\circ}{m}_j(x_{ij}), \widehat{m}_{Tec,j}(x_{ij})\right), \qquad j = 1,2, \ldots J \quad \mathrm{Te} \in \{OK, ANN, PR\} \tag{2. 3}$$

Information quality enhancement achieved by each technique is evaluated through training of the classifier with the recuperated dataset $\hat{X}_{Tec}$, and then testing with the validation dataset that is free from noise, outliers and missing values.

2.1.3 Case Study

The CSTR simulation model is used to generate a dataset. The datasets are generated under both normal and faulty conditions while the fault is caused by a step change in inlet concentration of the reactant from its initial value 5.11 mol/L to 5.13 mol/L along the process operation times. Nine datasets $X_{2000,4}$ are made that include noise, outliers and missing values. The amount of added noise ($\mathcal{N}(\mu = 0, \sigma = \pm 0.08)$) for all samples in nine datasets is fixed. Moreover, the number of outliers ($\mathcal{N}(\mu = 0, \sigma = \pm 0.17)$) in all datasets are 10% of samples, which are selected randomly. Percentage of missing values, with MCAR standard, in training datasets are changed, including 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% and 90% of the total number of samples. For evaluating the classifier testing dataset, $X_{1000,4}$, is used that is free from noise, outliers and missing values.

2.1.4 Results

Figure 2. 1 compares the performance of exploited techniques whereas number of missing samples are 200 (2000×0.1). Figure 2. 1 shows the high ability of the techniques (especially OK and ANN) for capturing the process real behavior despite the fact that their training is accomplished with noise and outliers.

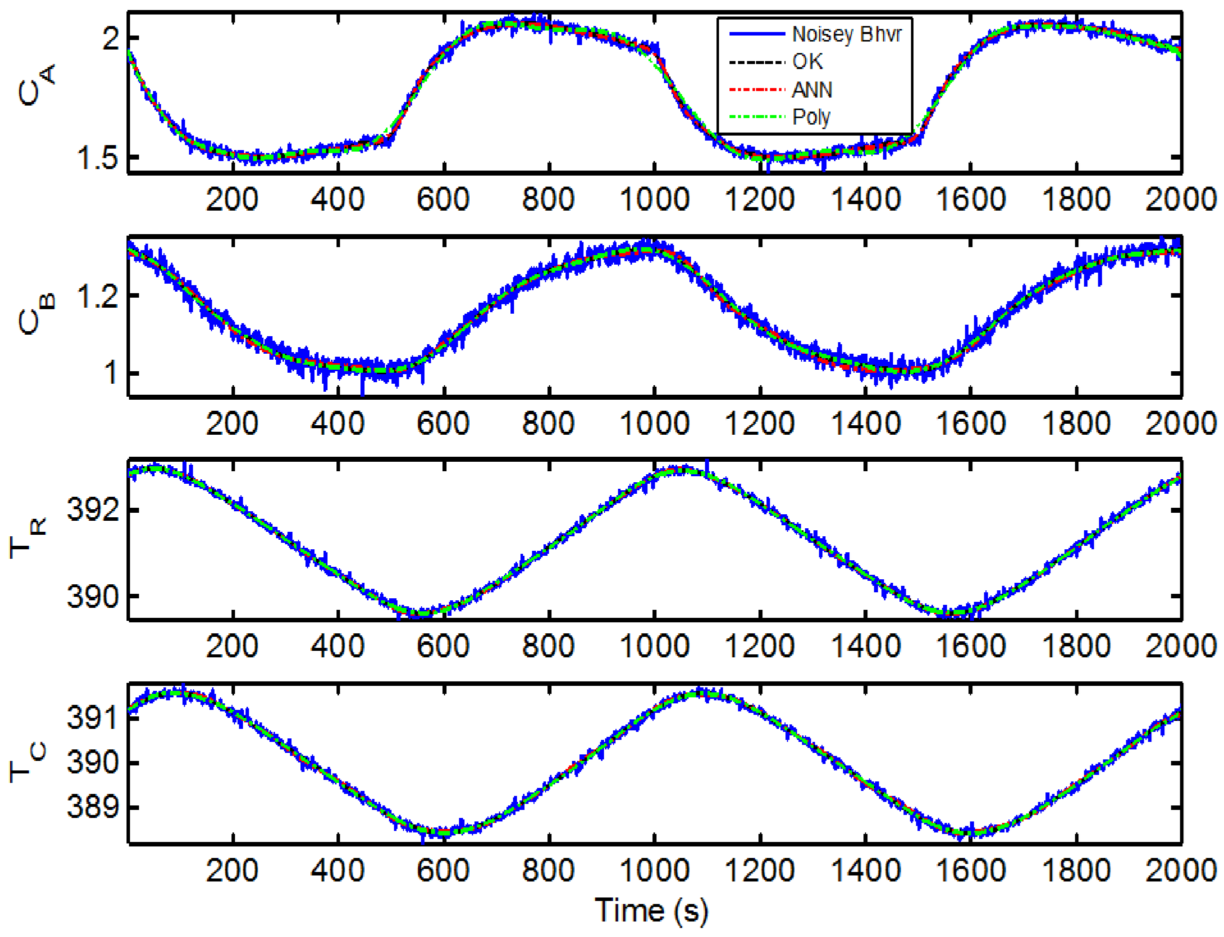

Figure 2. 1. Predictions of the OK, ANN and PR on training dataset with 10% of missing values.

Figure 2. 2 presents the $rmse_{Tec,j}$ of the applied techniques for smoothing and imputing four features of the nine training datasets. Figure 2. 2 illustrates that the models behave with high prediction accuracy even though 90% of samples in the training dataset are missed. Figure 2. 2 presents that the OK has the best prediction.

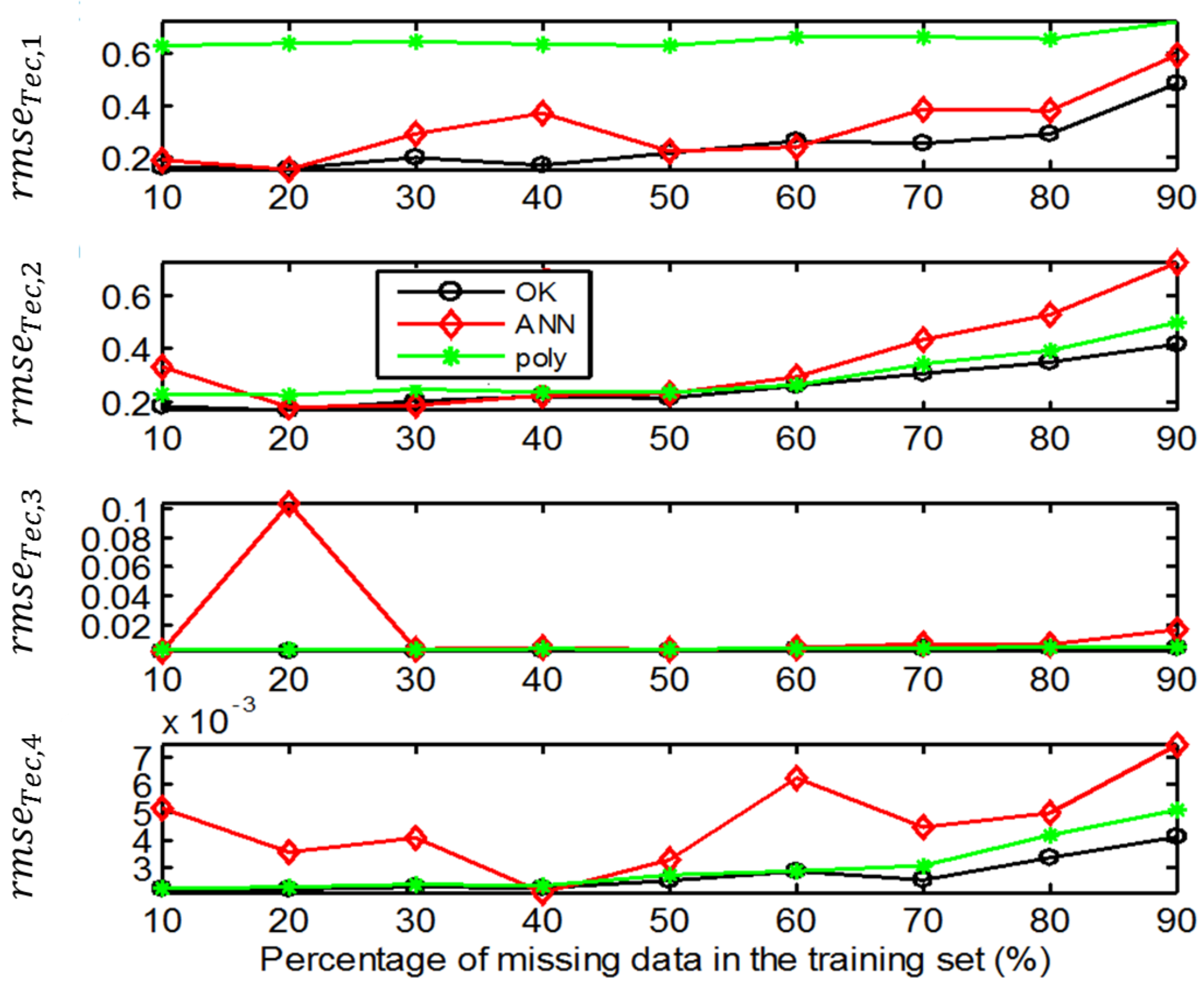

Figure 2. 2. $rmse_{tec,j}$ *for smoothing and imputation of nine datasets.*

The SVM is selected as the classifier, and Figure 2. 3 shows CAS of it whereas it is trained with each of the recuperated training datasets using the three imputation techniques. Additionally, for sake of comparison, a primitive way to impute the missing values that is replacement of them with the mean value of the measured features along the time is considered, too. In Figure 2. 3, “MEAN” refers to replacing missing values with the mean value of the features. With the OK as the imputation technique, and while percentage of missing values in training dataset change from 10% to 90%, mean of the F1 score is 99.28%. This amount for PR, ANN and “MEAN” are 99.23%, 98.57% and 89.01% respectively.

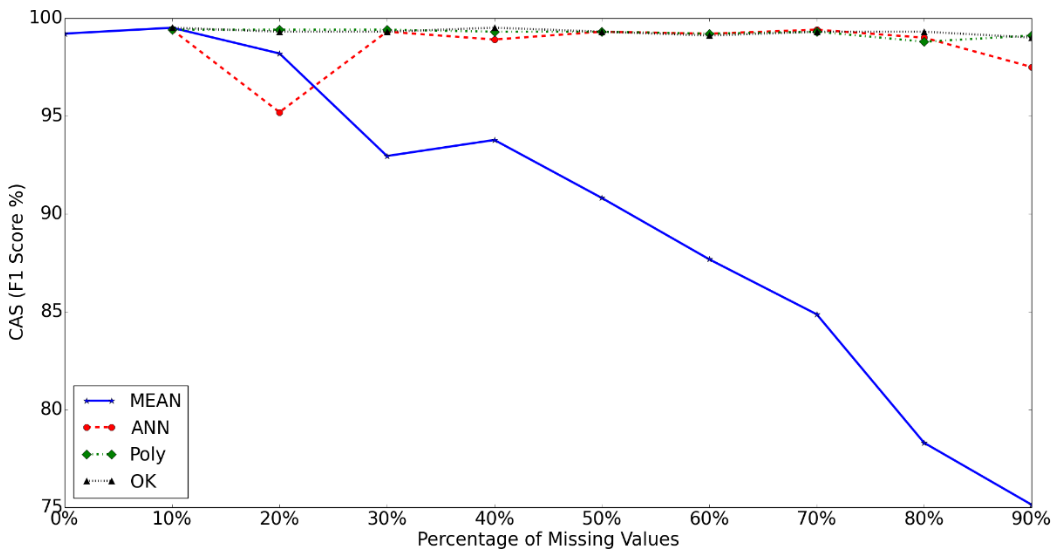

Figure 2. 3. Effects of missing values on classifier CAS.

2.1.5 Conclusions

This section presents techniques for enhancing data quality. Existing data driven classifiers usually require unspoiled data to achieve an efficient training. Nevertheless, real industrial process databases usually include high amounts of noise, outliers and missing values that seriously affect the quality of the data. Applications of exploited techniques to CSTR case study illustrate that the OK has higher prediction accuracy for predicting the missing values, and higher capability for smoothing noise and outlier values. The impact of these characteristics has been demonstrated by considerable enhancement in the classifier CAS while it is trained with the datasets treated by the proposed approaches.

2.2 Feature Selection

Processing huge amounts of data recorded by modern monitoring systems may be confusing, complex and time consuming. Feature selection algorithms are seen as important data processing methods that have proved advantages for improving FDD performance. Feature selection as an optimization problem is advisable in order to determine the optimal subset of features for conducting statistical analysis and building a machine learning model. In this section, two main approaches of feature selection algorithms including filter and wrapper methods are investigated.

2.2.1 Introduction

FDD of industrial processes is a challenging task that demands effective and timely decision making procedures under extreme conditions of highly interrelated data, large number of inputs and complex interaction between the symptoms and faults. For data driven FDD methods, quantity of training data may affect quality of classification, therefore feature processing methods are designed to analyze quantity of training data. The aim of them is to improve classification task, and lessen computational effort, training time as well as testing time.

There are three common ways of processing measured features, including feature extension, feature extraction and feature selection. Feature extension is about providing new features that are usually presented by some statistical and mathematical calculation based on available features. Feature extraction is about reducing number of features by transforming them into new space with the fewer dimensions [2], and feature selection reduces number of features by selecting important features among available measured features.

It is common that some features are non-informative because they are either irrelevant or redundant. Hypothetically, more features should lead to more discriminating capacity, but actually with a limited amount of training data excessive features not only significantly slow down the learning process, but also cause the classifier to over-fit the training data [94]. Features are classified into three categories [95] that are strongly relevant, weakly relevant, and irrelevant features. Strong relevance of a feature proves that the feature is vital for optimal subsets, and ignoring it will prevent determining the optimal combination of features. Weak relevance means that the feature probably could be included in the optimal subset, and irrelevance means that the feature is not required at all.

In feature selection, features that are relevant and not redundant must be selected. In [96], the goal of the feature selection is declared as determination of the best feature subsets for building a machine learning model. In [97], advantageous of feature selection are considered the following: facilitating data visualization and data understanding; reducing the measurement and storage requirements; reducing training time; and improving prediction performance.

With the availability of the labels, feature selection methods could be divided into supervised, semi supervised and unsupervised approaches. The supervised feature selection methods select optimal features by the correlation between features and class labels. In unsupervised feature selection approaches, features are selected by the capability of keeping certain properties of the data such as variance [96] [98].

Among different features selection methods, wrapper models and the filter models are more common [99]. The wrapper models use the predictive accuracy of a predetermined learning algorithm to quantitatively assess the candidate subsets. Wrappers methods are often criticized because they need massive amounts of computation, but efficient search strategies could be devised to assist [97]. For wrapper methods there are three critical topics that must be considered. First, way of searching in the space of all possible feature subsets, second, selecting performance indicator to guide the search, and third selecting type of the classifier [97].

The filter methods separate feature selection from classifier learning, and selects feature subsets independently of the learning algorithm [100] [94]. In filter methods the optimal subset of the features is selected based on relevancy and redundancy criteria. Relevancy criteria determine how well a feature discriminates between the classes and it must be maximized. Redundancy criteria measure how similar the features are and it must be minimized. In other words, redundancy criteria imply how much adding a future to a given set of features contributes to prediction [101].

In this section, wrapper and filter methods are compared through their application on TE benchmark. Selection of important features is done with optimization method based on the three combination criteria of relevancy and redundancy for filter methods and four criteria for wrapper methods. The comparison of methods is based on CAS and CPU time of training. A dimensionless score is suggested and applied for worth giving to classification performance and CPU time of training simultaneously while the classification performance has more weight.

.

2.2.2 Methodology

During diagnosing dataset X, Equation (2. 4):

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & \ddots & \vdots \\ x_{I1} & \ldots & x_{IJ} \end{pmatrix}, \quad \boldsymbol{x}_j = \left(x_{1j}, x_{2j} \ldots \ldots .. x_{IJ}\right) \qquad 2.\ 4$$

For having maximum CAS and minimum CPU time of training, some redundant features, $\boldsymbol{x}_j$, must be removed, thus Main Classifier (MC) and Feature Selection Method (FSM) must be selected, Equation (2. 5):

$$\begin{aligned} &\min(-CAS = f(MC_m, \boldsymbol{D}), CPU = f(MC_m, \boldsymbol{D})) \\ &s.t.\ \ MC_m \in \{MC_1, MC_2\ , \ldots MC_M\} \\ &\boldsymbol{D} = f(FSM_u) \\ &FSM_u \in \{FSM_1, FSM_2, \ldots FSM_U\} \\ &\boldsymbol{D} \in \left\{d_j \middle|\ d_j \in \{0,1\}\ \forall j = 1,2, \ldots J\right\} \end{aligned} \qquad 2.\ 5$$

MC is applied for training and testing with the final set of features. d_j is the binary decision for selecting feature $\boldsymbol{x}_j$ that could be 0 or 1, and $\boldsymbol{D}$ is the vector of all d_j. Figure 2. 4 presents solution space for feature selection.

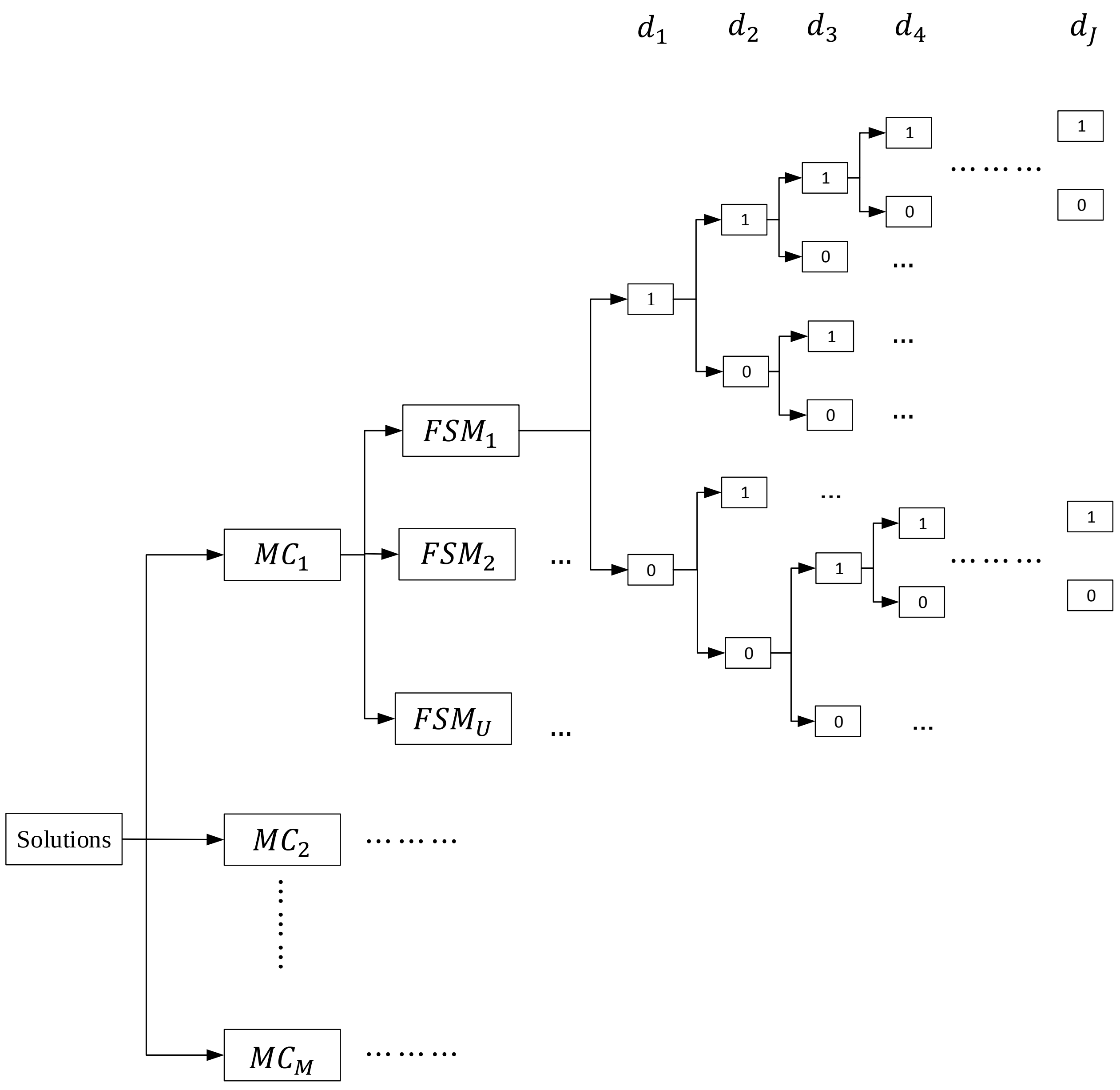

Figure 2. 4. Solution space.

Accordingly, in dataset X, some features may be selected for making dataset X', Equation (2. 6):

$$X' = \begin{pmatrix} x'_{11} & \cdots & x'_{1J'} \\ \vdots & \ddots & \vdots \\ x'_{I1} & \dots & x'_{IJ'} \end{pmatrix}, \quad J' = |\boldsymbol{D}|, J' < J \qquad 2.\ 6$$

In order to find the optimum features $\boldsymbol{D}$ in solution space, the optimization algorithm is applied based on Figure 2. 5. The objective function for wrapper methods is based on the CAS of the Wrapper Method Classifiers (WMC). In the filter approaches $\boldsymbol{D}$ is provided based on relation between features. So, the objective function of the filter methods calculated by dividing relevancy criteria by redundancy criteria, however difference between these two criteria could be another option [102].

After optimization task and determining $\boldsymbol{D}$, dataset is divided into the training and testing datasets. The selected MC will be trained and then CAS and CPU time are calculated. In order to assess combination of MC with the provided $\boldsymbol{D}$ by various FSM, the MGM score is defined Equation (2. 7):

$$\begin{aligned}
&MGM_{MC_m,FSM_u} = \omega \times \frac{\mathrm{CAS}_{MC_m,FSM_u}}{\mathrm{CAS}_{MC_m,ref}} - \frac{\mathrm{CPU}_{MC_m,FSM_u}}{\mathrm{CPU}_{MC_m,ref}}\\
&MC_m \in \{MC_1, MC_2, \dots MC_M\}\\
&FSM_u \in \{FSM_1, FSM_2, \dots FSM_U\}\\
&ref = reference\\
&\omega = weight\ coefficient
\end{aligned} \tag{2. 7}$$

MC and WMC are selected among four classifiers including SVM, DT, KNN and GNB. Reference in Equation (2. 7) is applying all the features.

Here, are a brief explanation for the applied relevancy and redundancy criteria in filter methods:

Max-Relevancy MR and Min-Redundancy (mR) [103]; MR is based on selecting the features with the highest relevance to the target classes, and it is based on mutual information values between individual features and classes. Mutual information is an indicator of relevance between two random features [104]. For mR mutual information between selected features must be minimized. Value Difference Metric VDM and Redundancy VDM (RVDM) [101]; they are based on this fact that conditional distributions of features must be distinct from each other. Fit Criterion (FC) and Redundancy FC (RFC) [101]; they use the average accuracy of the separation by the normalized distance from centers of distribution.

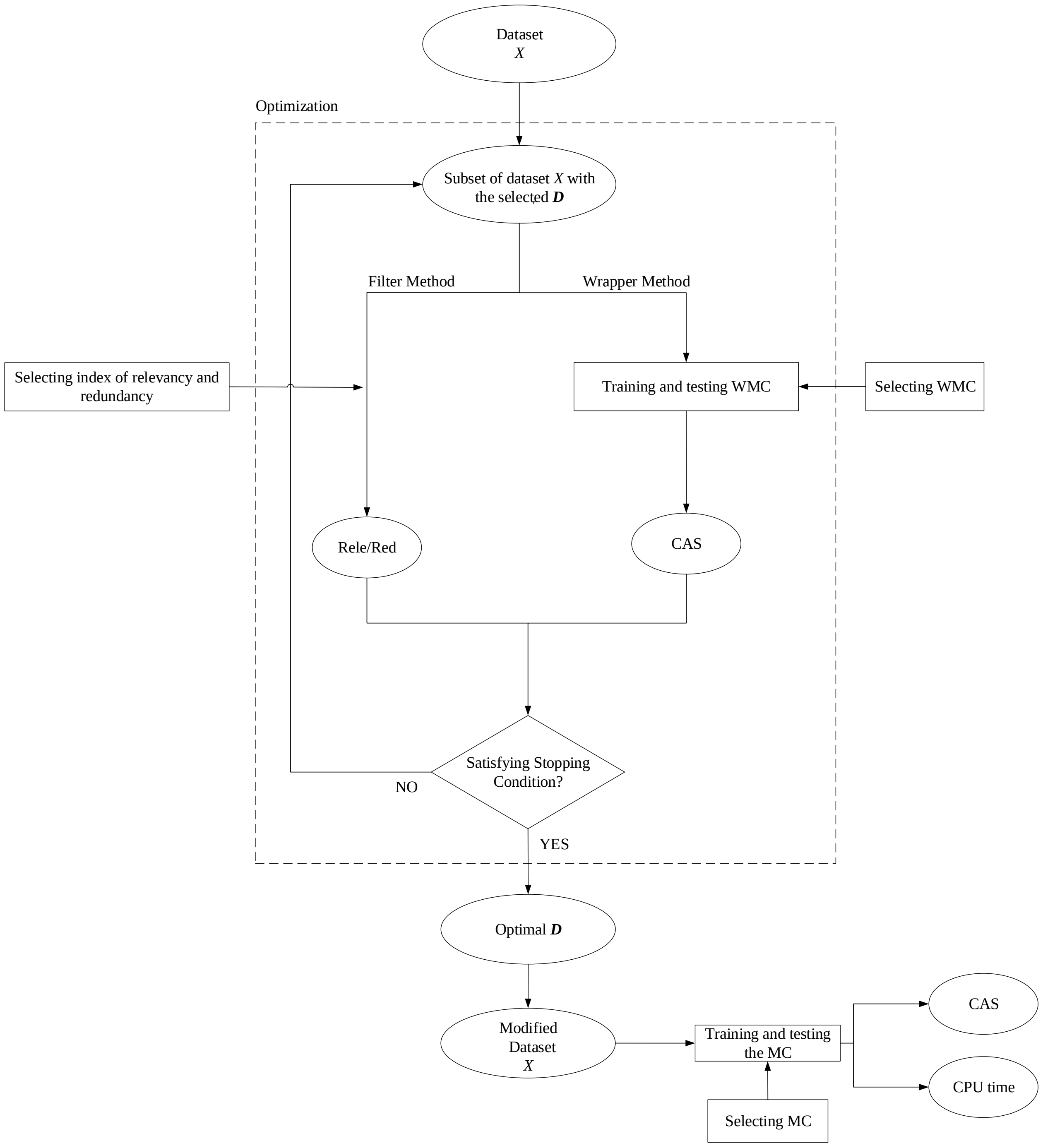

Figure 2. 5. Feature selection algorithm.

2.2.3 Case Study

In this section, a dataset is made with TE simulation that includes all 52 variables, and all faults with 5700 samples.

2.2.4 Results

Optimization algorithm, stopping condition and CAS are selected to be GA algorithm, number of the repetition, and F1 score, respectively. In addition, for MGM score ω is considered two. Table 2. 1 and Table 2. 2 present the results of applying filter and wrapper methods. Based on Table 2. 1 with filter methods and for all the MCs, F1 scores decrease, and for all MCs except SVM the CPU time of training decreases.

Based on Table 2. 2, for wrapper methods improvements in F1 score depend on both MCs and WMCs. Furthermore, with all the MCs and WMCs, CPU time of training decreases.

In Table 2. 3 and Figure 2. 6 MGM scores for FSMs and MCs are reported.

Table 2. 1. Results of applying filter methods.

Criteria of filter method	MR/mR	$VDM/RVDM$	FC/RFC	All features
Number of selected features $\|\boldsymbol{D}\|$	26	22	22	52
MC	CAS (F1 score %)			
DT	74	71	74	79
KNN	30	29	30	38
SVM	40	43	44	56
GNB	50	58	48	69
MC	CPU time (s)			
DT	0.148	0.110	0.130	0.226
KNN	0.007	0.006	0.007	0.013
SVM	9.140	7.510	8.430	3.990
GNB	0.008	0.008	0.008	0.016

Table 2. 2. Results of applying wrapper methods.

WMCs	DT	KNN	SVM	GNB	All features
Number of selected features $\lvert \boldsymbol{D} \rvert$	27	29	35	27	52
MC	CAS (F1 score %)				
DT	88	79	77	87	79
KNN	52	64	57	68	38
SVM	64	69	72	75	56
GNB	63	68	69	72	69
MC	CPU time (s)				
DT	0.130	0.120	0.140	0.110	0.226
KNN	0.009	0.007	0.009	0.008	0.013
SVM	0.700	0.690	0.990	0.800	3.990
GNB	0.010	0.010	0.010	0.009	0.016

Table 2. 3. MGM scores for applied MCs and FSMs.

	MGM							
	Filter Methods			Wrapper Methods				
MC	MR/mR	$VDM/RVDM$	FC/RFC	WMC				All features
				DT	KNN	SVM	GNB	
DT	1.219	1.311	1.298	1.653	1.469	1.330	1.716	1.000
KNN	1.040	1.065	1.040	2.045	2.830	2.308	2.964	1.000
SVM	-0.862	-0.346	-0.541	2.110	2.291	2.323	2.478	1.000
GNB	0.949	1.181	0.891	1.201	1.346	1.375	1.524	1.000

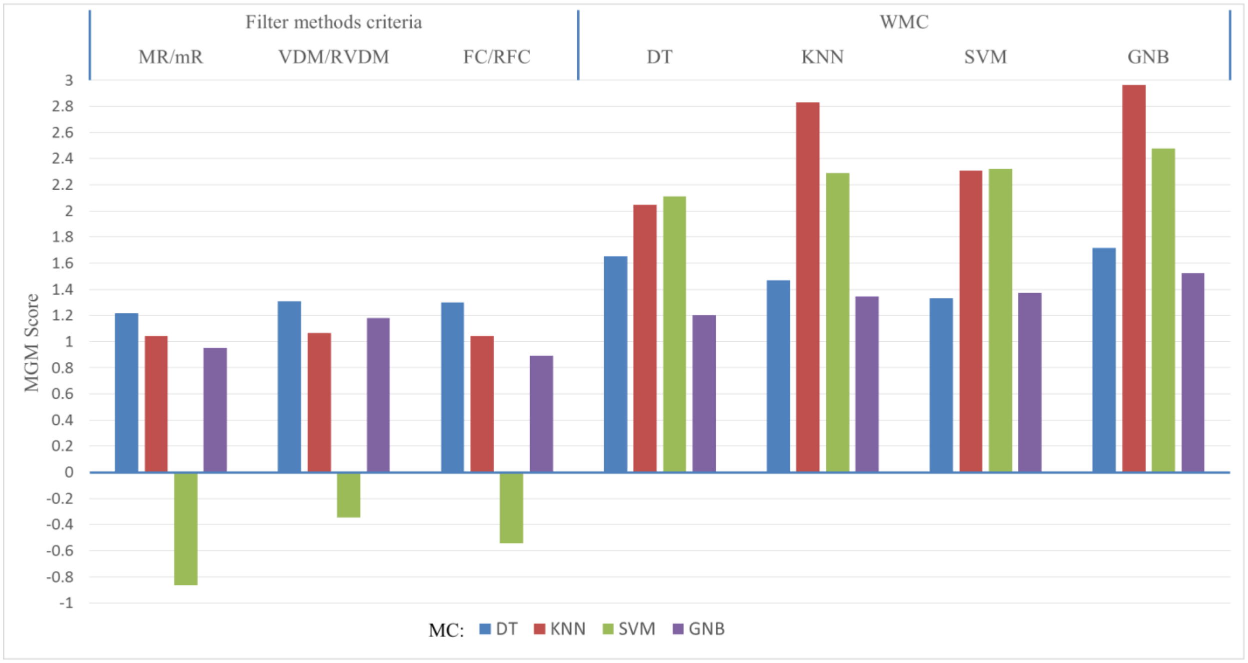

Figure 2. 6. MGM scores for the applied FCMs.

2.2.5 Conclusions

In order to improve performance of the FDD, several methods and criteria as feature selection approaches have been implemented. The best CAS in terms of F1 score (88%) is achieved with the wrapper method while MC and WMC are DT, and 27 of features are selected. However, applying DT as MC and WMC is not the most efficient solution in regard to MGM score. For all the wrapper methods while GNB is applied as WMC the greatest values for MGM score are achieved. Accordingly, the most efficient solution, based on MGM score, is provided by wrapper method in which KNN is applied as MC, and GNB is employed as WMC.

For most of the offline FDD applications difference of CPU time between various methods could be affordable. Reducing CPU time of training, in general, is an inferiority goal comparing with the accuracy of the classification task, but in online classification, advantages of features selection could be more vital for FDD.

2.3 Feature Extension

In this section, with aim of improving performance of the data driven FDD system, feature extension technique with an observer is investigated. The observer produces new error features that would be applied in two different styles; error features together with measured features could be exploited as extended features, or they could be replaced by the measured features. Both styles with various fault patterns and dynamic profiles of the process are studied.

2.3.1 Introduction

Other methods of feature processing are feature extension approaches that are applied in order to assist FDD system to have better performance. They are exploited for information improvement by adding (or replacing) new features to the measured features. They are required for adding more useful information of the process, and finding statistic attributes that describe behavior of the process [6]. However, they have few applications in FDD because it is more frequent to reduce features rather than extend them [105]. The new features are usually made by statistical or mathematical analyses of the measured features that are not explicitly included in the process measurements. In [6], standard deviations of the process variables are considered as extended features. Adding new features imply increasing CPU time of training, nevertheless this task has proved advantages for improving FDD performance [6].

The common way of producing new features is to exploit observers [28]. Observers have been widely used to assist FDD system [106] [107] [2]. Observers by identifying the real underlying behavior of the measured features provide the error features. The error features indicate the extent of the process malfunctioning; they should be close to zero while there is no fault in the process, and they have considerable values when the process is affected by the faults. In order to detect and diagnose the faults the error features could be compared with the threshold values, or they could be processed by statistical approaches [108].

Several studies for observer based FDD of nonlinear processes are addressed that are usually done by model based approaches [109] [110]. Although, in [111] a hybrid data driven framework is proposed in which MDK is applied as an observer in order to enhance performance of the unsupervised FDD for nonlinear dynamic conditions of the process. In the proposed framework, automatic and non-automatic clustering techniques are applied in order to estimate type of faults that may affect the process. Performances of the exploited FDD approaches are compared whereas they are trained and tested with measured features or error features. The results prove that employing only error features significantly assist the clustering techniques to prevent them from confusing the classes in dynamic conditions of the process.

In this section, data processing is done by feature extension approach in which an observer provide error features. FDD task is executed by training and testing classifiers with three scenarios: Using only measured features, together using measured features and error features, and using only error features. These scenarios are tested with various fault patterns and dynamic profiles of the three tanks benchmark.

2.3.2 Methodology

Error features, E, are generated by comparing estimated outputs by an observer, MDK, and the actual outputs [28], Figure 2. 7:

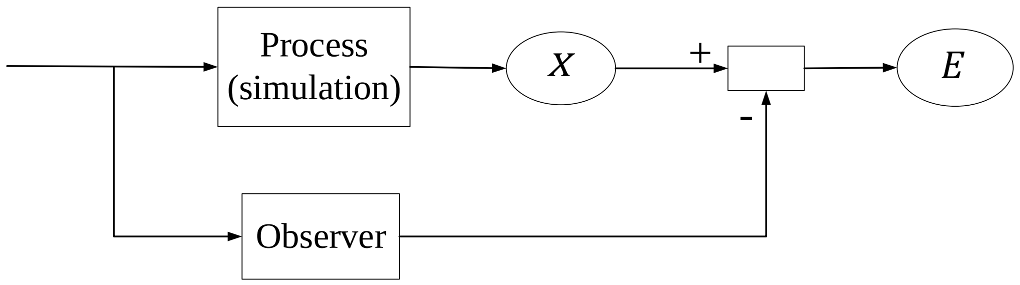

Figure 2. 7. Making error features with an observer.

Three different datasets, including dataset X (with measured features, $\boldsymbol{x}_j$), dataset E (with error features, $\boldsymbol{e}_j$), and dataset XE as union of the X and E (with measured and error features, $\boldsymbol{xe}_j$) are provided, Equation (2. 8) to Equation (2. 10):

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & \ddots & \vdots \\ x_{I1} & \dots & x_{IJ} \end{pmatrix}, \qquad X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_J), \boldsymbol{x}_j = (x_{1j}, x_{2j} \dots \dots \dots x_{Ij}) \tag{2. 8}$$

$$E = \begin{pmatrix} e_{11} & \cdots & e_{1J'} \\ \vdots & \ddots & \vdots \\ e_{I1} & \dots & e_{IJ'} \end{pmatrix}, \qquad E = (\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_{J'}), \boldsymbol{e}_j = (e_{1j}, e_{2j} \dots \dots \dots e_{Ij}) \tag{2. 9}$$

$$\begin{cases} XE = X \cup E, \quad XE = \begin{pmatrix} x_{11} & \cdots & x_{1J} e_{11} & \cdots & e_{1J'} \\ \vdots & \ddots & \vdots \; \vdots & \ddots & \vdots \\ x_{I1} & \dots & x_{IJ}\, e_{I1} & \dots & e_{IJ'} \end{pmatrix} \\ XE = \begin{pmatrix} xe_{11} & \cdots & xe_{1J} & \cdots & xe_{1J''} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ xe_{I1} & \dots & xe_{IJ} & \dots & xe_{IJ''} \end{pmatrix}, \quad J'' = J + J' \\ XE = \left(\boldsymbol{xe}_1, \boldsymbol{xe}_2, \dots, \boldsymbol{xe}_{J''}\right), \; \boldsymbol{xe}_j = \left(xe_{1j}, xe_{2j} \dots \dots .. xe_{Ij}\right) \end{cases} \qquad 2.\,10$$

Then performance of the three classifiers SMV, GNB and DT with these datases are investigated

2.3.3 Case Study

The method is applied to three tanks benchmark case study, with the all described conditions in the section 1.4.2. The training dataset has $I = 1800$ samples, and each test dataset has $I = 400$ samples.

2.3.4 Results

In this section, first training and validating the applied classifiers are discussed, and then by providing various dynamic profiles of the process robustness of the FDD is investigated. Finally, in the last subsection, FDD performance under different faulty scenarios is studied.

2.3.4.1 Training and Validation

Process data including inlets $(Q_1(t), Q_2(t))$ and levels of the tanks $(h_1(t), h_2(t), h_3(t))$ are collected with different process conditions that contain normal and different faults. Figure 2. 8–up shows inlet profiles, and Figure 2. 8-down shows fault scenario. In Figure 2. 8 the same sequences of the faults are repeated with different dynamic inlet profiles; sinusoidal, linear decreasing and linear increasing profiles.

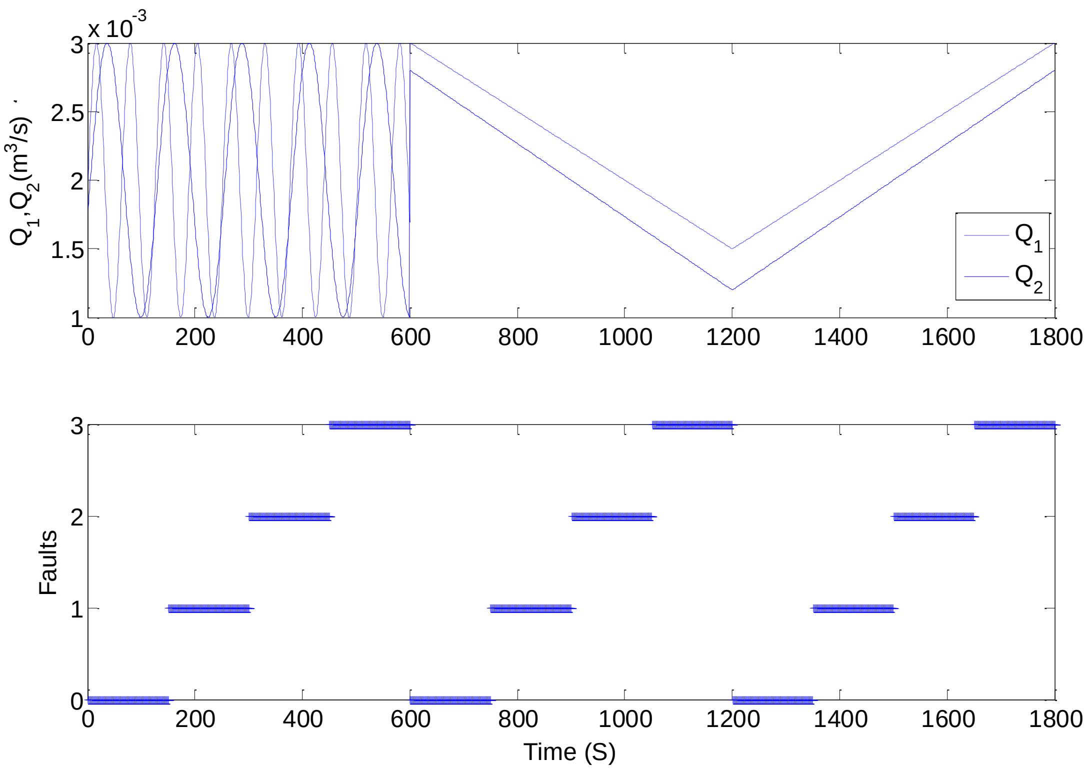

Figure 2. 8. Training and validation of the classifiers: inlet profiles (up), fault scenarios (down).

Table 2. 4 presents CAS of the classifiers in terms of F1 score while they are trained and validated using $\boldsymbol{x}_j$, $\boldsymbol{xe}_j$ and $\boldsymbol{e}_j$. As illustrated in Figure 2. 9, applying only error features for SVM and DT lead to better results. However, for GNB classifier the performances have not changed, significantly.

Table 2. 4. Validation of the classifiers.

CAS (F1 score %)					
Measured Features, $\boldsymbol{x}_j$					
Classifier	Nr	F1	F2	F3	Overall
SVM	11.2	63.2	56.3	39.3	43.1
GNB	87.5	79.3	91.5	79.2	84.4
DT	79.0	74.5	81.8	81.8	79.2
Measured and Error Features, $\boldsymbol{xe}_j$					
SVM	94.5	93.8	97.2	91.6	94.3
GNB	95.2	75.0	96.4	67.4	83.5
DT	97.0	88.5	97.8	86.1	92.3
Error Features, $\boldsymbol{e}_j$					
SVM	96.6	93.9	97.8	91.1	94.9
GNB	95.2	75.3	97.4	65.2	83.3
DT	97.9	89.3	98.0	88.5	93.4

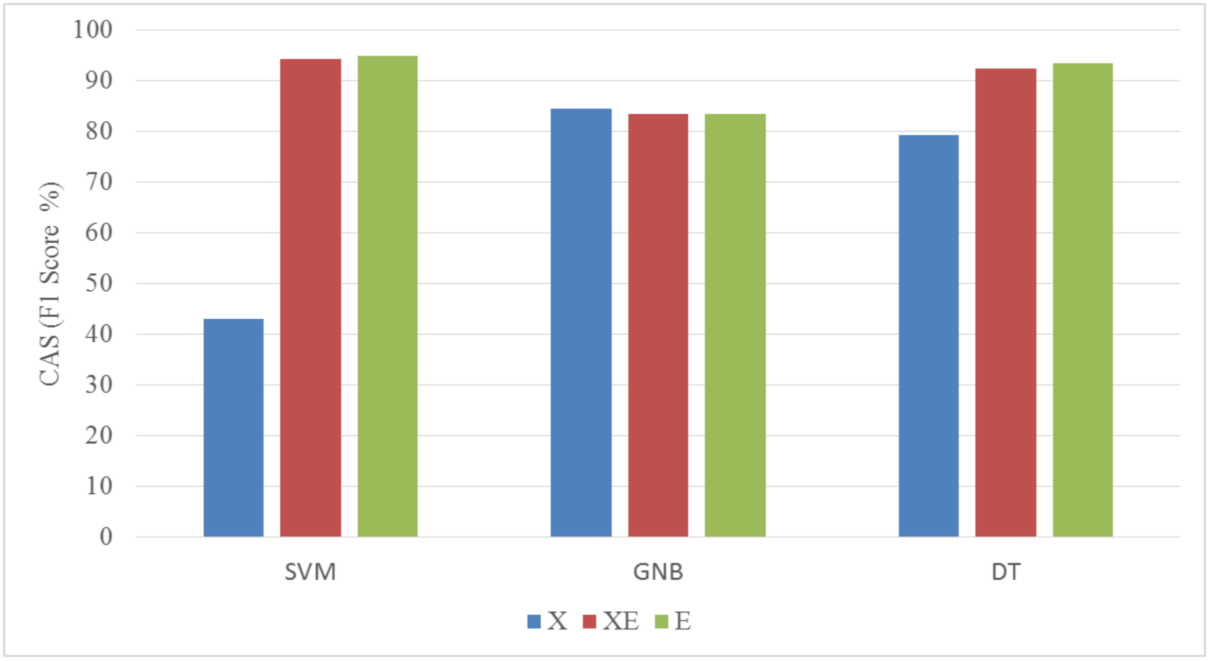

Figure 2. 9. Effects of the feature types on classifier CAS.

2.3.4.2 Robustness against Changes in the Dynamic Profiles of the Process Inlets

Test 1 involves a simple scenario in which both inlets are assumed constant, Figure 2. 10, while different faults occur, Figure 2. 11. More tests (test 2 to test 5) are carried out each one including different inlet profiles while the fault scenario is kept fixed; the inlet profiles present increasing complexity from test 2 to test 5. CAS of the classifiers with the F1 score are reported in Table 2. 5.

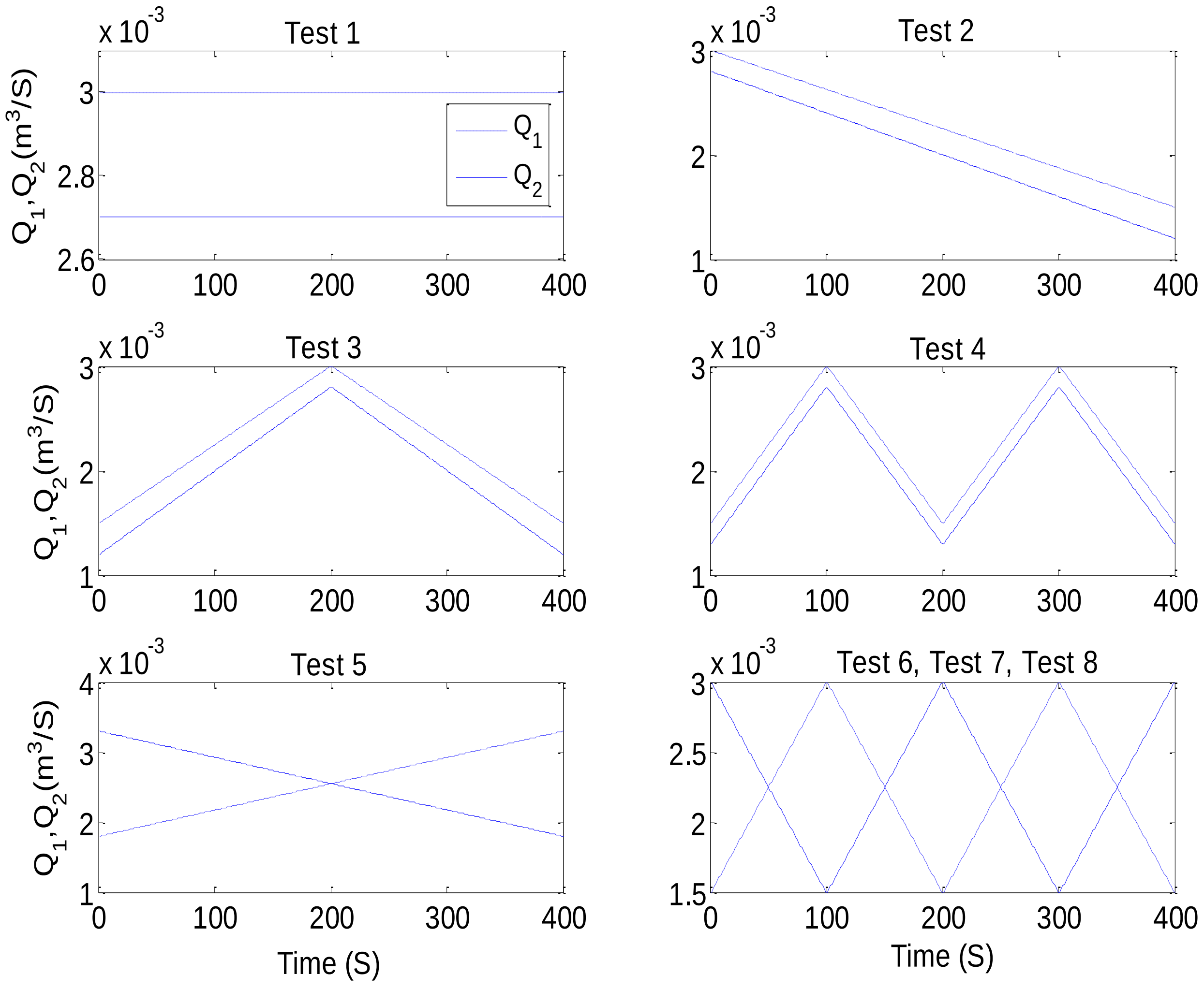

Figure 2. 10. Different inlet scenarios.

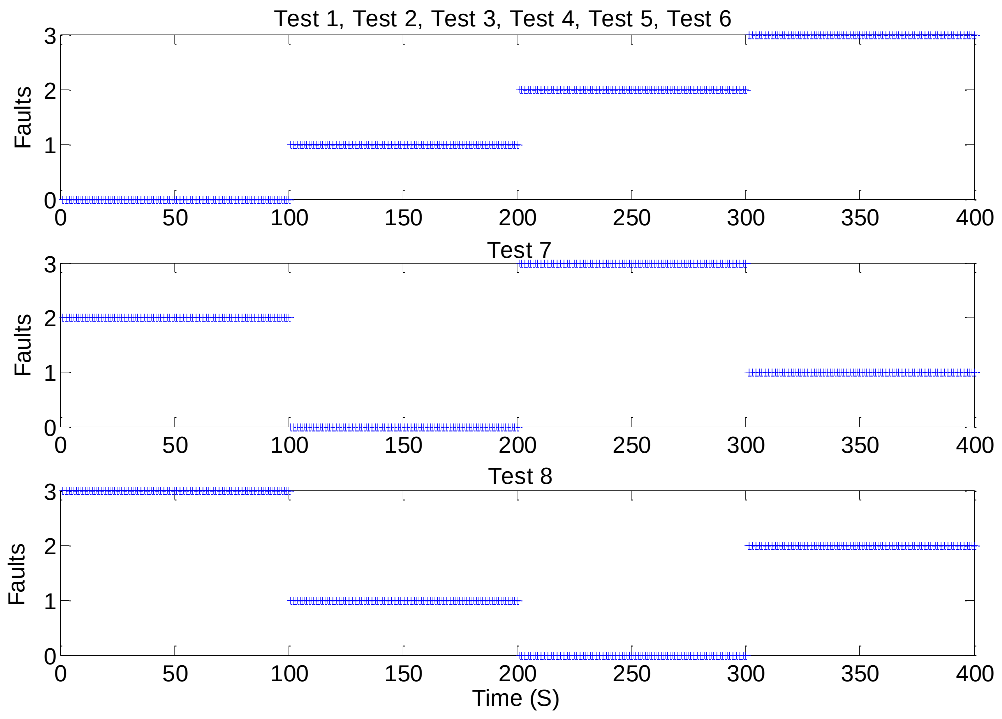

Figure 2. 11. Fault scenarios.

Results in Table 2. 5 reveal that, for all the classifiers, attained CAS by the $\boldsymbol{e}_j$ is usually higher. In Figure 2. 12, this trend for test 1 to test 5 for applied classifiers is presented.

Table 2. 5. CAS of the classifiers with different inlet scenarios.

		CAS (F1 score %)								
		Measured (X)			Measured and Error (XE)			Error (E)		
		SVM	GNB	DT	SVM	GNB	DT	SVM	GNB	DT
Test 1	Nr	92.5	62.0	3.8	91.3	94.6	98.0	97.5	96.0	98.9
	F1	0.0	0.0	0.0	68.7	0.0	0	94.7	15.5	52.1
	F2	66.0	3.2	64.7	94.7	93.8	96.4	95.3	94.4	96.4
	F3	84.7	0.0	59.9	77.7	61.2	63.3	93.1	62.8	93.1
	Overall	60.8	16.3	32.0	83.1	62.4	64.4	95.2	67.2	80.2
Test 2	Nr	92.9	40.0	82.4	92.1	92.0	97.5	95.2	92.7	92.9
	F1	84.6	71.3	86.5	96.9	85.6	100	96.9	86.6	87.8
	F2	84.2	63.5	83.3	96.3	93.7	99.0	96.3	94.8	96.9
	F3	94.7	72.6	95.3	95.8	78.4	97.4	94.4	80.2	78.7
	Overall	89.1	61.8	86.9	95.3	87.4	98.4	95.7	88.6	89.1
Test 3	Nr	62.5	0.0	94.1	93.4	95.0	94.0	96.1	96.1	95.8
	F1	47.7	33.3	64.7	93.8	47.2	59.1	95.4	48.6	70.1
	F2	76.3	58.2	94.6	93.6	95.3	92.6	95.8	95.3	92.2
	F3	79.5	47.3	54.9	94.0	32.0	64.8	94.4	33.1	71.8
	Overall	66.5	34.7	77.1	93.7	67.3	77.6	95.4	68.3	82.5
Test 4	Nr	51.3	0.0	60.5	92.1	93.9	97.5	94.7	95.6	98.4
	F1	36.1	36.5	48.3	86.3	57.3	58.3	88.2	61.4	84.1
	F2	61.7	39.3	66.9	97.4	95.9	98.5	96.4	95.9	94.6
	F3	56.0	19.0	37.8	83.9	33.4	62.2	87.5	34.4	83.5
	Overall	51.3	23.7	53.4	89.9	70.1	79.1	91.7	71.8	90.2
Test 5	Nr	0.0	17.2	92.6	92.4	95.1	94.1	95.6	95.6	90.9
	F1	3.9	56.1	14.8	97.4	90.1	89.1	98.0	85.0	89.5
	F2	32.1	43.2	68.0	95.3	95.1	98	95.9	94.9	91.5
	F3	96.9	0.0	91.1	96.9	91.6	97.4	96.4	80.0	87.0
	Overall	33.2	29.1	66.6	95.5	92.9	94.6	96.5	88.8	89.7

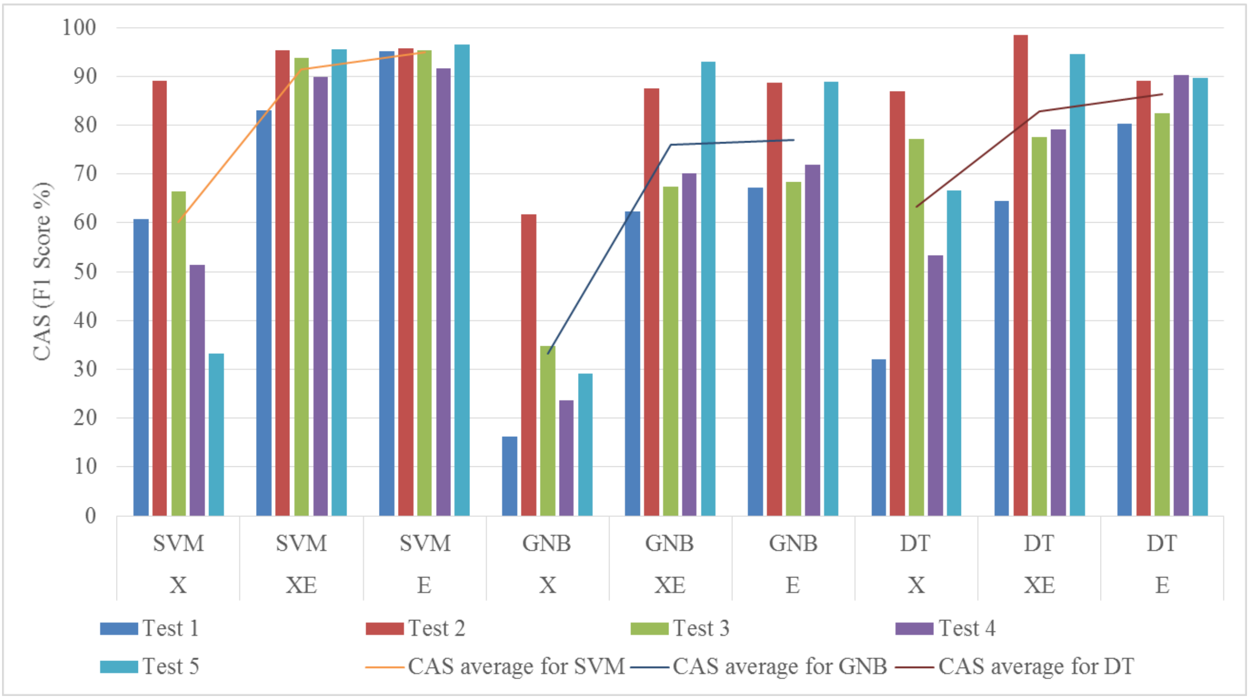

Figure 2. 12. CAS of the classifiers with different inlet scenarios.

2.3.4.3 FDD Performance under Different Faulty Scenarios

In order to analyze effects of more complex sequence of faults on the diagnosis capabilities, different fault scenarios have been incorporated with the same inlet profiles. Figure 2. 10 and Figure 2. 11 present fault and inlet scenarios of test 6, test 7 and test 8. Based on Table 2. 6, for the applied classifiers using $\boldsymbol{e}_j$ leads to higher CAS. Figure 2. 13 shows the effects of the applying error feature on performance improvement of the classifiers.

Table 2. 6. CAS of the classifiers with different fault scenarios.

		CAS (F1 score %)								
		Measured (X)			Measured and Error (XE)			Error (E)		
		SVM	*GNB*	*DT*	*SVM*	*GNB*	*DT*	*SVM*	*GNB*	*DT*
Test 6	Nr	31.2	14.2	29.7	94.3	95.2	92.0	96.1	95.6	96.4
	F1	70.4	64.5	45.4	78.4	65.7	64.2	93.0	65.2	84.7
	F2	63.3	63.0	63.2	97.9	96.9	92.8	97.4	96.9	96.5
	F3	42.3	5.5	17.6	65.8	9.0	24.8	90.1	80.8	79.7
	Overall	51.8	36.8	39.0	84.1	66.7	68.4	94.2	66.6	89.3
Test 7	Nr	30.1	9.5	12.5	96.0	93.6	68.4	95.3	93.6	79.2
	F1	76.0	66.2	50.2	98.0	68.7	73.5	97.5	69.4	96.0
	F2	63.2	69.4	60.7	98.0	94.7	82.1	96.6	94.7	89.2
	F3	50.6	3.6	16.7	95.8	14.8	60.1	96.3	19.8	90.7
	Overall	55.0	37.1	35.0	96.9	68.0	71.0	96.4	69.4	88.8
Test 8	Nr	32.4	05.7	23.3	96.4	95.8	93.6	97.4	95.8	95.2
	F1	73.0	65.3	48.6	91.7	68.5	62.8	98.0	68.5	78.5
	F2	59.2	70.4	61.9	98.4	99.0	98.4	99.5	99.0	98.0
	F3	52.8	10.8	26.4	92.4	17.0	47.3	96.0	20.0	73.2
	Overall	54.4	38.0	40.1	94.7	70.1	75.5	97.7	70.8	86.2

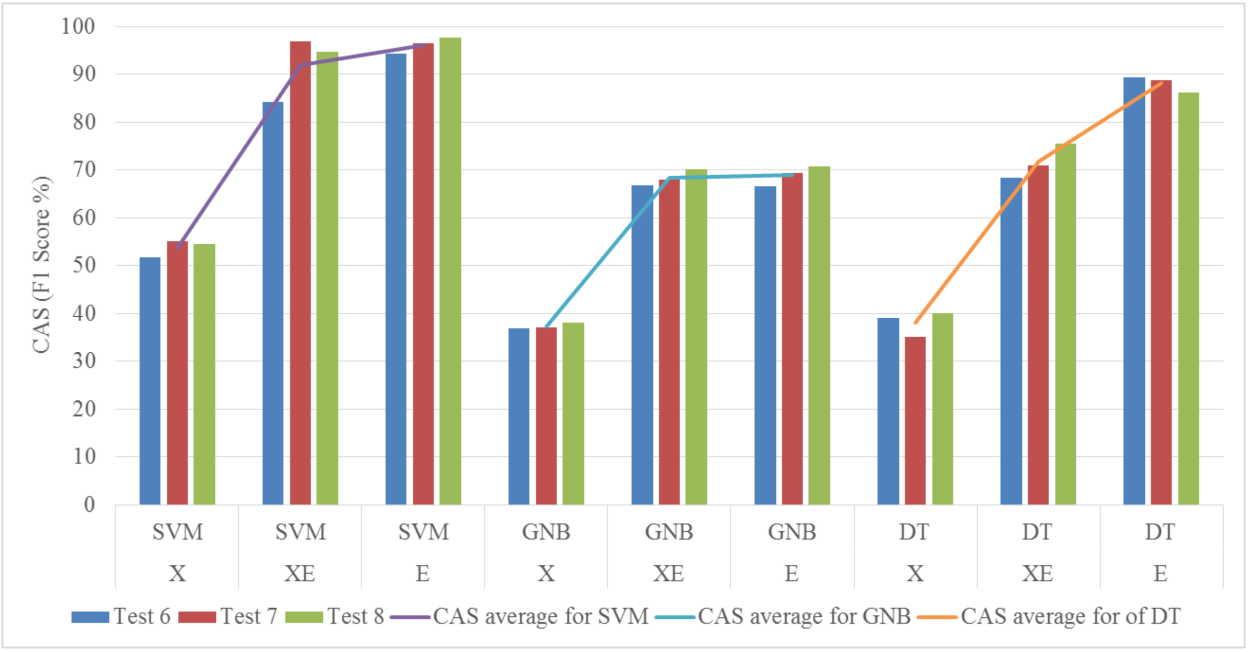

Figure 2. 13. CAS of the classifiers with different fault scenarios.

2.3.5 Conclusions

In this section, feature extension by means of the observer is addressed. The observer produces error features that assist better identification of the process state by the classifiers. The main advantage of the provided error features is that they compensate the effects of the manipulated inputs on the process outputs [28]. The obtained results in supervised FDD prove that replacing measured features by error features improve CAS of the classifiers. These results are validated with various fault patterns and dynamic profiles of the process. It is worth mentioning that as number of samples in all the test scenarios are only 400, required CPU time with $\boldsymbol{x}_j$, $\boldsymbol{xe}_j$ or $\boldsymbol{e}_j$ is trivial and almost same; it changes between 0.001 to 0.0009 second. Accordingly, they are not reported and for selecting the best features only CASs of the classifiers are applied.

In chapter 4, advantageous of the replacing error features by measured features for unsupervised FDD (clustering) will be discussed.

Chapter 3: Supervised Monitoring

3.1 Supervised FD Updating: Handling Concept Drift

The usual FDD approaches are inadequate for dealing with a rapid increase of data volume that is provided in the data stream. Therefore, available approaches must be modified or alternative ones must be exploited. The choice of FDD system highly depends on the demanded task and the need for interpreting the models [9]. The process industry operations are often affected by hidden contexts (such as fouling, aging etc.) result in CD, and thus FD systems need to be regularly updated. Methods based on different criteria have been developed and reported that mostly grouped as classification and sample selection approaches.

IL is a common adaptive approach that its efficiency for updating by the time reduces because it does not forget redundant information (old samples). On the other hand, Fixed sliding Window (FW), which is a usual sample selection algorithm for updating, forgets samples as they exceed a given time limit. Accordingly, some non-redundant information (samples) can be missed. This chapter addresses these limitations by proposing Dynamic sliding Window (DW). The DW as a sample selection algorithm forgets samples after an automatically adjustable time interval. The DW provides data windows, for training the FD, that are most relevant to the current concepts of the process. Furthermore, by combining the classification and sample selection approaches, the Incremental Learning Dynamic Window (ILDW) algorithm is proposed and implemented. The ILDW is intended to learn incrementally from the seen samples and to forget redundant information based on the DW algorithm.

3.1.1 Introduction

The data stream problem was introduced by Henzinger [15] in 1998, and it consists of monitoring arriving data at high rate. It has become an essential part of the knowledge discovery especially in the presence of the CD. In general, the concept could be defined as joint probability between sample $\boldsymbol{x}_i$ and assigned labeled to it y_i by the model, $\mathrm{P}\ (\boldsymbol{x}_i, y_i) = \mathrm{P}\ (\boldsymbol{x}_i)\mathrm{P}\ (y_i|\boldsymbol{x}_i)$. The concept to be learned is not always constant over time, sometimes it changes, this phenomenon is called CD [9]. The CD is defined as a change of the target concept due to changes in some hidden contexts [112]. The target concept is considered as recorded data from the process including all the classes [65] while hidden context is anything that has an influence on the process [113]. The hidden context in the process industries could be fouling, abrasion of mechanic components, catalyst activity changes, etc. [113]. Gama [114] identifies two drift categories in the concepts: when drift is gradual, and when drift is abrupt. These two types of drift correspond to a change in the conditional distribution over time.

For doing FDD tasks, accessing ways to the samples could vary, including all in a dataset, all in memory, partially in memory, and one by one in a stream; for each accessing mood different algorithms exist [9]. The usual accessing way to data is loading the completely training dataset and then process the data. In this way, two main steps of FDD including model learning and deploying the model to predict new samples are done subsequently. This may arise difficulties because of either the memory limitation or lack of accessing to all data in training step. The usual solution for dealing with the memory limitation is to divide the training dataset into several datasets and using parallelization techniques [9] [45]. Commonly, IL methods are required in the following circumstances: Time-dependent situations, obtaining data in batches, and large data comparing with the capacity of available memory [39]. There are two main approaches for implementing the IL, including online and batch methods. In online approach samples are added and analyzed one by one while in batch approach subset of samples are used [39]. In [115], type of supervised IL algorithm is developed that could learn from data, even if they belong to the new classes. Suggested algorithm is able to serve with different classifiers in addition to deal with fine-tuning and overfitting problems, efficiently.

While data is available in the stream, the learning model must be executed incrementally and gradually with low latency. In data stream, the main problem is to deal with large amounts of available data that are produced continuously by modern data acquisition systems. This is a critical issue because required information as well as models and patterns must be extracted from data streams for process monitoring purposes [15] [116] [117]. IL techniques are possible solutions to the scalability problems and large amounts of data. In [73], it is stated that the IL is based on receiving and integrating new samples without the need to learn from scratch. IL strategy could be executed with various classification algorithms such as DT [118] [119], SVM [120], rule based system [121] [112], Naïve Bayes [122] etc.

In [123], differentness between the data stream and other conventional models are recognized with four characteristics: Availability of data in online manner, lack of control over the order of the arriving data, unbounded size of data, and difficulty with retrieving data [123]. In [117], core difference of normal data mining and stream data mining is clarified by addressing online mining of changes in the stream data mining. For data stream mining suggested frameworks or algorithms must have some characteristic: Dealing with continuously unbounded arriving/new data, and tackling changes in data distribution (CD) [124].

By the time, the existing classes may drift or novel classes could appear. Variety of ND methods have been proposed to cope with the occurrence of new process faults/conditions [16] [17]. On the other hand, for managing CD another family of approaches, including IL, decremental learning, adaptive algorithm etc. have been suggested [14] [15]. Only in few studies appearance of new classes and drifting of the existing classes are addressed at the same time [125] [126].

For monitoring stream of data that contains CD, the challenging task is to recognize parts of the training data that are different from the current concepts of the process. Those data must be replaced with recently recorded data that address available concepts of the process. One common possible solution is discarding or forgetting those data after they become so-called old. The criterion for considering data as an old data is to apply predefined Time Interval (TI) [117]. For selecting appropriate TI, it must be considered that long TI will lead to keeping old data whereas short TI could head to overfitting [117].

Developed methods for dealing with CD in the data stream are categorized by three [15]: Single and integrated classification algorithms, implicit and explicit detection methods, and sample based approaches; each mentioned approaches could be grouped into more groups. The difference between implicit and explicit methods is based on using some CD detection approaches in explicit methods while there is no CD detection approach for the implicit methods [15] [127]. CD detection approaches could work with monitoring changes in the probability distribution, changes in relevance characteristics, features of the classification models, or accuracy of the classifiers [15].

For the classification algorithm family approaches, the classifier must adopt the latest available concepts of the process. In [127], for dealing with CD in the data stream ensemble of classification methods are applied. In the model, classifiers are weighted based on prediction accuracy of the currently available data. For updating OCS coping with the CD, a method is proposed that forgets some parts of the data whereas the latest data have highest weights [128]. In [129], a method for improving training phase of the classification algorithm, which has the ability to update with the changes, is developed. In the model, the classifier is updated with the specific time window, and it could handle online classification of the stream data.

The proposed algorithms and methods in the literature for CD detection could be categorized by supervised and unsupervised groups. In the supervised CD detection methods, some performance indices are monitored and reducing them along the time imply the existence of the CD in the process. In the unsupervised CD detection methods, some statistical properties of the features are monitored [13]. In [13], an algorithm is proposed that tracks the number of samples in the uncertainty region of a classifier as a metric to detect CD. In [130], for detecting CD with few numbers of samples statistical tests have been applied; the method is validated by various CD scenarios and results prove that it could handle various types of CD rapidly and accurately. In [131], for CD detection an algorithm is developed that is effective for problems with well separated and balanced classes. It is based on the drift degree that is calculated by comparing samples of two consecutive datasets.

In another aspect, data driven approaches for dealing with CD could be divided into two groups, including sample weight, and sample selection [15]. Sample weight approaches are based on this rule that weight and importance of each sample by the time reduces [15] [132]. Sample selection methods are more applicable, and sliding data window in them could be fixed or dynamic/moving [15] [112] [133] [134] [135]. Adaptive window approach in [136] uses a sliding window with variable size. In the method, size of the window is adjusted with the rate of observed CD thus the algorithm dynamically alters the window size.

In some CD detection methods an AC is applied; AC monitors samples to assign them to one of the available classes, and with monitoring CAS of them the CD in the arriving samples could be traced [135]. In [137], proposed algorithm for handling CD has two classifiers: A stable and a reactive one. The stable learner predicts based on all of the seen samples while the reactive one predicts based on a window of recent samples. By monitoring difference in accuracy between the two learners over the window, the CD is traced and the stable classifier is replaced by the reactive classifier.

In this chapter, DW algorithm is proposed that is a supervised algorithm of sample selection family methods for FD updating. It dynamically tunes the sliding window size considering current concepts of the process. DW is suggested for covering weakness of FW (sequence-based window) that possibly forgets the samples that are not redundant yet, and a flaw of IL algorithm that keeps all the redundant samples (information). Furthermore, in order to exploit advantageous of the IL and DW, simultaneously, ILDW algorithm is suggested. ILDW learns incrementally from seen samples, and after a TI, the classifier is retrained in order to forget redundant samples. The proposed algorithms, DW and ILDW, are compared with the FW, IL, Non-Updated Classifier (NUC) and Non-Incremental Learning (NIL) algorithms. NUC refers to the algorithm in which the classifier is trained initially, and it is applied for classifying new arriving samples without any updating. In NIL algorithm, all the samples are applied for retraining the classifier after a predefined TI. In order to compare various algorithms in terms of accuracy and CPU time of training dimensionless score, called MGM, is defined; the greater assigned MGM score is the more efficient algorithm is.

3.1.2 Methodology

It is supposed that samples arrive as a batch of dataset, X_k; “k” is counter of TI and each dataset contains fixed number of samples, I, that is a collection of $j = 1,2, \dots\dots J$ features, Equation (3. 1):

$$X_k = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{kI1} & \cdots & x_{kIJ} \end{pmatrix}, \mathrm{k} = 1,2, \ldots \ldots \mathrm{K}, \qquad \mathrm{i} = 1,2, \ldots \ldots \mathrm{I}, \qquad \mathrm{j} = 1,2, \ldots \ldots \mathrm{J} \tag{3. 1}$$

For IL approaches, the updating problem is to select specific samples in each batch of dataset, X_k, while for sample based approaches the updating problem is efficiently selecting data window, WI_k, in each TI, Equation (3. 2):

$$\begin{gathered} WI_k = [X_{k-r}, \ldots . X_{k-1}, X_k] \;\; \forall k, r \in \mathbb{N} \;\; \delta\, k - r \geq 1 \\ r = card\,(WI_k) - 1 \end{gathered} \tag{3. 2}$$

In which ℕ is the sign of natural numbers, and card stands for cardinality.

In order to compare different updating methods, MGM score that is dimensionless is defined, Equation (3. 3):

$$\begin{gathered} MGM_u = \omega \times \frac{\overline{CAS}_u}{\overline{CAS}_{ref}} - \frac{\overline{CPU}_u}{\overline{CPU}_{ref}} \\ u \in \{FD\ updating\ methods{:}\ NIL, NUC, IL, DW, ILDW\}, \\ ref = Reference\ method \end{gathered} \tag{3. 3}$$

In which $\overline{CAS}$ and $\overline{CPU}$ are average of CAS and average of CPU time; CAS is considered F1 score. Value of ω must be tuned for giving weight to CAS against CPU time.

Among developed classifiers, SVM appropriately fits for incremental learning because of its theoretical foundations [034] [035] thus it is selected as the classifier. The following sections explain the applied FD updating approaches, data preparation as well as hidden context scenarios and results.

3.1.3 Updating Fault Detection System with Classification Techniques:

For assessment and comparative purposes, three updating methods are studied that are NIL, NUC and IL while the classifier in all is SVM.

3.1.3.1 Non-Incremental Learning (NIL):

This is a typical and simple way of training and updating the classifiers with arriving samples/datasets. In the algorithm new dataset, X_k, is integrated with the seen datasets and learning phase with the WI_k must be done from scratch, Equation (3. 4):

$$WI_k = \{X_1, \ldots\ldots X_{K-2}, X_{K-1}, X_k\} \tag{3. 4}$$

In this algorithm, all samples have the same weight and there is no strategy for discarding and forgetting some parts of them for reducing CPU time of training [138]; the purpose of assessing this algorithm is to determine a reference approach of updating.

3.1.3.2 Non-Updated Classifier (NUC):

For the NUC the classifier is trained initially with the first dataset, X_1, and it monitors samples of new datasets without updating. The purpose of assessing this approach is to determine a reference for not updated classifier as well as comparing other updating approaches with this algorithm.

3.1.3.3 Incremental Learning (IL):

In IL algorithm, at each step of updating only a subset of the data is considered. Syed et al. [139] for the first time proposed an algorithm for IL with SVM. In the algorithm, new incoming samples are combined with original support vectors if they violate the KTT conditions. New samples satisfying KKT conditions are supposed to have no influence over updating the hyperplane. Accordingly, support vectors are chosen among original support vectors and new samples that violate KTT [32] thus a specific set of samples from each dataset are selected. In order to improve performance of the IL algorithm for updating SVM in [140] a promising approach is presented. In the presented algorithm, specific samples are discarded while the classification accuracy is kept within an acceptable range and training speed is improved. Hyperplane-distance SVM is an algorithm developed in [138] for improving the performance of the IL. In this algorithm, in the final training set there are three groups of samples. The first group of samples are support vectors of the classifier, the second group of samples are those in the newly available dataset that violate the KKT conditions, and the third group of samples are those that are between the center of each class and the hyperplane.

3.1.4 Updating Fault Detection System with Sample Based Approaches

Two sample based approaches are also applied for updating the FD: FW and DW, which are explained in the following subsections.

3.1.4.1 Dynamic Sliding Window (DW):

The DW algorithm uses a MC and an AC. The latest arriving dataset is benefitted for two purposes; it is attached to the WI_K as the last element, and it is exploited for retraining the AC. In the DW algorithm a Threshold (Tr) is defined that is applied for adjusting the cardinality of the WI_k. The Tr is the minimum accuracy of the AC on the tested dataset that must be met and it is tuned experimentally.

In the kth TI, the dataset X_k is applied for retraining the AC; then the AC is employed for classifying and testing the first dataset of the WI_k, which is X_{k-r}. If the Tr is met no more action is required, otherwise X_{k-r} will be discarded from WI_k and the next subsequent dataset, X_{k-r+1}, will be classified. If the accuracy of AC on X_{k-r+1}satisfies the Tr, it is kept in the WI_k and the MC will be retrained. Datasets that remain in the WI_k are supposed to be equally important and they will be used for (re)training and updating the MC.

3.1.4.2 Fixed Sliding Window (FW):

Another type of the sample based approaches is FW. In this type of sliding window algorithm, Adjusted Cardinality (ACr) parameter is fixed and it must be tuned, experimentally. In the algorithm at the kth TI by adding $X_k$to the WI_k as the last dataset, the first dataset in WI_k will be removed and the MC will be retrained with the WI_k, Equation (3. 5):

$$WI_k = \{X_{k-ACr+1}, \ldots\ldots X_{k-2}, X_{k-1}, X_k\} \tag{3. 5}$$

3.1.5 Updating Fault Detection System with Combination of Sample Based Approaches and Classification Techniques

3.1.5.1 Incremental Learning Dynamic Window (ILDW)

Incremental Learning Dynamic Window (ILDW) is a hybrid of IL and DW methods in order to exploit advantages of both. The ILDW algorithm is similar to DW, but between two retraining tasks of the MC, it is adjusted based on IL algorithm. Figure 3. 1 shows all the applied algorithms for FD updating.

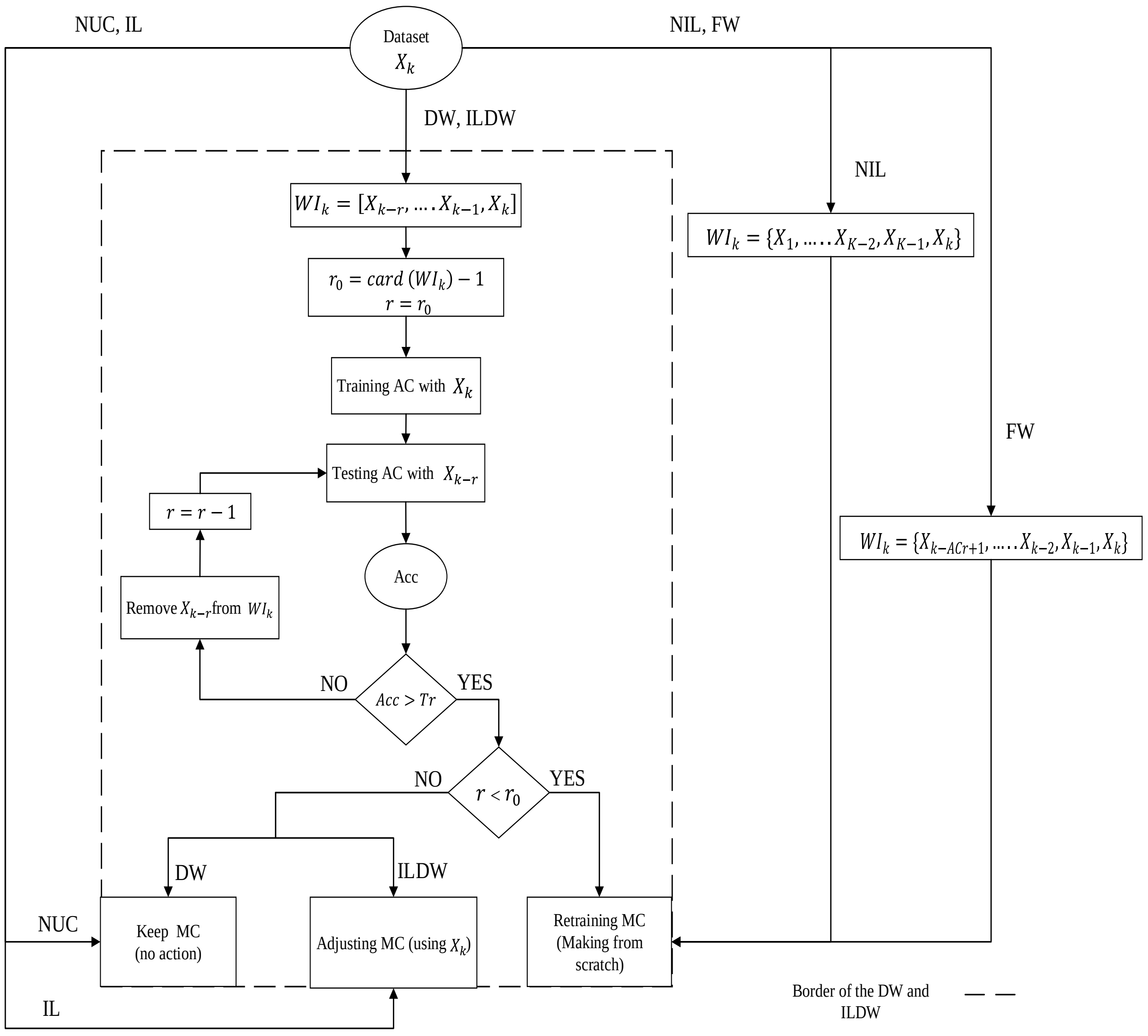
NUC, IL
Dataset X_k
NIL, FW
DW, ILDW
NIL
$WI_k = [X_{k-r}, \ldots X_{k-1}, X_k]$
$WI_k = \{X_1, \ldots X_{K-2}, X_{K-1}, X_k\}$
$r_0 = card\,(WI_k) - 1$
$r = r_0$
Training AC with X_k
FW
Testing AC with X_{k-r}
$r = r - 1$
$WI_k = \{X_{k-ACr+1}, \ldots X_{k-2}, X_{k-1}, X_k\}$
Acc
Remove X_{k-r} from WI_k
NO
$Acc > Tr$
YES
NO
$r < r_0$
YES
DW
ILDW
NUC
Keep MC (no action)
Adjusting MC (using X_k)
Retraining MC (Making from scratch)
IL
Border of the DW and ILDW

Figure 3. 1 Applied algorithms for the supervised FD updating.

3.1.6 Hidden Context Scenarios and Datasets

Six hidden context scenarios are considered for simulating CD in the process. In scenario 1 and scenario 3, the hidden context changes linearly while in the rest of the scenarios the hidden context has nonlinear profiles. Scenario 1 and scenario 2, Figure 3. 2, are employed for tuning parameters of the updating methods while the rest of the scenarios, Figure 3. 3, are applied for testing and validating.

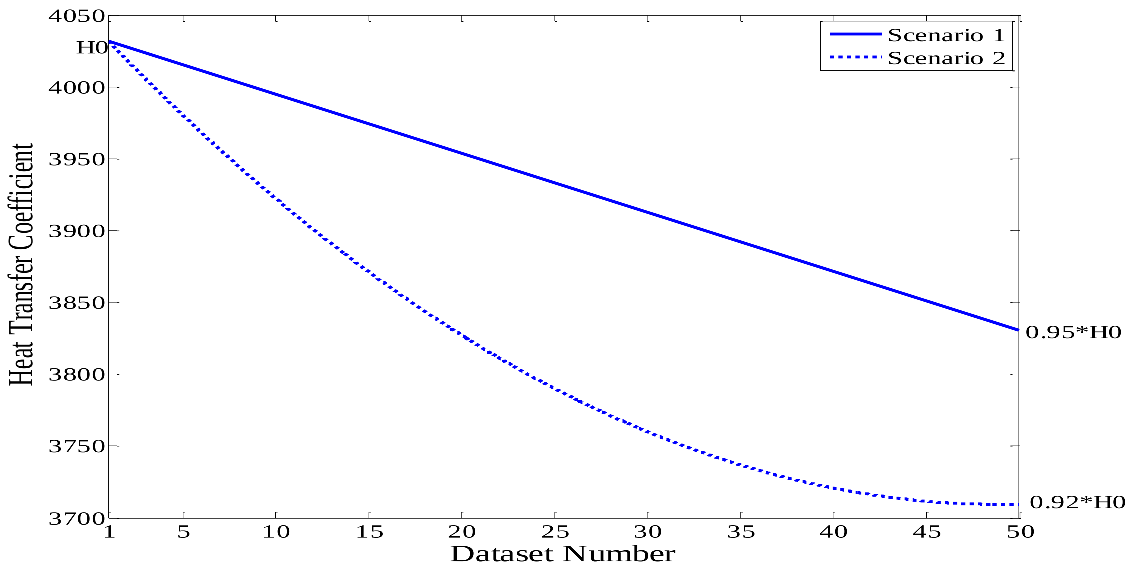

Figure 3. 2. Hidden context scenarios for tuning parameters of FD updating methods.

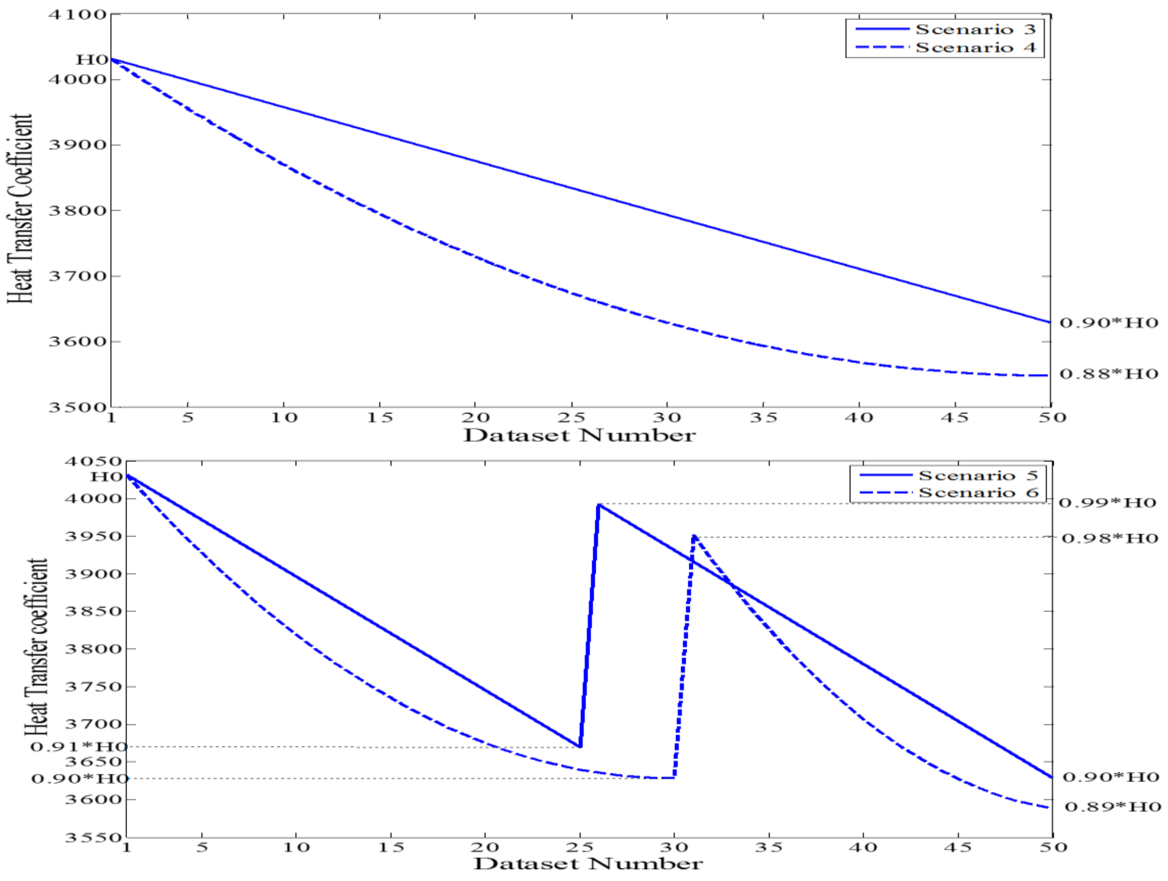

Figure 3. 3. Hidden context scenarios for testing FD updating methods.

For all these hidden context scenarios, 50 datasets are prepared. Each dataset contains 1000 normal and faulty samples, and each dataset has four features. The fault is a step change in the inlet concentration of the reactant. The normal inlet concentration of the reactant C_A equals to 5.1+v_o *mol/L* whereas v_o is a Gaussian noise; $v_o \approx \mathcal{N}(\mu = 0, \sigma = 0.045\ mol/L)$, and in the faulty condition C_A changes to 5.13+v_o *mol/L*.

3.1.7 Results

In the following sections, first, parameters of the updating algorithms are tuned, and then they are validated and compared.

3.1.7.1 Model Tuning

Scenario 1 and scenario 2 are applied for tuning parameters of the FD updating algorithms for having maximum MGM score. For calculating MGM score, ω and reference method are considered two and NIL, respectively.

3.1.7.2 Classification Based Techniques

Table 3. 1 shows results of scenario 1 and scenario 2 on three classification based techniques. These approaches do not have a parameter for tuning, but their performances are compared based on average F1 score, $\overline{F1}$, average CPU time (s), $\overline{CPU}$, average Number of the Support Vectors (NSV), $\overline{NSV}$, and MGM score. Figure 3. 4 compares performance of the three techniques; based on MGM score NUC is more efficient than the rest because required CPU time in NIL and IL is inefficiently high.

Table 3. 1. Performance of the classification based FD updating techniques for scenario 1 and scenario 2.

	Scenario 1			Scenario 2		
	NIL	NUC	IL	NIL	NUC	IL
$\overline{F1}$ (%)	80.31	69.7	79.95	77.95	67.69	77.61
$\overline{CPU}\ (s)$	24.52	0.03	15.42	25.84	0.03	17.45
$\overline{NSV}$	20231.08	1000	19358.72	21336.94	1000	20369.64
MGM Score	1.00	1.73	1.36	1.00	1.74	1.32

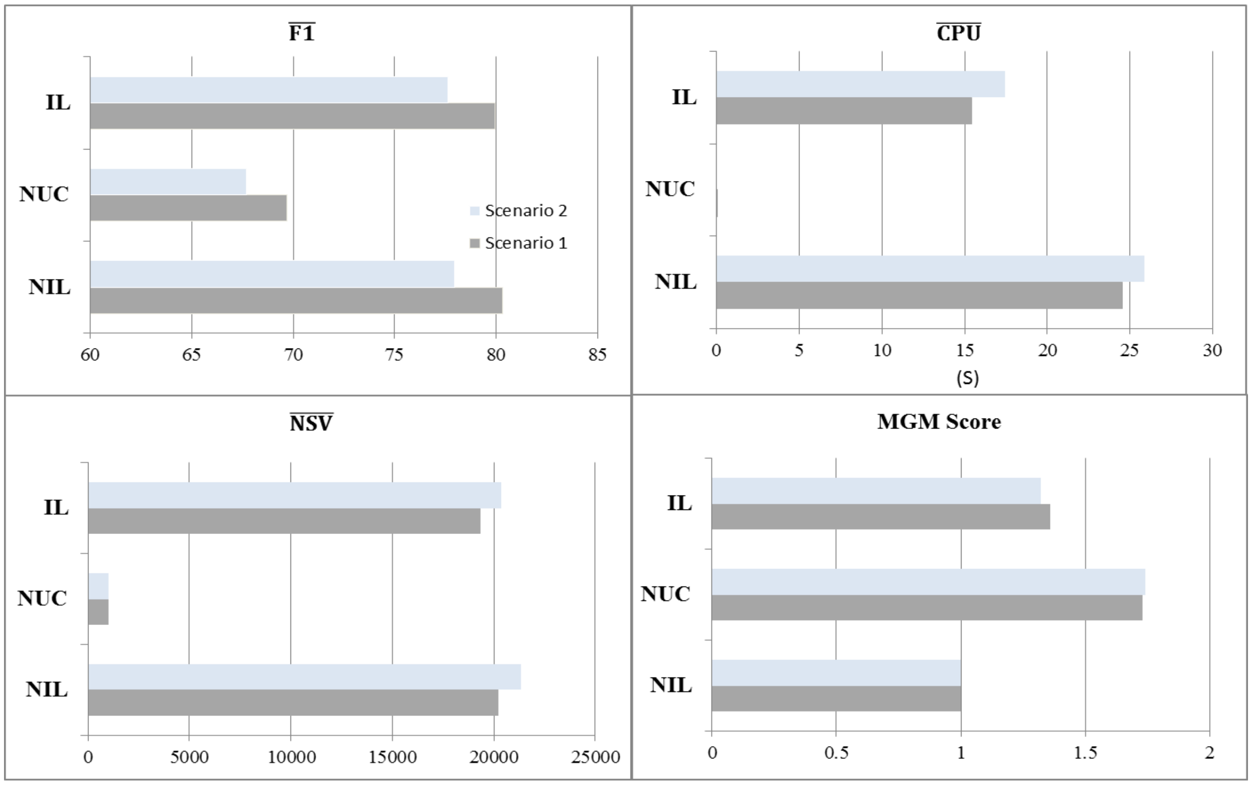

Figure 3. 4. Performance of the classification based FD updating techniques for scenario 1 and scenario 2.

3.1.7.3 Dynamic Sliding Window (DW):

For investigating effects of Tr on the DW algorithm eight different Trs, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% with scenario 1 and scenario 2 are applied, Table 3. 2. The maximum tested value for Tr in both scenarios is 70% because accuracies have not passed this amount. With the Tr of 70% the maximum MGM scores for both scenarios are attained, Table 3. 2 thus 70% is selected as the value of Tr. In Table 3. 2, $\overline{card}$ refers to mean length of the window, and by increasing Tr it decreases. Figure 3. 5 presents effects of the Tr on MGM score, average scaled F1 score, and average CPU time. Average scaled F1 score and CPU time are calculated with the Equation (3. 6), in which reference method is considered NIL:

.

$$\begin{gathered} Averaged\ scaled\ \overline{F1}_u = \frac{\overline{F1}_u}{\overline{F1}_{ref}} \\ Averaged\ scaled\ \overline{CPU}_u = \frac{\overline{CPU}_u}{\overline{CPU}_{ref}} \\ u \in \{FD\ updating\ methods\}\ , ref\ = Reference\ method \end{gathered} \tag{3. 6}$$

Table 3. 2 Effects of Tr on MGM score, average scaled F1 score and CPU time in DW algorithm.

Tr (%)	$\overline{card}$	$\overline{F1}$ (%)	$\overline{CPU}$ (s)	$\overline{NSV}$	MGM Score
Scenario 1					
10	0.40	79.05	12.19	15068.0	1.47
20	0.40	79.05	12.04	15025.83	1.48
30	0.38	78.98	11.54	14707.83	1.50
40	0.37	78.56	10.81	14306.97	1.52
50	0.37	78.28	10.16	13907.79	1.54
60	0.29	76.79	6.06	11011.46	1.67
70	0.03	71.79	0.03	1000.00	1.79
80	N/A	N/A	N/A	N/A	N/A
Scenario 2					
10	0.29	77.68	6.22	10869.48	1.75
20	0.28	77.48	5.94	10585.79	1.76
30	0.27	77.45	5.88	10481.65	1.76
40	0.26	77.26	5.78	10303.61	1.76
50	0.26	77.41	5.27	9869.77	1.78
60	0.20	77.17	3.2	7779.0	1.86
70	0.03	72.70	0.03	1000.0	1.86
80	N/A	N/A	N/A	N/A	N/A

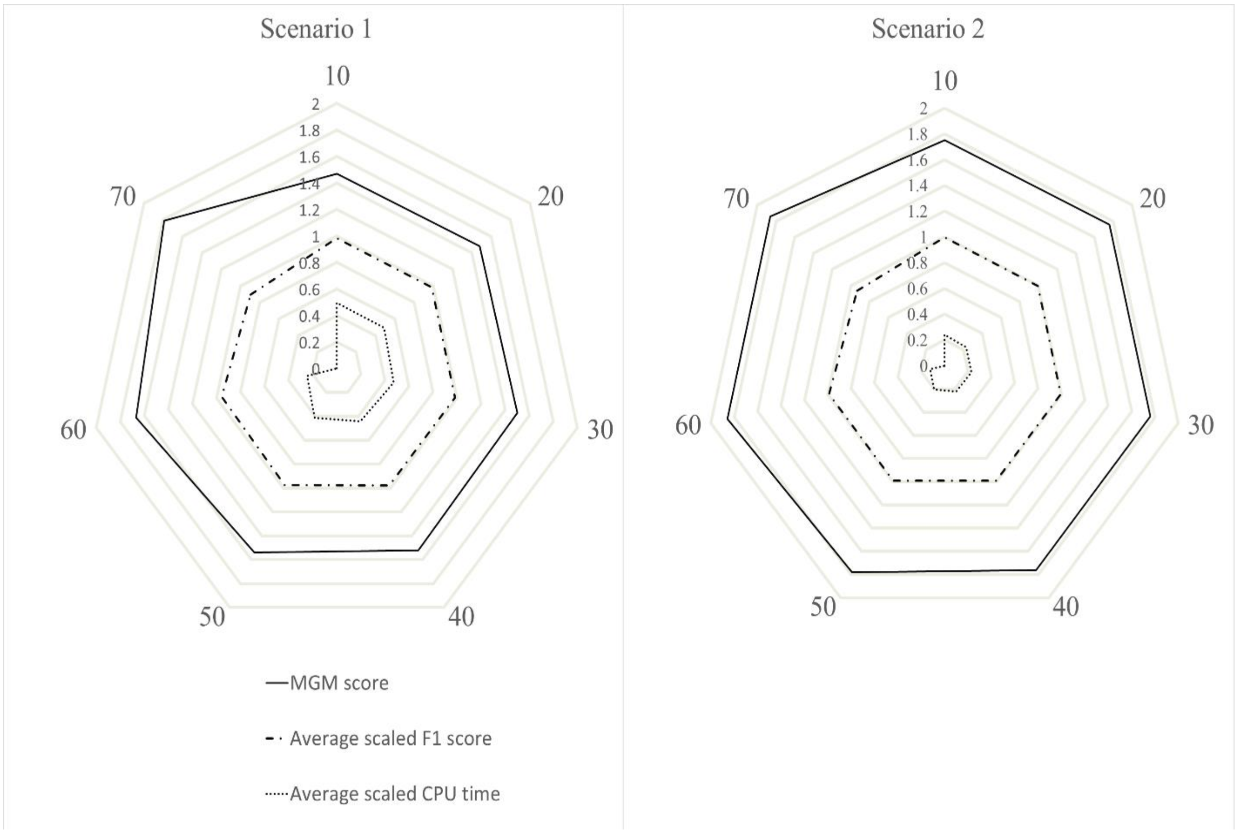

Figure 3. 5. Effects of Tr on MGM score, average scaled F1 score and CPU time in DW algorithm.

3.1.7.4 Fixed Sliding Window (FW):

In the FW algorithm, the cardinality of the WI_k is a critical issue therefore effects of the cardinality with eight different values for ACr with scenario1 and scenario 2 is studied, Table 3. 3. Maximum MGM scores for scenario 1 and scenario 2 are reached while ACr is three and four, respectively. Figure 3. 6 presents effects of the ACr on MGM score, average scaled F1 score, and CPU time that are calculated by Equation (3. 6).

Table 3. 3. Effects of ACr on MGM score, average scaled F1 score and CPU time in FW algorithm.

ACr	$\overline{F1}$ (%)	$\overline{CPU}$ (s)	$\overline{NSV}$	MGM Score
Scenario 1				
1	67.03	0.12	1977.55	1.66
2	69.85	0.29	2936.77	1.73
3	73.77	0.50	3875.59	1.82
4	73.97	0.85	4790.77	1.81
5	74.57	1.16	5551.91	1.81
10	75.40	2.93	8020.18	1.76
15	76.18	5.65	10600.22	1.67
20	77.09	8.44	12824.77	1.58
Scenario 2				
1	67.55	0.12	1977.55	1.73
2	71.13	0.30	2936.77	1.81
3	74.59	0.51	3874.81	1.89
4	75.50	0.83	4748.89	1.91
5	75.94	1.17	5500.89	1.90
10	76.82	2.98	7964.67	1.86
15	78.07	5.70	10635.18	1.78
20	78.58	8.66	13025.71	1.68

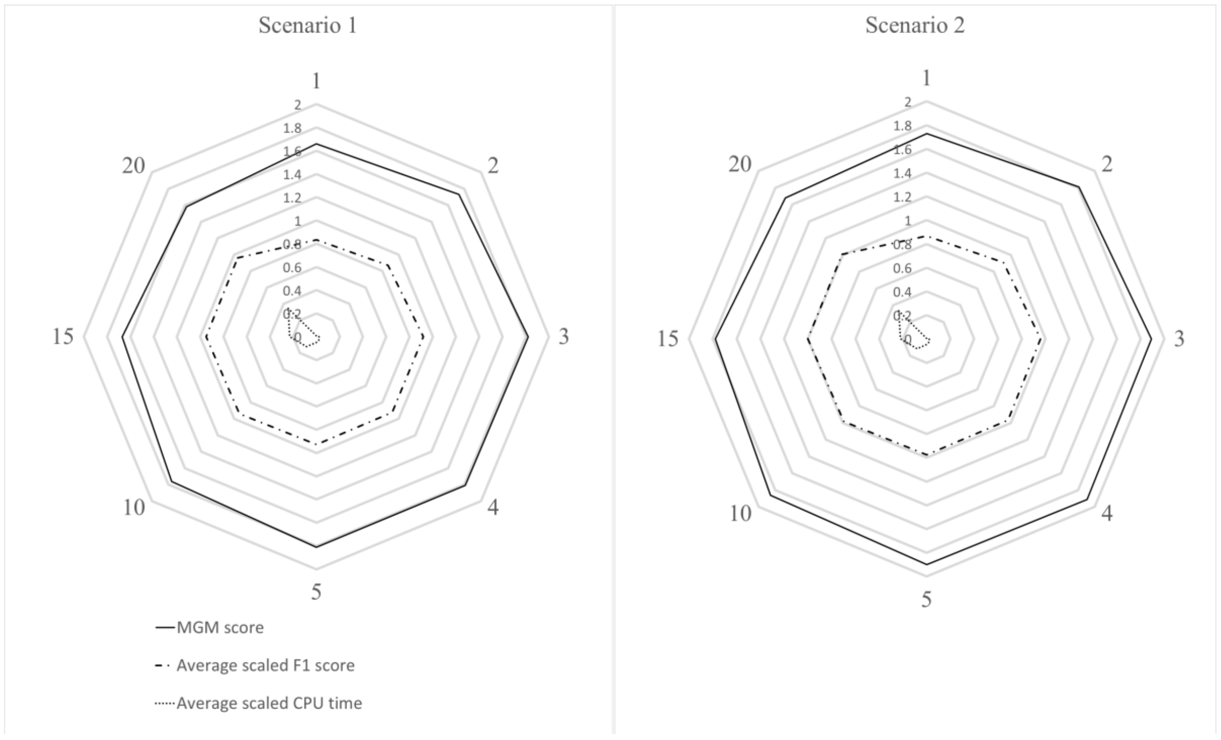

Figure 3. 6. Effects of ACr on MGM score, average scaled F1 score and CPU time in FW algorithm.

3.1.7.5 Incremental Learning Dynamic Window (ILDW)

Table 3. 4 presents performance of the ILDW with different Trs for scenario 1 and scenario 2. The maximum MGM scores for both scenarios are attained while Tr is tuned on 70%. Figure 3. 7 presents effects of the Tr on average MGM score, average scaled F1 score and average scaled CPU time that are calculated by Equation (3. 6).

Table 3. 4. Effects of Tr on MGM score, average scaled F1 score and CPU time in ILDW algorithm.

Tr (%)	$\overline{card}$	$\overline{F1}$ (%)	$\overline{CPU}$ (s)	$\overline{NSV}$	MGM Score
Scenario 1					
10	0.398	78.30	8.81	14609.80	1.59
20	0.397	78.30	8.76	14596.08	1.59
30	0.390	78.26	8.49	14357.36	1.60
40	0.382	78.05	8.10	14071.08	1.61
50	0.373	77.85	7.78	13757.88	1.62
60	0.291	76.63	4.90	11035.16	1.71
70	0.041	75.26	1.49	5707.60	1.81
80	0.039	80.26	15.63	19358.72	1.36
90	0.039	80.26	15.64	19357.26	1.36
Scenario 2					
10	0.286	78.58	4.97	10804.6	1.82
20	0.278	78.43	4.76	10540.32	1.83
30	0.275	78.42	4.58	10438.68	1.83
40	0.270	78.29	4.55	10270.16	1.83
50	0.260	77.83	4.31	9924.2	1.83
60	0.204	77.36	2.90	8009.16	1.87
70	0.042	74.75	1.01	4720.28	1.88
80	0.039	78.32	17.34	20369.64	1.34
90	0.039	78.32	17.35	20369.46	1.34

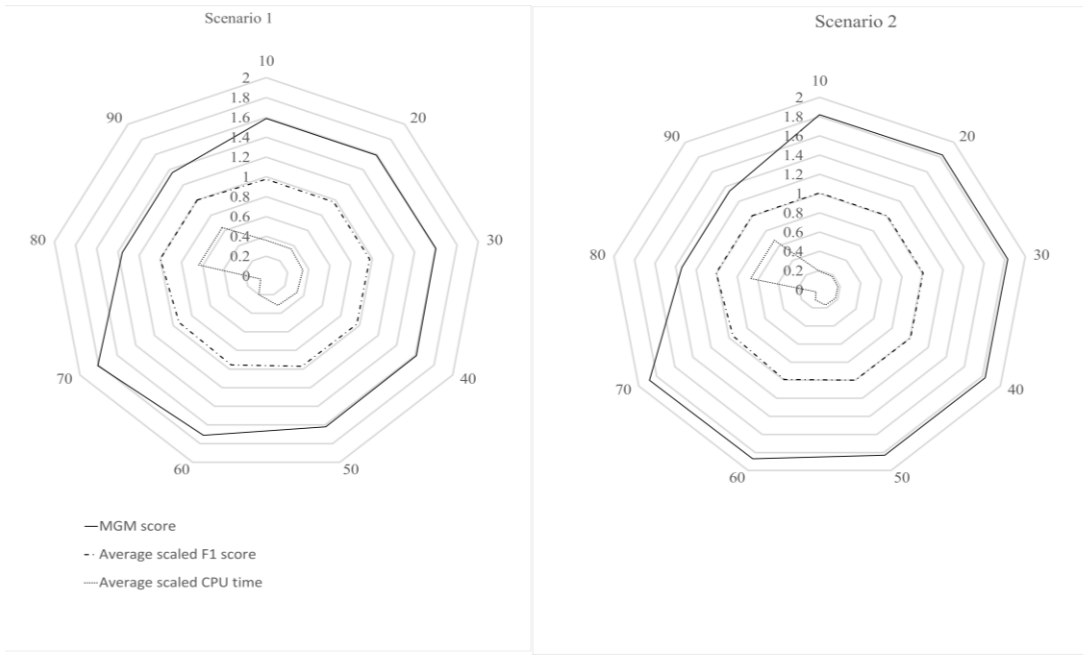

Figure 3. 7. Effects of Tr on MGM score, average scaled F1 score and CPU time in ILDW algorithm.

3.1.8 Result for Scenario 3, 4, 5, 6

In order to maximize MGM score, the optimum value of parameters for each algorithm are implemented. For FW algorithm, ACr is tuned to be three and for DW and ILDW the Tr is adjusted on 70%. FD updating algorithms are compared by four scenarios, Table 3. 5 presents the results of six updating algorithms with these scenarios.

For scenario 3 and scenario 4 ILDW has better performance in terms of MGM score, by 1.88 and 1.99, respectively. In the scenario 3, the assigned MGM score to the NUC is 1.74. This implies that for scenario 2 updating algorithms that their assigned MGM scores are less than 1.74 are not efficient algorithms, and this could be generalized for the rest of the scenarios. In scenario 3, although the $\overline{F1}$ score for the IL is 79.07%, its $\overline{CPU}$ time of training has a large value, 16.72 s, which makes IL an inefficient algorithm. In scenario 2, minimum $\overline{NSV}$ is related to NUC and then DW while in the scenario 4 the simplest (minimum $\overline{NSV}$) model after NUC is FW.

In scenario 4, ILDW not only has the highest MGM score, but also it achieves to the highest $\overline{F1}$ score. For the scenario 5 and 6 assigned MGM scores to DW have the greatest values. In the scenario 5 the highest $\overline{F1}$ score is obtained by IL, but consumed $\overline{CPU}$ time of training, 17.99 s, puts it far from an efficient algorithm. In the scenario 5, assigned MGM score to the FW algorithm is less than NUC that implies weakness of FW while hidden context scenario has unpredictable profiles with the sharp changes.

In the scenario 6, similar to other scenarios the highest $\overline{F1}$ score is achieved by IL with the highest $\overline{CPU}$ time of training. Figure 3. 8 compares presented FD updating algorithms in terms of MGM score, average scaled F1 score, and average CPU time, calculated by Equation (3. 6).

Table 3. 5. Comparison of six FD updating algorithms by four scenarios.

Scenario 3				
	$\overline{F1}$ (%)	$\overline{CPU}$ (s)	$\overline{NSV}$	MGM Score
NIL	78.42	25.45	21185.32	1.00
NUC	68.27	0.04	1000	1.74
IL	79.07	16.72	20105.84	1.36
FW(3)	74.29	0.52	3875.18	1.87
DW	71.49	0.049	1183.67	1.82
ILDW	75.08	0.85	4412.3	1.88
Scenario 4				
NIL	76.74	26.35	21812.0	1.00
NUC	67.34	0.03	1000	1.75
IL	77.29	17.83	20885.96	1.34
FW(3)	76.17	0.51	3867.10	1.97
DW	77.50	0.98	4686.36	1.98
ILDW	77.80	0.91	4670.16	1.99
Scenario 5				
NIL	78.39	25.71	21640.56	1.00
NUC	67.76	0.03	1000	1.73
IL	79.00	17.99	20797.02	1.32
FW(3)	68.17	0.51	3845.18	1.72
DW	77.18	0.30	2895.87	1.96
ILDW	74.10	0.34	3107.2	1.88
Scenario 6				
NIL	76.37	25.92	22121.58	1.00
NUC	67.11	0.03	1000	1.76
IL	77.95	18.75	21208.4	1.32
FW(3)	70.40	0.51	3839.38	1.82
DW	76.5	0.38	2953.02	1.99
ILDW	74.92	0.47	3322.72	1.94

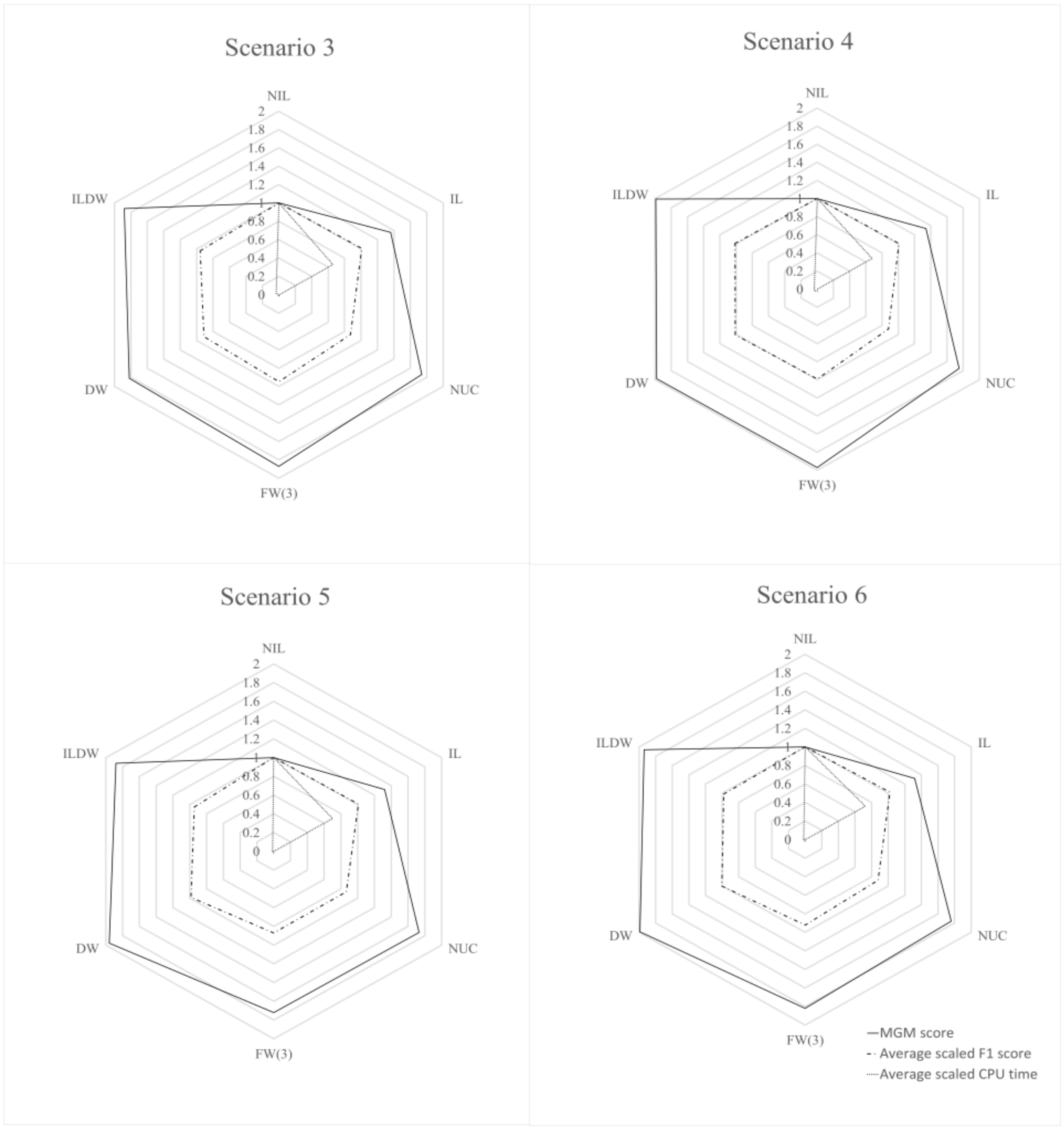

Figure 3. 8. Comparison of six FD updating algorithms by four scenarios.

3.1.9 Conclusions

Two types of FD updating algorithms, classification based and sample based, for handling CD in the process are investigated. Moreover, in one of the proposed updating algorithm, ILDW, these two approaches are simultaneously employed. The aim of the proposed algorithms, DW and ILDW, is providing data window for, dynamically and efficiently, updating the FD system. For comparing FD updating methods, MGM score is defined based on weighted F1 score and CPU time of training. Six FD updating algorithms, including IL, NIL, NUC, FW, DW and ILDW are analyzed with four scenarios of hidden contexts. In all the scenarios, DW and ILDW have higher MGM score in comparison with the other algorithms. The results prove the efficiency of the suggested algorithms for forgetting redundant information and FD updating.

3.2 Characterizing the Concept Drift

In this section, a methodology for implicit quantification of the change in CD amount is provided. The method consists of an incremental learned classifier and non-automatic clustering methods. The proposed algorithm would provide information for updating either by adjusting or by retraining.

3.2.1 Introduction

Existence of the CD could lead to the deterioration of the FDD performance thus various methods have been developed for FDD updating in order to manage CD. Although most of the methods available for FDD updating are able to update it and keep FDD accuracy in the acceptable range, they usually do not provide information about changes in CD amount (magnitude). The estimation of changes in amount of the CD in the process is of essential importance because the continuous FDD updating, disregarding magnitude of the CD, could easily drive the process to unsafe operating conditions.

In the explicit FDD updating methods, some statistical characteristics of the samples for CD detection are traced. Among them DDM [135], EDDM [141] and STEPD [130] methods are more common [15] [142]. DDM detects CD by analyzing the error rate in classification; EDDM is similar, but uses the distance between two classification errors rather than the error rate; STEPD monitors the predictions, and it has two thresholds for significance levels of the CD detection and warning about CD. Classification accuracy is among widely used CD indicators especially by integrated classification algorithms. The classification accuracy could be calculated with any of the performance indices that are applied in the literature [15] [143].

In [144], for monitoring changes in CD amount, two transformation functions, Hotelling's $T^2$and Q statistic, are applied. In the proposed framework, the arriving datasets are initially classified by incremental learned classifiers. Then, samples are transformed into new space with the transforming functions. The transforming functions are provided for each class with samples of the reference dataset, which is an arbitrary dataset. Next, samples of the reference dataset and arriving dataset for each class, in new space, are compared based on RMSE. For the applied scenarios, results show that the proposed framework could detect the CD in the process even for considerably trivial of it.

This section presents a methodology for supervised tracing of changes in the CD amount. In regard to "precision" definition a new index is suggested that could provide information about changes in the CD amount. In the methodology, after classifying arriving datasets, and with the aid of non-automatic clustering samples of each class are compared to the related class of reference dataset.

The proposed framework is tested on CSTR case study while reducing value of heat transfer coefficient is a cause of the CD.

.

3.2.2 Methodology

Assume datasets arrive in the fixed TI, Equation (3. 7):

$$X_k = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{KI1} & \cdots & x_{kIJ} \end{pmatrix}, \qquad \boldsymbol{x}_{ki} = (x_{ki1}, x_{ki2} \ldots\ldots..x_{kiJ}) \tag{3. 7}$$

The proposed framework is considered supervised thus there is a function such as L that assigns each sample to one class, Equation (3. 8):

$$L(\boldsymbol{x}_{ki}) = c \ \ , c \in \{0,1, \ldots.C\} \quad \forall\, \boldsymbol{x}_{ki} \tag{3. 8}$$

In which, 0 is label of normal class.

Accordingly, each dataset X_k could be classified into different classes, Equation (3. 9):

$$X_k = \bigcup_{\forall c} \Gamma_{kc} = \{\Gamma_{k0}, \Gamma_{k1}, \ldots..\Gamma_{kC}\} \tag{3. 9}$$

In which Γ_{kc} is defined by Equation (3. 10):

$$\Gamma_{kc} = \{\boldsymbol{x}_{ki} | L(\boldsymbol{x}_{ki}) = c\} \tag{3. 10}$$

Each Γ_{kc} could have a different number of samples, Equation (3. 11):

$$\Gamma_{kc} = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{ka1} & \cdots & x_{kaJ} \end{pmatrix} \tag{3. 11}$$

An arbitrary dataset as reference dataset X_{k^*} must be selected, Equation (3. 12):

$$X_{k^*} = \begin{pmatrix} x_{k^*11} & \cdots & x_{k^*1J} \\ \vdots & \ddots & \vdots \\ x_{k^*I1} & \cdots & x_{k^*IJ} \end{pmatrix}, X_{k^*} = \bigcup_{\forall c} \Gamma_{k^*c} = \{\Gamma_{k^*0}, \Gamma_{k^*1}, \ldots .\Gamma_{k^*C}\},$$
$$\Gamma_{k^*c} = \begin{pmatrix} x_{k^*11} & \cdots & x_{k^*1J} \\ \vdots & \ddots & \vdots \\ x_{k^*b1} & \cdots & x_{k^*bJ} \end{pmatrix} \quad 3.\ 12$$

In the proposed framework, for each class of each dataset the Augmented Dataset (AG) must be made, Equation (3. 13):

$$AG_{kc} = \Gamma_{k^*c} \cup \ \Gamma_{kc} = \begin{pmatrix} x_{k^*,1,1} & \cdots & x_{k^*,1,J} \\ \vdots & \ddots & \vdots \\ x_{k^*,b,1} & \cdots & x_{k^*,b,J} \\ x_{k,b+1,1} & \cdots & x_{k,b+1,J} \\ \vdots & \ddots & \vdots \\ x_{k,b+a,1} & \cdots & x_{k,b+a,J} \end{pmatrix}, \quad 3.\ 13$$
$$A_k = \bigcup_{\forall c} A_{kc} = \{AG_{k0}, AG_{k1}, \ldots \ldots AG_{kC}\}$$

In the next step, the AG_{kc} must be processed with the Non Automatic Clustering Function (NACF), Equation (3. 14):

$$(m_{kc}, n_{kc}) = NACF(AG_{kc}) \quad 3.\ 14$$

In which m_{kc}, n_{kc} are made with non-automatic clustering of AG_{kc} for having two clusters, $AG1_{kc}$ and $AG2_{kc}$, Equation (3. 15):

$$AG1_{kc} = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{km1} & \cdots & x_{kmJ} \end{pmatrix}, \quad AG2_{kc} = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{kn1} & \cdots & x_{knJ} \end{pmatrix},$$
$$m > n \rightarrow \ m_{kc} = m, \quad n_{kc} = n \quad 3.\ 15$$

Value of n_{kc} implies number of samples in each class that are clustered differently from the rest of samples. Accordingly, changes in Concept Drift Amount (cda) for each class of each dataset could be calculated by Equation (3. 16):

$$cda_{kc} = \frac{n_{kc}}{m_{kc} + n_{kc}} \tag{3. 16}$$

Regarding the definition of the precision, Equation (3. 17):

$$Percision = \frac{t_p}{t_p + f_p} \tag{3. 17}$$

In which t_p is TP and f_p is FP. So P_{kc} would be defined with Equation (3. 18):

$$P_{kc} = \frac{m_{kc}}{m_{kc} + n_{kc}} \tag{3. 18}$$

Reformulating Equation (3. 16) results in Equation (3. 19):

$$cda_{kc} = 1 - P_{kc} \tag{3. 19}$$

The cda_{kc} for each class could changes between zero to one. If there is no CD in Γ_{kc}, the assigned value is zero, and if samples of Γ_{kc} are completely different from Γ_{k^*c}, its value will be one. CDA_k is defined as a set of all cda_{kc}, Equation (3. 20):

$$CDA_k = \{cda_{k0}, cda_{k1}, \dots cda_{kC}\} \tag{3. 20}$$

Values in CDA_k indicate the FD system must be either adjusted for trivial changes or retrained for significant changes. Figure 3. 9 summarize Equation (3.7) to Equation (3. 20) in which AV is Adjusting Value and RV is Retraining Value. AV and RV should be tuned, experimentally.

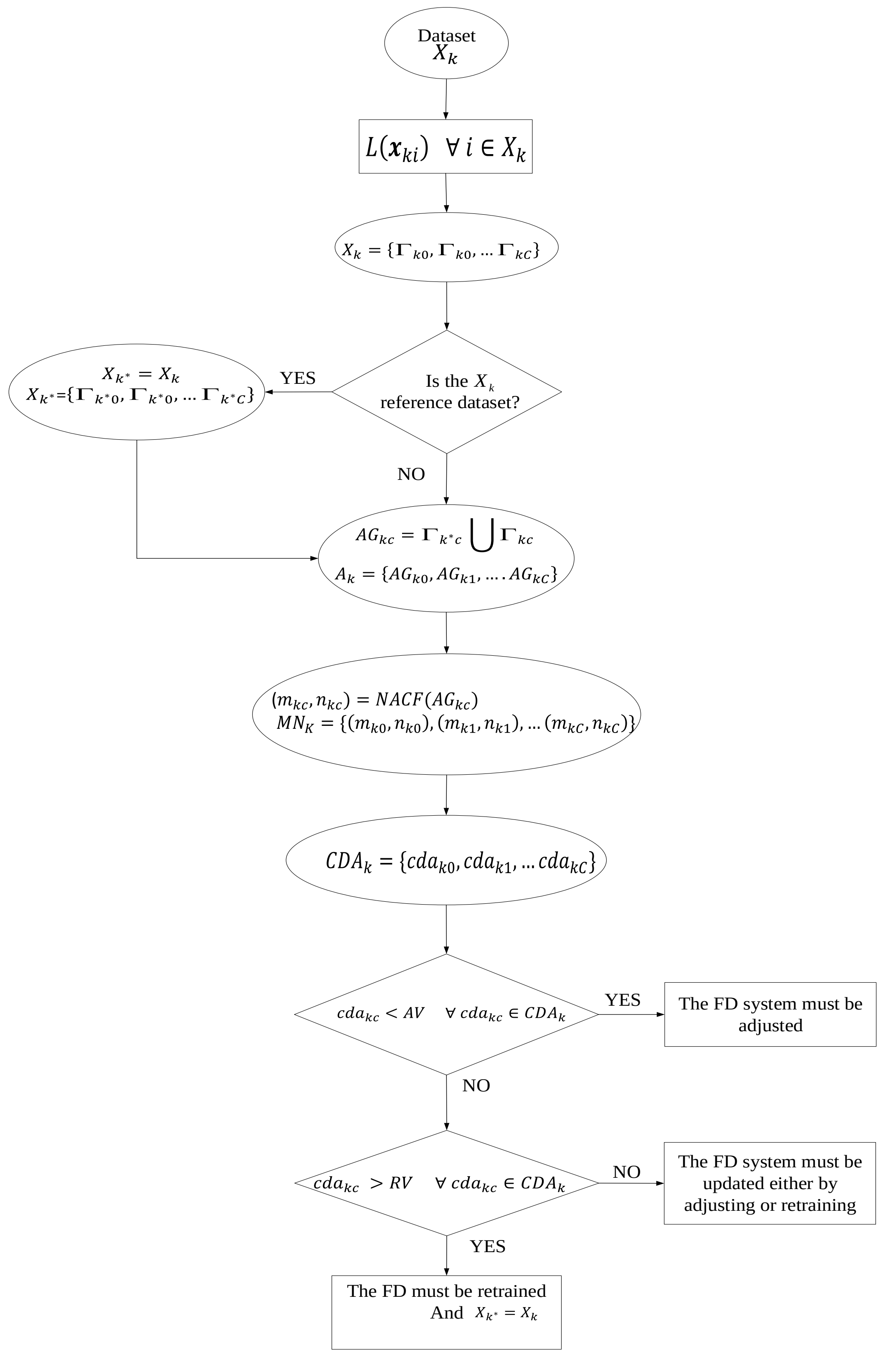
Dataset X_k
$L(\boldsymbol{x}_{ki}) \quad \forall\, i \in X_k$
$X_k = \{\boldsymbol{\Gamma}_{k0}, \boldsymbol{\Gamma}_{k0}, \ldots \boldsymbol{\Gamma}_{kC}\}$
Is the X_k reference dataset?
YES
$X_{k^*} = X_k$
$X_{k^*}=\{\boldsymbol{\Gamma}_{k^*0}, \boldsymbol{\Gamma}_{k^*0}, \ldots \boldsymbol{\Gamma}_{k^*C}\}$
NO
$AG_{kc} = \boldsymbol{\Gamma}_{k^*c} \bigcup \boldsymbol{\Gamma}_{kc}$
$A_k = \{AG_{k0}, AG_{k1}, \ldots . AG_{kC}\}$
$(m_{kc}, n_{kc}) = NACF(AG_{kc})$
$MN_K = \{(m_{k0}, n_{k0}), (m_{k1}, n_{k1}), \ldots (m_{kC}, n_{kC})\}$
$CDA_k = \{cda_{k0}, cda_{k1}, \ldots cda_{kC}\}$
$cda_{kc} < AV \quad \forall\, cda_{kc} \in CDA_k$
YES
The FD system must be adjusted
NO
$cda_{kc} > RV \quad \forall\, cda_{kc} \in CDA_k$
NO
The FD system must be updated either by adjusting or retraining
YES
The FD must be retrained And $X_{k^*} = X_k$

Figure 3. 9. Framework of monitoring changes in CD amount.

3.2.3 Case Study

The applied case study is CSTR; the normal inlet concentration of the reactant is 5.1+v_o mol/L, and $v_o \approx \mathcal{N}(\mu = 0, \sigma = 0.045\, mol/L)$ is Gaussian noise. Fault is defined as step change of the inlet concentration from its normal value to 5.13+v_o mol/L. Heat transfer coefficient ($H_0 = 4032\, kJ/h.m^2.K$) is designed to linearly decrease to reach 095*H_0 at the end of the operating time as the source of the CD. 12 datasets are collected each contains 4000 samples, and the first dataset is selected as the reference dataset, X_{1^*}.

3.2.4 Results

Figure 3. 10 presents cda_{kc} of 12 datasets; they are calculated for normal (c=0) and faulty (c=1) samples, separately. Figure 3. 10 shows the proposed algorithm is able to trace CD changes in the process over the time. For normal samples, the cda_{kc} increases from 0 in the first dataset (k=1) to 0.16 in the last dataset (k=12) while for the faulty samples the cda_{kc} increases from 0 in the first dataset (k=1) to 0.32 in the last dataset (k=12).

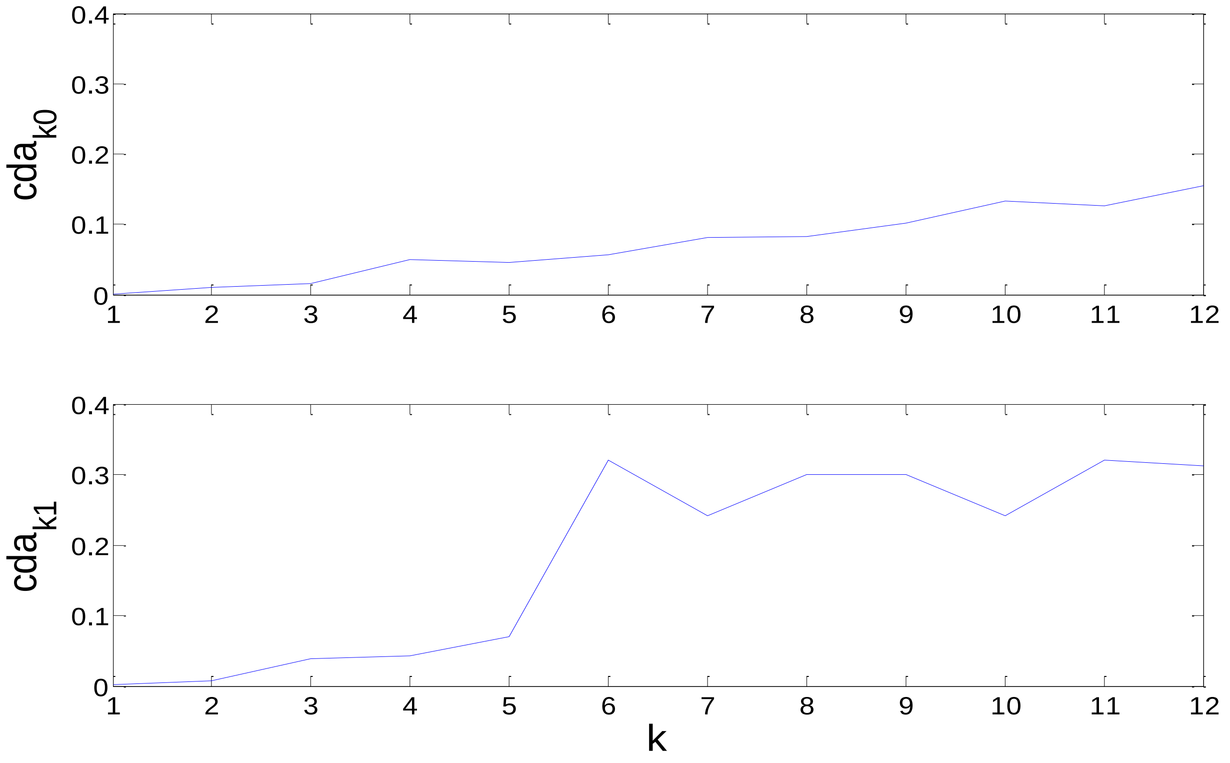

Figure 3. 10. Concept drift amount for normal (up) and fault 1 (down)

3.2.5 Conclusions

Existing methods of supervised FDD updating usually do not provide information about changes in CD amount. Therefore, in this section this necessity is addressed. The results through a benchmark case study illustrate the capabilities of the proposed framework for tracking changes of the CD amount. The framework is able to provide information along the operating time that can be used to prevent the process to reach unsafe operating conditions. As it is explained in the previous section, DW and ILDW algorithms use accuracy of the AC for deciding about retraining or adjusting the FD system. The provided framework for tracing the CD amount could be an alternative index that could assist DW and ILDW algorithms deciding about retraining or adjusting the FD system.

Chapter 4: Unsupervised Monitoring

4.1 Unsupervised FDD Updating: Handling new Faults

FDD systems for chemical processes have been studied profoundly in the literature mostly with the assumption of prior knowledge of condition labels (supervised learning). Among FDD methods, those that concentrate on the advent of novel conditions have received less attention. This problem has been commonly undertaken by ND methods with supervised approaches that are mainly developed to detect novel samples without a clear and general strategy for updating the FDD with them. This section addresses this problem by developing a hybrid automatic unsupervised data driven framework for FDD updating. It is composed of ND with OCCs to detect samples following novel patterns and automatic clustering to diagnose them according to novel clusters. An observer is incorporated for data processing and enhancing the FDD robustness. The FDD updating is performed by modifying the existing clusters and detecting new clusters for assembling models to predict them in an unsupervised automatic mode.

4.1.1 Introduction

Managing abnormal situations have received great attention in recent years mainly because of its crucial role in preventing losses and safety issues. As process plants become more modern and complicated, it is essential to continuously update FDD methods in order to keep up with them [145]. The FDD approaches require all information about the classes to diagnose normal and faulty operation modes. However, some unknown classes to FDD may exist (due to the high complexity of the monitored process), or new operation modes and classes may appear [18]. Therefore, FDD systems must be equipped with methods that make them able to detect novel classes, timely. None of the ND methods could be seen as the best one as their abilities depend on statistical properties of data [16]. A reliable ND should have a tightly closed decision boundary on available samples, and its decision boundary should update over time based on the current concepts [146].

In the literature, three terms, including ND, anomaly detection and outlier detection are used frequently; they must be clearly distinguished as they could be confused. The similarity of these three terms is that they are related to find out patterns that are different from learned patterns [74]. ND could be defined as recognition of the novel concepts that may refer to the following: Appearance of new concepts, changes in the current concepts, and presence of the noise in the current concepts [114]. In [147] and [148], ND is defined as recognition of input that differs in some respects from previous inputs. In [149], ND is defined as identification of abnormal behaviors from one regime to another. In [47], ND is addressed as a problem of identifying new patterns that are previously unseen.

Anomaly and outlier are two terms used sometimes interchangeably [148]. In [150], anomaly detection is described as a task of finding samples that violate expected patterns, but they are not necessarily novel. Additionally, in contrast to anomaly detection approaches, those detected samples by ND methods could be incorporated into the FDD models. In [151], outliers are defined as those samples that are not consistent with the majority of the samples, and in [86] they are mentioned as a type of the data abnormality. In [152], outliers are defined as samples that appear to deviate noticeably from other samples. Outlier detection methods try to find out samples that violate the normal behavior while ND methods seek for a set of samples that describe the new concepts [74].

ND approaches could be grouped as online and offline methods. For the offline ND methods three aspects are involved, including number of the classes associated with the known concepts, number of the classifiers, and supervised or unsupervised way of ND [74]. Considering number of classes, the known concept could be assumed that is made by only one class [153] or more classes [154]. Considering number of classifiers, they would be made by one classifier [155] or set of classifiers [156]. In unsupervised approaches, training phase requires only positive samples, and they could be categorized by [146]: SVM based [157], nearest neighbors based [158], clustering based [159], and probability density based [160] methods.

One quite common method for ND is applying SVM based approaches; they make decision hyperplane boundary that encloses majority of training samples. New samples that fall outside the hyperplane are considered as novel samples [146] [161]. Nearest neighbors based techniques are founded on this assumption that positive samples lie near their neighborhoods while novel samples lie far away from their neighbors [146] [162]. In clustering based methods, training samples are grouped to some clusters, then new samples are checked as if they belong to existing clusters or not. The new samples are considered as novel samples if they do not belong to available samples [163].

Probability density based approaches generate a simplified model of the training dataset distribution. In these approaches, type of the ND depends on linear or nonlinear distribution of data [48] [164]. Statistical approaches of ND are based on modeling data in consideration of its statistical properties, and estimating whether the test samples belong to the same distribution or not [16]. Statistical approaches of ND do density modeling with the training dataset and detect test samples as novel if they are in regions of low density. ND with the statistical approaches is grouped into two, including parametric and non-parametric approaches. The main difference between these two is that in non-parametric approaches there is no assumption on the statistical properties of data whereas in parametric approaches the default assumption is the Gaussian distribution of data [16]. However, for modeling more complex form of data distribution, other approaches must be applied [165] [166].

Neural network based approaches are among those common methods that are applied for the ND. The advantage of these approaches is that few parameters need to be optimized for training networks, and no prior assumptions on the properties of data are made. A consideration of these approaches is that they cannot be retrained efficiently comparing with statistical models [17]. These approaches are useful if its architecture is selected correctly; a small network will have difficulties in learning while a large network may lead to overfitting and poor performance [148] [17].

In online ND as samples arrive continuously, three tasks must be executed: Classification, ND and updating. The classification task assigns new samples to the defined classes or novel class [74]. Some approaches label the arriving samples as novel samples if the models consider them as new [167], but in another group of approaches first they are labeled as unknown, and then they are kept for more analysis [168]. In [74], the updating approaches for ND in data stream have been categorized with three criteria: First, as if the updating task is done with or without feedback; second, number of the classifiers; and third, type of the forgetting mechanism. The assumption for the methods that use the feedback is that the true labels for all samples are available. The style of updating with the different number of classifiers is different; whereas with one classifier the updating is usually based on incrementally learning approaches [167], with several classifiers the updating is based on training a new classifier and removing the old one [154].

The FDD systems must learn and update with new classes or concepts that appear in the process. However, if they only learn without forgetting any so-called old or redundant information, their efficiency will decrease by the time. The importance of the forgetting strategies increases while there are many classes, and FDD monitors the process for a long time. Nevertheless, another problem sometimes appears in which the concepts or classes that are disappeared and forgotten by FDD appear again. This issue in the literature is addressed by recurring context. Obviously, when a class or concept reappears it is effort wasting if the FDD system takes them as a novel class and learn again from the scratch. Thus, in those ND approaches that forget the disappeared classes this problem must be considered in order to reduce false alarm rate, computational effort etc. [74]. This problem is quite common, but few works have addressed it [74] [169] [170] [171].

This section presents a hybrid data driven FDD framework based on OCCs and automatic clustering allowing novelty detection and FDD updating in an unsupervised manner. Most of the researches in ND are only dedicated to detecting new classes, but not to the exploitation of this new knowledge. In order to address this issue, the proposed FDD framework automatically updates the FDD after diagnosing new class(es) by a set of OCCs and unsupervised automatic clustering. Moreover, MDK is incorporated as an observer that provides error features. The suggested FDD framework is implemented and validated with three tanks benchmark.

4.1.2 Methodology

In the following two subsections, first, the proposed unsupervised FDD updating framework is described then data processing with an observer is explained.

4.1.2.1 Problem Definition and Proposed Framework

The first part of the problem consists of a preliminary offline unsupervised classification (clustering) of the historical dataset into a collection of statistically different C process conditions (normal and faulty). Based on these C labels, unsupervised classification models, OCCs, need to be developed for efficiently classifying arriving samples. The problem is comprised of periodically updating the classification models to learn new process faults automatically.

Consider an initial dataset, X, Equation (4. 1):

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & \ddots & \vdots \\ x_{I1} & \cdots & x_{IJ} \end{pmatrix} \tag{4. 1}$$

In which $\boldsymbol{x}_i$ is, Equation (4. 2):

$$\boldsymbol{x}_i = (x_{i1}, x_{i2} \dots \dots .. x_{iJ}) \tag{4. 2}$$

Assume no simultaneous faults so that this initial dataset X may be automatically clustered into C disjoint subsets by a clustering function, $Cl(X, \beta)$. A cluster threshold parameter (β) needs to be tuned as a minimum size for each subset, Equation (4. 3):

$$Cl(X,\beta) \Rightarrow \begin{cases} A_c \subseteq X \quad \forall c : A_c \cap A_{q\neq c} = \emptyset \wedge |A_c| \geq \beta \\ B = X \backslash A \quad (A \equiv \cup A_c) \\ \bar{C}(\boldsymbol{x}_i) = c \quad \forall \boldsymbol{x}_i \in A_c \\ \bar{C}(\boldsymbol{x}_i) = -1 \quad \forall \boldsymbol{x}_i \in B \end{cases} \qquad 4.3$$

The function $\bar{C}(\boldsymbol{x}_i)$ provides the label for each sample $\boldsymbol{x}_i$. Therefore, the labeled sets A_c, and consequently the complementary set B are determined.

If the true label of each sample, $L\,(\boldsymbol{x}_i)$, was a priori known, the Clustering Accuracy (CA) of each subset could be assessed by means of Equation (4. 4), in which $\veebar$ is XOR and $\neg\veebar$ is the negation of it. As an example, Table 4. 1 illustrates the accuracy by clustering a collection of samples into arbitrary set A$_2$.

$$CA_c = \frac{\sum_{x_i \in A_c} \neg \veebar \{\bar{C}(\boldsymbol{x}_i), L\,(\boldsymbol{x}_i)\}}{|A_c|} \quad \forall c \qquad 4.4$$

Table 4. 1. Assigned labels, true labels, XOR and negation of XOR for samples of arbitrary cluster, A$_2$.

$\bar{C}(\boldsymbol{x}_i)$	$L\,(\boldsymbol{x}_i)$	$\veebar \{\bar{C}(\boldsymbol{x}_i), L\,(\boldsymbol{x}_i)\}$	$\neg\veebar \{\bar{C}(\boldsymbol{x}_i), L\,(\boldsymbol{x}_i)\}$
2	4	1	0
2	2	0	1
2	3	1	0
2	2	0	1
2	2	0	1
2	2	0	1

Thus, four of six samples are predicted correctly and $CA_2 = 4/6 = 0.66$.

Given the A_c clusters, unsupervised classifiers are developed, accordingly. In order to set up the OCCs, each cluster is randomly divided into two disjoint subsets, including training subset, A_c^T, and validation subset, A_c^V, Equation (4. 5):

$$A_c^T \cup A_c^V = A_c \ \ , \ A_c^T \cap A_c^V = \emptyset \ \ , \ being \ \ |A_c^T| = \alpha|A_c| \ and \ \ |A_c^V| = (1-\alpha)|A_c| \tag{4. 5}$$

In which, α must be selected.

The prediction of each OCC for each sample, $h_c(\boldsymbol{x}_i)$ is set to meet the conditions in Equation (4. 6):

$$h_c(\boldsymbol{x}_i) = \begin{cases} 1 & if\ \boldsymbol{x}_i \in A_c \\ 0 & if\ \boldsymbol{x}_i \notin A_c \end{cases} \ \forall\, \boldsymbol{x}_i \ \leftrightarrow \ \frac{\sum_{\boldsymbol{x}_i \epsilon A_c^T} h_c(\boldsymbol{x}_i)}{\left|A_c^T\right|} = 1 \ \ \forall c \tag{4. 6}$$

Validation Prediction Performance (VPP) of each OCC can be calculated with respect to each validation set, Equation (4. 7), as well as an Overall Validation Prediction Performance (OVPP) of each OCC with respect to the whole set of validation data , Equation (4. 8):

$$VPP_{cq} = \frac{\sum_{\boldsymbol{x}_i \epsilon A_q^V} h_c(\boldsymbol{x}_i)}{\left|A_q^V\right|} \leq 1 \ \ \forall c, q \tag{4. 7}$$

$$OVPP_c = \frac{\sum_{\boldsymbol{x}_i \epsilon \cup A_q^V} h_c(\boldsymbol{x}_i)}{\left|\cup A_q^V\right|} \leq 1 \ \ \forall c \tag{4. 8}$$

Given the set of OCCs and provided an acceptable performance for the historical/ initial dataset X, the next step is using them for the classification of a new dataset X^*, Equation (4. 9):

$$X^* = \begin{pmatrix} x_{11}^* & \cdots & x_{1J}^* \\ \vdots & \ddots & \vdots \\ x_{I'1}^* & \cdots & x_{I'J}^* \end{pmatrix} \tag{4. 9}$$

Assume again the availability of the true label for each sample, $L\ (\boldsymbol{x}_i^*)$, the Individual Prediction Performance (IPP) of each OCC could be estimated using Equation (4. 10):

$$IPP_c = \frac{\sum_{\boldsymbol{x}_i^* \in X^*} \neg \veebar \{h_c(\boldsymbol{x}_i^*), L\ (\boldsymbol{x}_i^*)\}}{|X^*|} \ \ \forall c, \qquad |X^*| = I' \tag{4. 10}$$

The assumption of no simultaneous faults allows defining a net prediction given by the whole set of OCCs for each new sample, $H(\boldsymbol{x}_i^*)$, that assigns the new sample $\boldsymbol{x}_i^*$ to the A or B subsets. This is given by Equation (4. 11) and illustrated in Table 4. 2.

$$H(\boldsymbol{x}_i^*) = \begin{cases} \text{c} & \text{if} \ \veebar \{h_{\boldsymbol{c}}(\boldsymbol{x}_i^*), \forall c\} \\ 0 & otherwise \ \neg \veebar \{h_c(\boldsymbol{x}_i^*), \forall c\} \end{cases} \tag{4. 11}$$

Table 4. 2. Assigning $\boldsymbol{x}_i^*$ to the possible subsets.

$h_1(\boldsymbol{x}_i^*)$	$h_2(\boldsymbol{x}_i^*)$	$h_3(\boldsymbol{x}_i^*)$	$\veebar \{h_{\boldsymbol{c}}(\boldsymbol{x}_i^*), \forall c\}$	$\neg \veebar \{h_c(\boldsymbol{x}_i^*), \forall c\}$	$H(\boldsymbol{x}_i^*)$	Cluster
1	0	0	1	0	1	A_1
0	1	0	1	0	2	A_2
0	0	1	1	0	3	A_3
0	0	0	0	1	0	B
1	1	1	0	1	0	B
1	1	0	0	1	0	B
1	0	1	0	1	0	B
0	1	1	0	1	0	B

Finally, the Net Prediction Performance (NPP) for the dataset X^*can be calculated based on Equation (4. 12):

$$NPP = \frac{\sum_{\boldsymbol{x}_i^* \in X^*} \neg \veebar \{H(\boldsymbol{x}_i^*), L\,(\boldsymbol{x}_i^*)\}}{|X^*|}\ , \qquad |X^*| = I' \tag{4. 12}$$

Following this scheme, the assignment of new samples to the unlabeled set B allows a quantitative criterion for updating the models once the size of B exceeds an Updating Threshold (UT), ($|B| \geq UT$).

The implementation of this scheme is done with the proposed framework that is presented in Figure 4. 1 and Figure 4. 4 and has two main modules: The Offline Model Updating (OMU) module, Figure 4. 1, and the Online Monitoring (OM) module, Figure 4. 4.

The initial dataset, X, must be processed by the OMU module in order to determine the “cluster dataset” A, the classification models (OCCs), the UT value, and the unlabeled dataset, B. UT is successively updated by the summation of the Updating Threshold Parameter (UTP) and the number of samples in the current dataset B ($UT \leftarrow UTP + |\mathrm{B}|$).

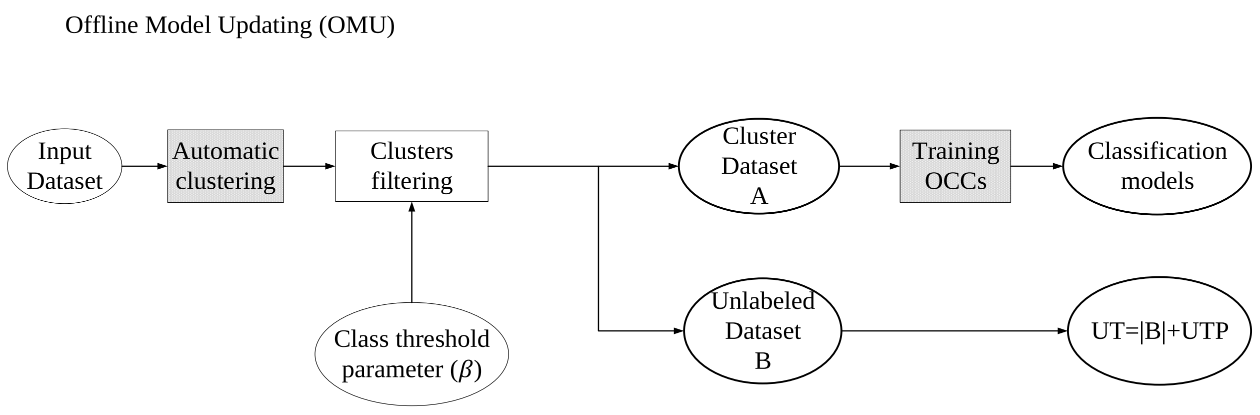

Figure 4. 1. Proposed framework: OMU module.

In Figure 4. 1 there are two black boxes corresponding to the functionality of the automatic clustering and OCCs. The type of implemented tools in these two black boxes could be replaced by other standard tools.

The automatic clustering may deal with two types of changes. In the first type, arriving samples lead to extension or modification of existing clusters, Figure 4. 2 whereas in the second type, arriving samples make new clusters that are completely different from existing clusters, Figure 4. 3.

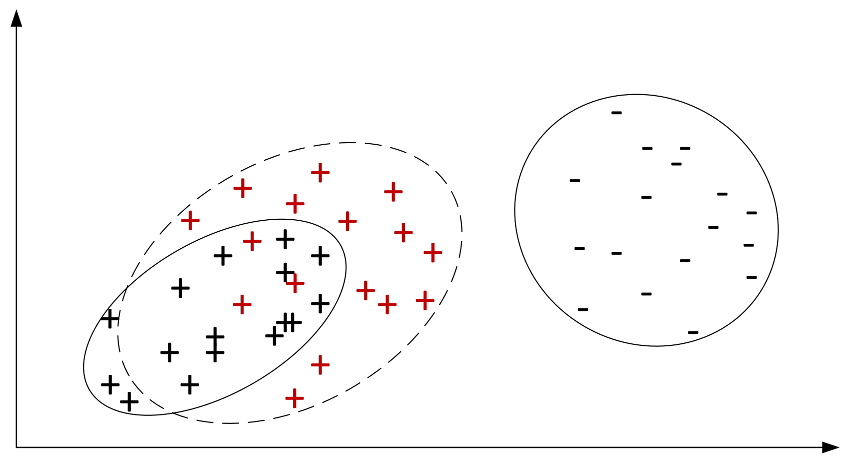

Figure 4. 2. Extension of the existing cluster.

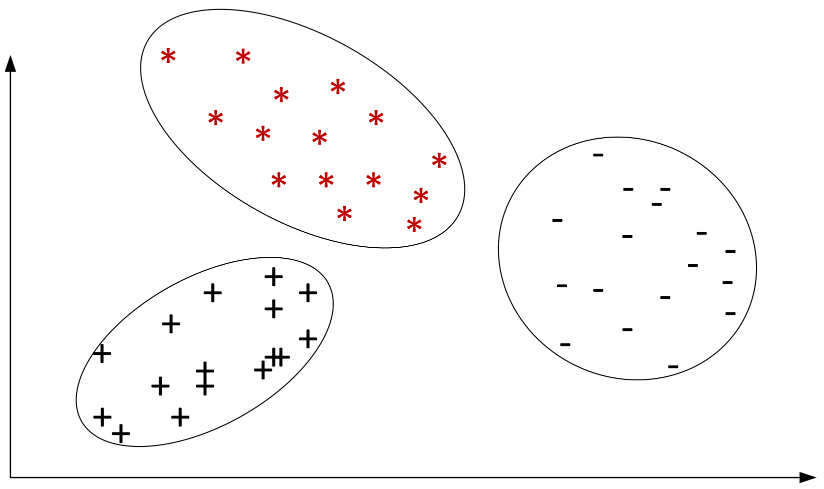

Figure 4. 3. Appearance of a new cluster.

In the OM module, Figure 4. 4, the online samples are initially processed and then are classified by a set of classifiers, Equation (4. 11). The online samples that are diagnosed as elements of A_c subsets are discarded while those not clearly belonging to any of them are kept and assigned to B^*dataset; B^* is the unlabeled dataset in OM for online samples. The reason is based on this fact that while the OCCs could assign the online samples to the predefined clusters those samples do not have any new information for updating the border of the predefined cluster. On the other hand, assigned samples to the B^* possibly could update the borders of the predefined cluster/subsets or make new cluster thus they are saved [46].

After adding each sample to the unlabeled dataset, B^*, in online monitoring module, a condition must be checked: “Do number of samples in the unlabeled dataset, B*, meet the UT?”, Equation (4. 13):

$$if\ |B^*| \geq UT \tag{4. 13}$$

Furthermore, for practical implementation another condition must be checked: “it is not updating?” that implies OMU is on progressing or not. If answers of both conditions are “Yes” then OMU will be triggered by OM. The inlet dataset of OMU, X^*, is made with the union of the cluster dataset, A, and B^* based on Equation (4. 15):

$$A = \begin{pmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & \ddots & \vdots \\ x_{I''1} & \dots & x_{I''J} \end{pmatrix}, B^* = \begin{pmatrix} x^*_{11} & \cdots & x^*_{1J} \\ \vdots & \ddots & \vdots \\ x^*_{I'''1} & \dots & x^*_{I'''J} \end{pmatrix}, X^* = A \cup B^*$$

$$X^* = \begin{pmatrix} x_{1,1} & & x_{1,J} \\ \vdots & \dots & \vdots \\ x_{I'',1} & & x_{I'',J} \\ x^*_{(I''+1),1} & \ddots & x^*_{(I''+1),J} \\ \vdots & & \vdots \\ x^*_{(I''+I'''),1} & \cdots & x^*_{(I''+I'''),J} \end{pmatrix} = \begin{pmatrix} x^*_{11} & \cdots & x^*_{1J} \\ \vdots & \ddots & \vdots \\ x^*_{I'1} & \dots & x^*_{I'J} \end{pmatrix} \qquad 4.\ 14$$

Online Monitoring (OM)

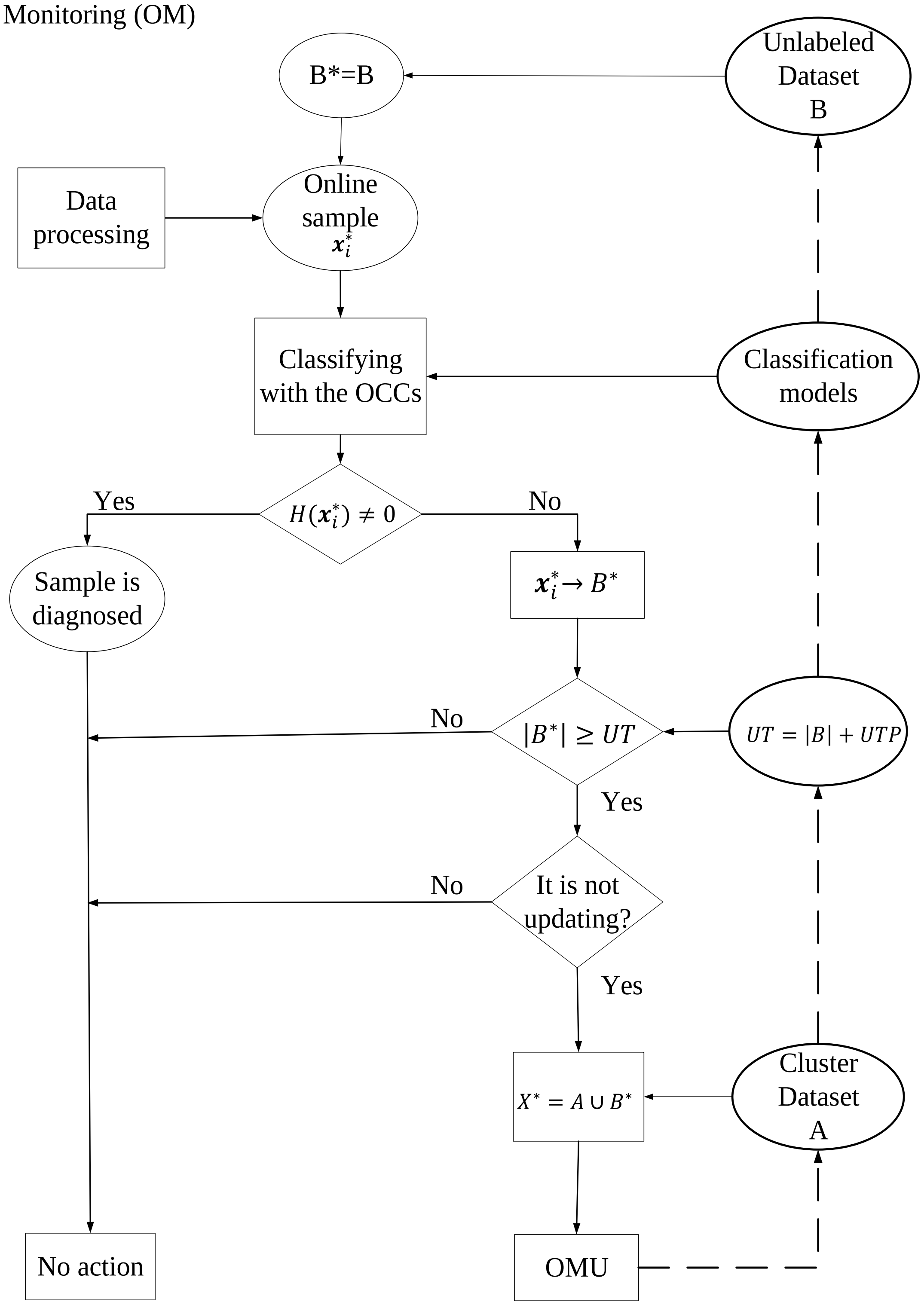

Figure 4. 4. Proposed framework: OM module.

By calling OMU, a new round of automatic clustering with the updated dataset, X^*, must be done. The automatic clustering could be performed with the combination of cluster dataset, A, and unlabeled dataset, B^* as X^* and with one round of automatic clustering, based on Equation (4. 3). Although increasing number of the clusters, could result in increasing the possibility of confusing the clusters. Therefore, an alternative method is suggested and implemented.

An alternative and more reliable way is to merge the unlabeled dataset B^* with only one existing cluster A_c, and with the rest at subsequent updating steps. First updating step produces the clusters A_0^u and B^{*1} by processing the dataset $X^* = A_0 \cup B^{*0}$ in which $B^{*0} = B^*$. Second updating step produces clusters A_1^u and B^{*2} by processing the dataset $X^* = A_1 \cup B^{*1}$ and so on, Equation (4. 15):

$$\begin{array}{lll} Updating\ A_0 & Cl\big((A_0 \cup B^{*0}), \beta\big) \Rightarrow & A_0^u, B^{*1} \\ Updating\ A_1 & Cl\big((A_1 \cup B^{*1}), \beta\big) \Rightarrow & A_1^u, B^{*2} \\ & \vdots & \\ Updating\ A_c & Cl\big((A_c \cup B^{*c}), \beta\big) \Rightarrow & A_c^u, B^{*(c+1)} \end{array} \qquad 4.\ 15$$

Thus, some samples of B^{*c} may be clustered with A_c and vice versa. Those samples of A_c that are clustered into $B^{*c}$are removed because they may confuse the clustering function Cl in the next updating rounds. Of course, the outcomes of this scheme are sequence dependent, but tests have been run proving that different updating sequences produce the same outcomes within ±1% changes in CA and CPU time.

4.1.3 Data Processing

The MDK predictor of the three tanks includes the construction and the training of three dynamic models. Each MDK predictor approximates the future value of each tank level as a function of the previous values of the system inlets $Q_1(t)$, $Q_2(t)$ and the levels h_1(t), h_2(t), h_3(t). As a first modeling trail and in the same time in order to keep the dynamic structure of the models as simple as possible, the process is assumed to have no significant delay, Equation (4. 16):

$$\left.\begin{array}{l}\hat{h}_1(t+1) = f_1\big[h_1(t), h_2(t), h_3(t), Q_1(t), Q_2(t)\,\big]\\ \hat{h}_2(t+1) = f_2[h_1(t), h_2(t), h_3(t), Q_1(t), Q_2(t)]\\ \hat{h}_3(t+1) = f_3[h_1(t), h_2(t), h_3(t), Q_1(t), Q_2(t)]\end{array}\right\} \quad 4.\ 16$$

A fault free random signal of the process input variables has been used for training the MDKs predictor that will be used to estimate the process outputs (tank levels). The observer is validated and its performance assessed; Figure 4. 5-left shows the validation inlet scenarios while Figure 4. 5-right shows the predicted tank levels (dotted red lines) compared to the exact outputs (solid black lines) and the process measured outputs (solid blue lines). The results illustrate the very high prediction accuracy of the observer, and its efficient ability to identify the real underlying behavior of the outputs. The observer achieves a very small normalized root mean square error of 1.05%, 1.1%, 1.02 % for each model respectively. The results also emphasize the high capabilities of the MDK observer to predict a multivariate behavior over a relatively large time horizon.

By applying the observer for feature extension, instead of processing dataset X, Equation (4. 17), all the tasks that described in previous sections will be done by error set, E, Equation (4. 18):

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1J} \\ \vdots & \ddots & \vdots \\ x_{I1} & \cdots & x_{IJ} \end{pmatrix}, \qquad X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_J), \boldsymbol{x}_j = (x_{1j}, x_{2j} \dots \dots .. x_{IJ}) \quad 4.\ 17$$

$$E = \begin{pmatrix} e_{11} & \cdots & e_{1J'} \\ \vdots & \ddots & \vdots \\ e_{I1} & \cdots & e_{IJ'} \end{pmatrix}, \qquad E = (\boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_{J'}), \boldsymbol{e}_j = (e_{1j}, e_{2j} \dots \dots .. e_{Ij}) \quad 4.\ 18$$

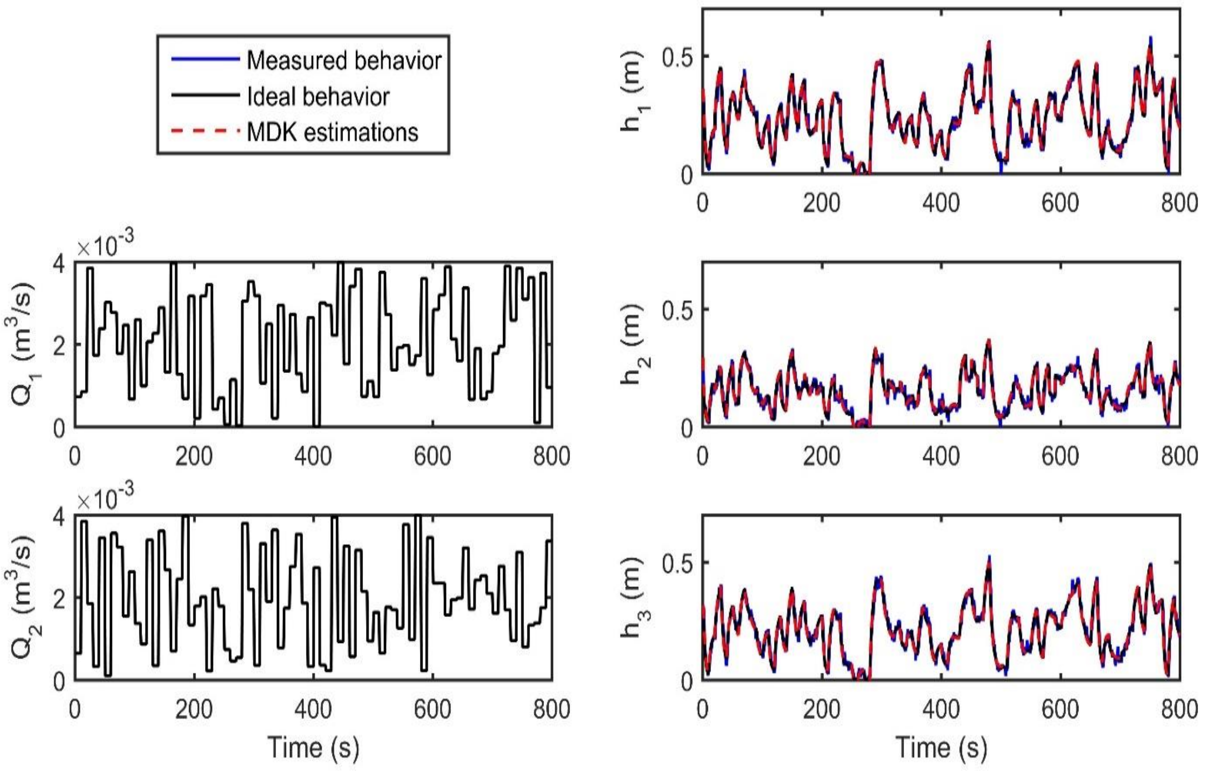

Figure 4. 5. Input-output signal for the MDKs observer validation.

4.1.4 Results

The proposed FDD framework is validated and assessed with the fault pattern given in Figure 4. 6. The initial dataset, X, is assumed to have two clusters: Nr and F1. During the time horizon (1000 s), two new faults, F2 and F3, appear while the fault pattern switches between different conditions/clusters. The complete time horizon is studied in four equal time intervals (250 s). The results from this fault pattern are next organized in figures showing the net prediction ($H(\boldsymbol{x}_i^*)$), comparative tables presenting prediction performance (IPP_c and NPP), clustering accuracy (CA_c), and a table of classification validation (VPP_{cq} and $OVPP_c$).

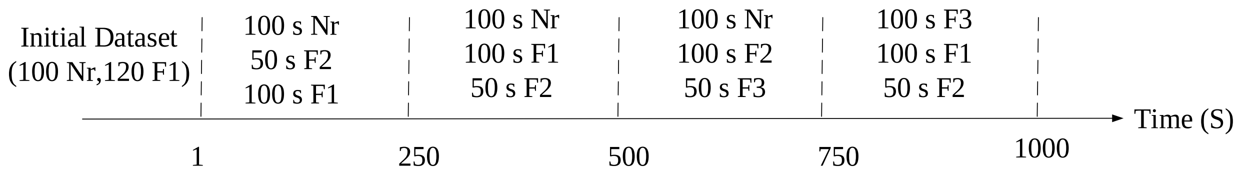

Figure 4. 6. Fault pattern.

The OMU module is executed for the first time on the initial dataset to produce: The “cluster dataset”, A; the classification models; the updating threshold, UT; and the unlabeled dataset, B. Next, during the first 250 s of the time horizon, 50 samples of F2 appear in the fault scheme. Then they increase to 100 samples during the second period (250 s to 500 s). Since F2 samples do not exist in the initial dataset, the framework must subsequently detect them as novel samples.

As UTP is regulated to be 80 samples, and initially there is no sample in B, it is expected at 480 s for the first time OMU is called. Regarding that UTP is tuned experimentally it could have different values from its adjusted amount, 80 samples. If the UTP value is too small, the number of calling the OMU will increase as well as the computational effort. On the other hand, if it is too large, the framework cannot timely detect the new clusters.

The cluster threshold parameter, β, is set on 30 samples. If number of samples in each newly created cluster is greater than β, a new OCC will be trained based on it. Similar to UTP, there are two extreme limits for β. If it is too small, the trained OCCs will be weak, and if it is too large, the OCCs of the new conditions will be developed too late for timely diagnosis.

4.1.5 Running the Framework

The fault pattern, Figure 4. 6, is assessed and discussed with two different inlet flowrate scenarios, Figure 4. 7. In scenario1, Figure 4. 7-left, both inlet flowrates are steady and fixed at $Q_1(t) = 3 \times 10 - 3\ m^3/s\ and\ Q_2(t) = 2.5 \times 10 - 3\ m^3/s$. In scenario 2, Figure 4. 7-right, the two inlet flowrates oscillating between 1 to 3×10^{-3} m^3/s ($Q_1 = \ sin\ (3 \cdot t), Q_2 = \ cos\ (1.5 \cdot t)$). Scenario 1 could be regarded as simpler one while scenario 2 is characterized by an extreme dynamic profile.

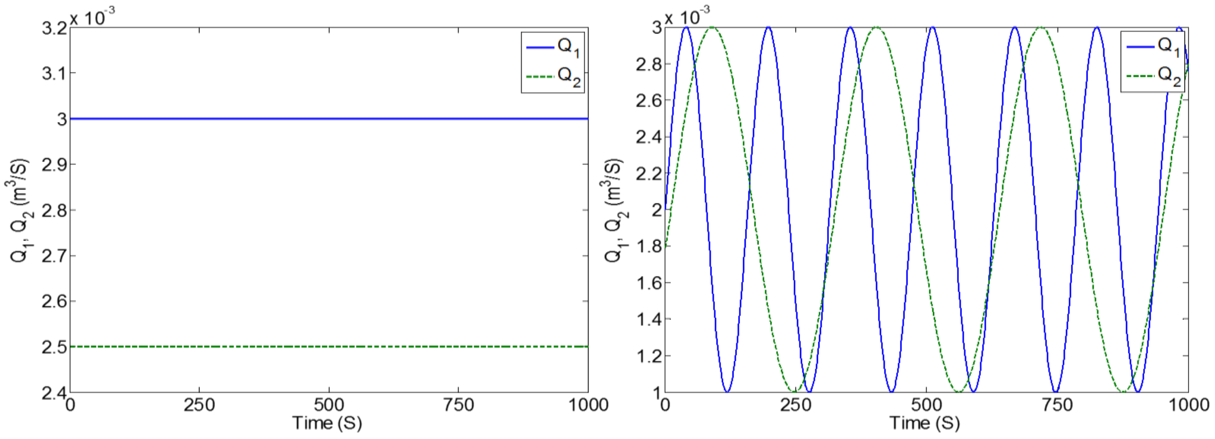

Figure 4. 7. Scenario 1 (left) and Scenario 2 (right) of the inlet flowrates.

Figure 4. 8 and Figure 4. 9 illustrate the results for these two scenarios with the same fault pattern. Figure 4. 8 presents the net prediction, $H(\boldsymbol{x}_i^*)$, of the OCSs for scenario 1. OMU is called twice and each update takes 16 s, approximately. In the first 250 s, NPP is 95.2%. After detecting 80 new samples, the first call of OMU starts at time 475 s, and OCS_F2 becomes available since 491 s. From 501 s to 750 s, the NPP of the three classifiers (OCS_Nr, OCS_F1 and OCS_F2) is 89.6%, resulting from 26 wrong predictions out of 250. The second OMU call starts from 760 s, and then from 776 s the OCS_F3 starts monitoring the process. Figure 4. 8 shows how before 776 s F3 samples are labeled as $B^*$and after that, they are diagnosed as F3. For the last interval (750 s to 1000 s) NPP involves all four OCSs and its value is 94%. The overall NPP value for the whole time horizon (1 s to 1000 s) is 93.1%. In Figure 4. 8, the thin striped rectangle in front of the F2 from 1 s to 491 s indicates during that time interval F2 samples are not diagnosed as a defined class. For F3 samples the thin striped rectangle is drawn till 776 s and after that they are diagnosed as the defined class. The thin striped rectangles in Figure 4. 9 for samples of F2 and F3 have the same meaning.

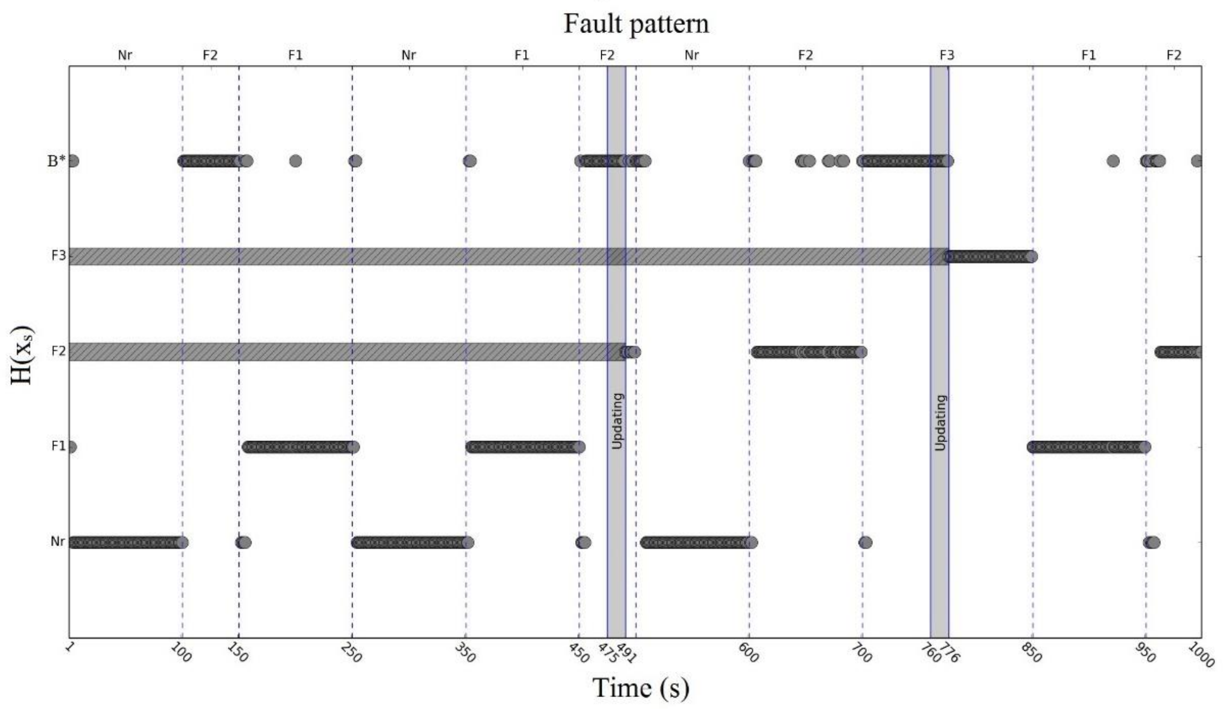

Figure 4. 8. Framework performance with scenario 1.

Figure 4. 9 presents net predictions, $H(\boldsymbol{x}_i^*)$ of the OCSs for scenario 2. The same fault pattern causes two OMU calls, again each one takes 16 s, approximately. The first updating starts at 470 s and the OCS_F2 is added at 486 s. The second update starts at 819 s and then at the 835 s the OCS_F3 is incorporated into the FDD system. Among wrong diagnosed samples no one is confused with F2 while there is only one sample that is confused with F3.

.

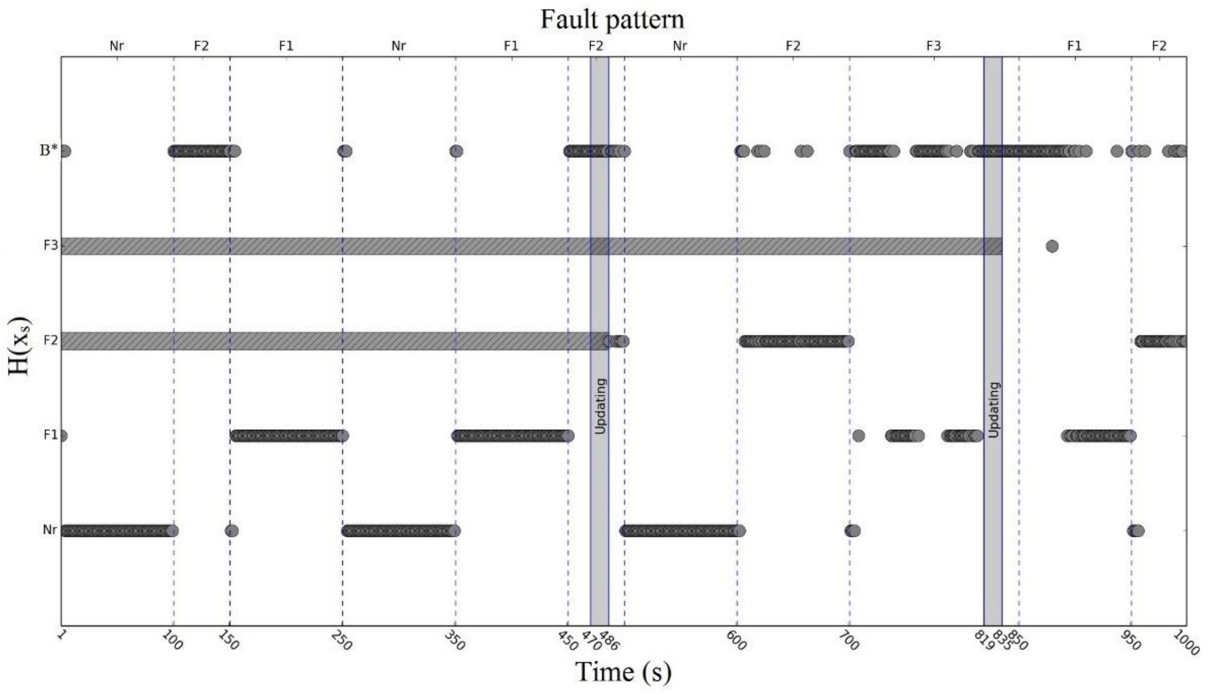

Figure 4. 9. Framework performance with scenario 2.

Table 4. 3 presents the performance assessment, NPP and IPP_cvalues, for all OCSs in all the time intervals. For scenario1 and scenario 2, in the first time interval, there is quite high NPP with two OCSs and there are only 12 and 11 wrong predictions, respectively. This confirms that the OMU on the initial dataset is executed successfully. In the second time interval, 251 s to 500 s, in scenario 1 there are 16 wrong predictions while in scenario 2 there are 15 wrong predictions. In this time interval, OCS_F2 is added; in scenario 1 it predicts nine samples that two of them are diagnosed wrongly, and in scenario 2 it has seven correct predictions and seven incorrect predictions.

In the third time interval, 501 s to 750 s, NPP based on three OCSs for scenario1 is 89.6%, and for scenario 2 is 88%. In this time interval for both scenarios, newly updated OCSs, including OCS_Nr and OCS_F1, and added OCS, OCS_F2, have quite high IPPs. This proves that the first call of OMU is performed effectively. Then, in the last time interval for both scenarios, by the second call of the OMU the new OCS, OCS_F3, is added, and it has quite high IPP. In this time interval in the scenario 1, IPP of the OCSs and NPP prove the second call of the OMU is done suitably, too. But in the scenario 2, IPP of the OCS_F1 is 67.6% and consequently NPP is relatively low, 55.6%. The OCS_F1 confuses F1 with F3; this could arise from either OMU tasks such as clustering and training of the OCS or defects in the OCS algorithm. In the next section details of the clustering and classification of the OMU is discussed.

Table 4. 3. IPP and NPP of the classifiers for two scenarios.

	Scenario 1			Scenario 2		
1 s ≤ t ≤ 250 s	Correct	Incorrect	IPP/NPP (%)	Correct	Incorrect	IPP/NPP (%)
OCS_Nr	241	9	96.4	240	10	96.0
OCS_F1	239	11	95.6	239	11	95.6
H (Net prediction)	238	12	95.2	239	11	95.6
251 s ≤ t ≤ 500 s	Correct	Incorrect	IPP/NPP (%)	Correct	Incorrect	IPP/NPP (%)
OCS_Nr	239	11	95.6	246	4	98.4
OCS_F1	242	8	96.8	244	6	97.6
OCS_F2	7	2	77.7	7	7	50.0
H (Net prediction)	234	16	93.6	235	15	94.0
501 s ≤ t ≤ 750 s	Correct	Incorrect	IPP/NPP (%)	Correct	Incorrect	IPP/NPP (%)
OCS_Nr	237	13	94.8	242	8	96.4
OCS_F1	250	0	100.0	237	13	94.8
OCS_F2	235	15	94.0	238	12	95.2
H (Net prediction)	224	26	89.6	220	30	88.0
750 s ≤ t ≤ 1000 s	Correct	Incorrect	IPP/NPP (%)	Correct	Incorrect	IPP/NPP (%)
OCS_Nr	246	4	98.4	245	5	98.0
OCS_F1	249	1	99.6	169	81	67.6
OCS_F2	236	14	94.4	235	15	94.0
OCS_F3	224	0	100.0	148	15	90.7
H (Net prediction)	235	15	94.0	139	111	55.6
1 s ≤ t ≤ 1000 s	Correct	Incorrect	IPP/NPP (%)	Correct	Incorrect	IPP/NPP (%)
OCS_Nr	964	36	96.4	973	27	97.3
OCS_F1	980	20	98.0	889	111	88.9
OCS_F2	478	31	93.9	480	34	93.3
OCS_F3	224	0	100.0	148	15	90.7
H (Net prediction)	931	69	93.1	835	165	83.5

4.1.6 Evaluation of the Automatic Model Updating: Clustering and Validation of the Classifiers

In this section, details of the OMU performance on the initial dataset, and online datasets is discussed. CPU time of clustering refers to the required time for partitioning the input dataset into clusters while CPU time of classification refers to the required time of training. After diagnosing the clusters by automatic clustering, α in Equation (4. 5) is selected to be 0.8.

Table 4. 4 presents results of the clustering tasks for both scenarios based on CA, CPU time and percentage of the removed samples. For both scenarios, all the clustering tasks are reported with the high CAs. Clustering tasks of the first and second OMU calls are done by applying Equation (4. 15); reasonable CAs prove the consistency of it. For both scenarios, in the second OMU call, although new cluster is added, CPU time of the clustering comparing with the first OMU call does not meaningfully change.

After clustering task, the provided subsets are divided into training and validation sets. Table 4. 5 presents the VPP_{cq}, and $OVPP_c$ for the initial dataset and two OMU calls. In the first OMU call, performance of the OCS_F2 for detecting F1 samples, as negative samples, is $VPP_{21} = 52.6\%$ while in scenario 2 is 63.1%. In the second OMU call for both scenarios VPP_{21} improve to 100% that implies effectiveness of second OMU call for updating OCS_F2.

In scenario 2 and for both OMU calls, F1 samples are well clustered, Table 4. 4; this proves that low performance of the OCS_F1 in the last time interval of the scenario 2 (reported in Table 4. 3) is not because of the clustering task. On the other hand, OVPPs of the OCS_F1 in scenario 2 for both OMU calls are 98.1% and 94.2%, respectively. These imply that the poor performance of the OCS_F1 in the last time interval of the scenario 2 is not because of the training task of the classifier. Thus, the relatively poor performance of the OCS_F1 in the last time interval of the scenario 2 is because of the OCS algorithm. Advanced kernel functions for the OCS or replacing OCS with better OCC could be an effective way, but studying them is beyond scope of this section.

Table 4. 4. Clustering performance for scenario 1 and scenario 2.

	Scenario 1				Scenario 2			
	Initial Dataset							
Cluster	Consisting of (%)				Consisting of (%)			
	Nr	F1	F2	F3	Nr	F1	F2	F3
Nr	98	2	-	-	97	3	-	-
F1	0	100	-	-	0	100	-	-
F2	-	-	-	-	-	-		-
F3	-	-	-	-	-	-	-	-
CA (%)	99				98.5			
CPU time (s)	3.2				3.2			
Removed samples (%)	0				0			
	First OMU							
Cluster	Consisting of (%)				Consisting of (%)			
	Nr	F1	F2	F3	Nr	F1	F2	F3
Nr	92.9	3.9	3.2	-	92.1	0	7.9	-
F1	0	100	0	-	0	100	0	-
F2	0	3	97	-	0	3.1	96.9	-
F3	-	-	-	-	-	-	-	-
CA (%)	95.0				95.3			
CPU time (s)	15.7				15.8			
Removed samples (%)	1				0.7			
	Second OMU							
Cluster	Consisting of (%)				Consisting of (%)			
	Nr	F1	F2	F3	Nr	F1	F2	F3
Nr	94.2	3.3	2.5	0	89.3	3.5	6.1	1.1
F1	0	97	1	2	0	98	0	2
F2	0	1.9	98.1	-	0	1.4	98.6	0
F3	0	0	1.2	98.8	0	1.5	0	98.5
CA (%)	95.2				94.1			
CPU time (s)	15.7				15.5			
Removed samples (%)	1.4				1.6			

Table 4. 5. Validation of the classifiers.

	Initial Dataset									
	Scenario 1					Scenario 2				
Classifier	VPP_{cq}(%)				$OVPP_c$ (%)	VPP_{cq} (%)				$OVPP_c$ (%)
	Nr (q =0)	F1 (q =1)	F2 (q =2)	F3 (q=4)		Nr (q =0)	F1 (q =1)	F2 (q =2)	F3 (q =3)	
OCS_Nr (c=0)	100	100	-	-	100	95	100	-	-	97.4
OCS_F1 (c=1)	100	94.7	-	-	97.4	100	94.7	-	-	97.4
OCS_F2 (c=2)	-	-	-	-	-	-	-	-	-	-
OCS_F3 (c=3)	-	-	-	-	-	-	-	-	-	-
CPU time (s)	0.1					0.1				
	First OMU									
	VPP_{cq} (%)				$OVPP_c$ (%)	VPP_{cq} (%)				$OVPP_c$ (%)
	Nr (q =0)	F1 (q =1)	F2 (q =2)	F3 (q=4)		Nr (q =0)	F1 (q =1)	F2 (q =2)	F3 (q=4)	
OCS_Nr (c=0)	96.1	100	100	-	98.1	90.9	100	100	-	96.2
OCS_F1 (c=1)	100	89.4	100	-	96.3	100	94.7	100	-	98.1
OCS_F2 (c=2)	100	52.6	80	-	80	100	63.1	91.6	-	84.9
OCS_F3 (c=3)	-	-	-	-	-	-	-	-	-	-
CPU time (s)	0.1					0.1				
	Second OMU									
	VPP_{cq} (%)				$OVPP_c$ (%)	VPP_{cq} (%)				$OVPP_c$ (%)
	Nr (q =0)	F1 (q =1)	F2 (q =2)	F3 (q=4)		Nr (q =0)	F1 (q =1)	F2 (q =2)	F3 (q=4)	
OCS_Nr (c=0)	100	95	100	100	98.6	87.5	100	100	100	95.6
OCS_F1 (c=1)	96.1	95	100	87.5	94.4	100	85	100	92.3	94.2
OCS_F2 (c=2)	96.1	100	100	100	98.6	95.8	100	100	100	98.5
OCS_F3 (c=3)	100	100	100	100	100	100	90	100	69.2	91.3
CPU time (s)	0.1					0.1				

4.1.8 Conclusions

Updating ability is an essential necessity to any FDD system, especially if it is intended to perform in an unsupervised manner. Although the ND methods are reported in the literature, the strategy for updating the FDD with the detected novel samples is hardly addressed. Thus, this section presents a hybrid framework for unsupervised automatic updating of data driven FDD.

Modules have developed for assigning samples to either existing clusters or new clusters. The modified clusters, by adding or removing samples, are applied for retraining the OCCs aimed at updating them. Therefore, the successive updating procedure contributes to an enhanced learning strategy. With the proposed strategy, instead of inefficiently managing all historical records, the significant number of samples are always kept to model the different process patterns. The framework is validated by benchmark using two inlet flowrate scenarios and a fault scheme in which two new clusters appear in the process. Moreover, the proposed framework is evaluated by three main performance indices, including NPP, CA and OVPP. For both scenarios, the reported results imply that the framework could perform appointed tasks of unsupervised automatic FDD updating.

4.2 Unsupervised FD Updating: Handling Concept Drift

In chapter 3, ILDW algorithm for dealing with CD in the supervised manner have been proposed, in this section ILDW algorithm in an unsupervised style is presented and compared with the supervised IL.

.

4.2.1 Introduction

Unsupervised CD detection techniques are commonly unreliable because they produce a large number of false alarms [13]; one common cause of this could be mislabeling. Recently, some new unsupervised algorithm for CD detection is proposed that have promising results [13] [172]. In [13], the number of samples in a classifier's uncertainty region is applied as a metric for detecting drift and the results show high detection rate, high prediction performance, and low false alarm rate.

In supervised FD approaches, the classifiers are trained with the labeled samples, and the assumption is that they are correct and reliable. For real practice that is usually unsupervised or semi supervised, assigning labels to samples is a challenging task that needs efforts and examinations. Accordingly, mislabeling may occur because of the several possible reasons such as expert errors, lack of information etc. [173] [174]. Mislabeling the training samples is an unavoidable problem that destroys learning procedure. Addressing this, in [175] a semi supervised algorithm is proposed that has encouraging results. Many other methods for preventing, detecting, and cleaning mislabeled samples could be found in the literature [176].

In process monitoring, Mahalanobis and Euclidean distances have lots of applications [82] [177] [7]. Mahalanobis distance can be considered as a general case of the Euclidean distance, and samples with the same Mahalanobis distance have the same probability [178]. In [173], Mahalanobis distance is applied for handling uncertain and missing labels, and in [179] it is applied for comparison of CD detection methods. In [180], based on the Euclidean distance a classifier is applied, and in [17] it is exploited for ND. In [49], for measuring the similarity between samples the Euclidean distance is applied.

In this section, a framework is proposed in order to update the FD model in an unsupervised style. The goal of the proposed framework is to provide a strategy for unsupervised handling CD for FD updating rather than offer new tools for updating. Therefore, in the framework, arriving datasets are initially classified by MC, and then samples of each class are filtered based on Euclidean distance in order to discard possible wrong labeled samples by MC. The reason for filtering is that the provided labeled samples by MC are used for training the AC. In the framework, adjusting or retraining the MC, and training the AC are similar to ILDW algorithm that are explained in section 3.1. The framework is validated by CSTR with two different hidden context scenarios and the results are compared with supervised IL.

.

4.2.2 Methodology

Assume samples arrive in dataset, X_k, Equation (4. 19):

$$X_k = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{kI1} & \cdots & x_{kIJ} \end{pmatrix}, \mathrm{k} = 1,2, \ldots \ldots \mathrm{K}, \qquad \mathrm{i} = 1,2, \ldots \ldots \mathrm{I}, \qquad \mathrm{j} = 1,2, \ldots \ldots \mathrm{J} \qquad 4.\ 19$$

$$\boldsymbol{x}_{ki} = (x_{ki1}, x_{ki2} \ldots \ldots x_{kiJ})$$

k is counter of TI. The first main step in the proposed unsupervised algorithm is diagnosing X_k to the classes by MC, Equation (4. 20):

$$MC(X_k) = \{\Gamma_{k0}, \Gamma_{k1}, \cdots \Gamma_{kc} \ldots \ldots \Gamma_{kC}\}, \quad c = 0,1, \ldots . C \qquad 4.\ 20$$

In which, Equation (4. 21):

$$\Gamma_{kc} = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{ka1} & \cdots & x_{kaJ} \end{pmatrix}, \Gamma_{kc} = \{\boldsymbol{x}_{ki} | MC(\boldsymbol{x}_{ki}) = c\} \qquad 4.\ 21$$

Because of the unavoidable MC mislabelling, a filtering step is required; so, one parameter, Removing Percentage (RP), is defined. RP is applied for filtering all the Γ_{kc}, and it indicates the percentage of the samples in each class that must be removed in order to maximally reduce the number of wrong diagnosed samples; the RP must be tuned, experimentally. Removed samples are selected among those that are farther, in regard to Euclidean distance, from the center of each class. Thus, error spreading reduces in the next steps of training. After filtering, each Γ_{kc} are modified, Equation (4. 22):

$$\Gamma'_{kc} = \begin{pmatrix} x_{k11} & \cdots & x_{k1J} \\ \vdots & \ddots & \vdots \\ x_{ka'1} & \cdots & x_{ka'J} \end{pmatrix}, a > a' \tag{4. 22}$$

Then the X'_k will be, Equation (4. 23):

$$\bigcup_{\forall c} \Gamma'_{kc} = \{\Gamma'_{k0}, \Gamma'_{k1}, \ldots\ldots \Gamma'_{kC}\} = X'_k \tag{4. 23}$$

In the unsupervised proposed algorithm, the aim is to select efficiently a window of dataset, WI_k, at each time interval, k, as shown in Equation (4. 24):

$$\begin{gathered} WI_k = [X'_{k-r}, \ldots. X'_{k-1}, X'_k] \ \forall k, r \in \mathbb{N} \ \delta \ k - r \geq 1 \\ r = card\ (WI_k) - 1 \end{gathered} \tag{4. 24}$$

In which, ℕ is the sign of natural numbers. In the unsupervised algorithm a Threshold (Tr) is defined. Tr is the minimum accuracy of the AC on X'_{k-r} that must be met, and it is tuned experimentally. If the Tr is satisfied, no extra action is required. If obtained accuracy does not satisfy the Tr, then the X_{k-r} is discarded and the next sequenced dataset, X'_{k-r+1}, in WI_kwill be tested and so on. After meeting the Tr, the MC may be retrained or adjusted. Figure 4. 10 presents the proposed unsupervised FD updating algorithm.

In addition, unsupervised ILDW and supervised IL are compared by MGM score, Equation (4. *25*)

$$\begin{gathered} MGM_u = \omega \times \frac{\overline{CAS_u}}{\overline{CAS_{ref}}} - \frac{\overline{CPU_u}}{\overline{CPU_{ref}}} \\ u \in \{FD\ updating\ methods: Unsupervised\ ILDW, supervised\ IL\}, \\ ref = Reference\ method \end{gathered} \tag{4. 25}$$

In which, CAS, ω and reference method are considered F1 score, two and supervised IL, respectively.

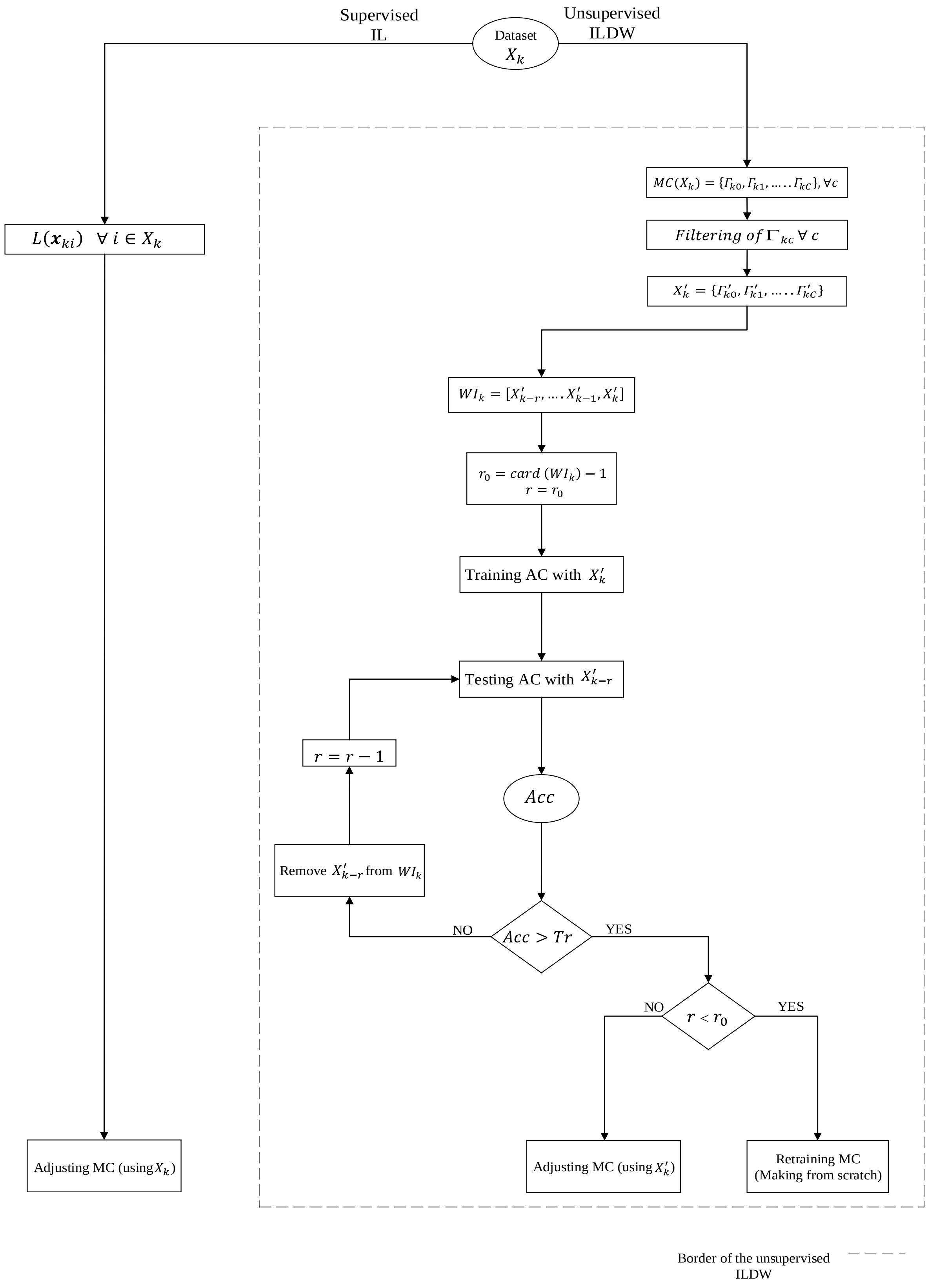
Supervised IL
Dataset X_k
Unsupervised ILDW
$L(\boldsymbol{x}_{ki})\ \ \forall\, i \in X_k$
$MC(X_k) = \{\Gamma_{k0}, \Gamma_{k1}, \ldots \Gamma_{kC}\}, \forall c$
$Filtering\ of\ \boldsymbol{\Gamma}_{kc}\ \forall\ c$
$X'_k = \{\Gamma'_{k0}, \Gamma'_{k1}, \ldots \Gamma'_{kC}\}$
$WI_k = [X'_{k-r}, \ldots X'_{k-1}, X'_k]$
$r_0 = card\ (WI_k) - 1$
$r = r_0$
Training AC with X'_k
Testing AC with X'_{k-r}
$r = r - 1$
Acc
Remove X'_{k-r} from WI_k
NO
$Acc > Tr$
YES
NO
$r < r_0$
YES
Adjusting MC (using X_k)
Adjusting MC (using X'_k)
Retraining MC (Making from scratch)
Border of the unsupervised ILDW

Figure 4. 10. Proposed framework of unsupervised ILDW algorithm for FD updating.

4.2.3 Case Study, Data Preparation and Hidden Context Scenarios

The applied case study is CSTR, and for each scenario 50 datasets are prepared. Each dataset contains 1000 samples of normal and faulty. The normal inlet concentration of the reactant C_A equals to $5.1 + v_0\ mol/L$, where v_0 is a Gaussian noise; $v_0 \approx N\ (\mu = 0, \sigma = 0.045\, mol/L)$. A fault is expected to affect the process that is defined as a step change in the inlet concentration of the reactant, C_A, from its normal value to $5.13 + v_0\, mol/L$.

Two hidden context scenarios are considered, Figure 4. 11 and Figure 4. 12. In the first scenario, Figure 4. 11, heat transfer coefficient from the first to 25th dataset drops to 95% of its initial value, $H_0(4032\ kJ/h.m^2.K)$, then till 50th dataset it drops to 85% of H_0. In the second scenario, Figure 4. 12, heat transfer coefficient during 26 datasets drops to 90% of H_0 while in the 27th dataset it increases to 99% of H_0, then it decreases to 90% of H_0 until the last dataset.

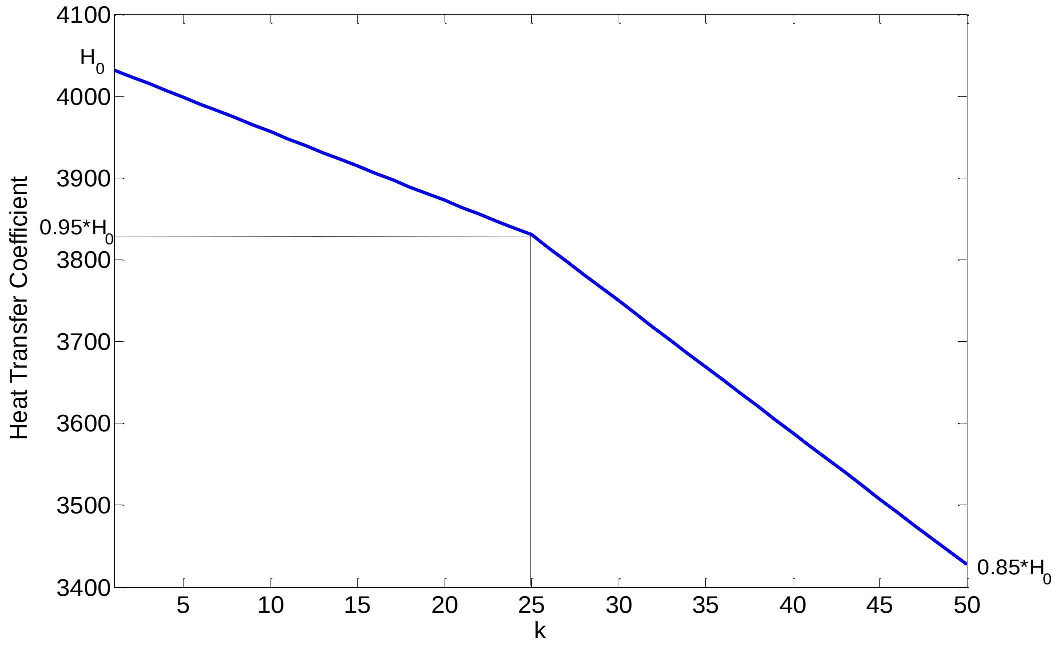

Figure 4. 11. First scenario of hidden context.

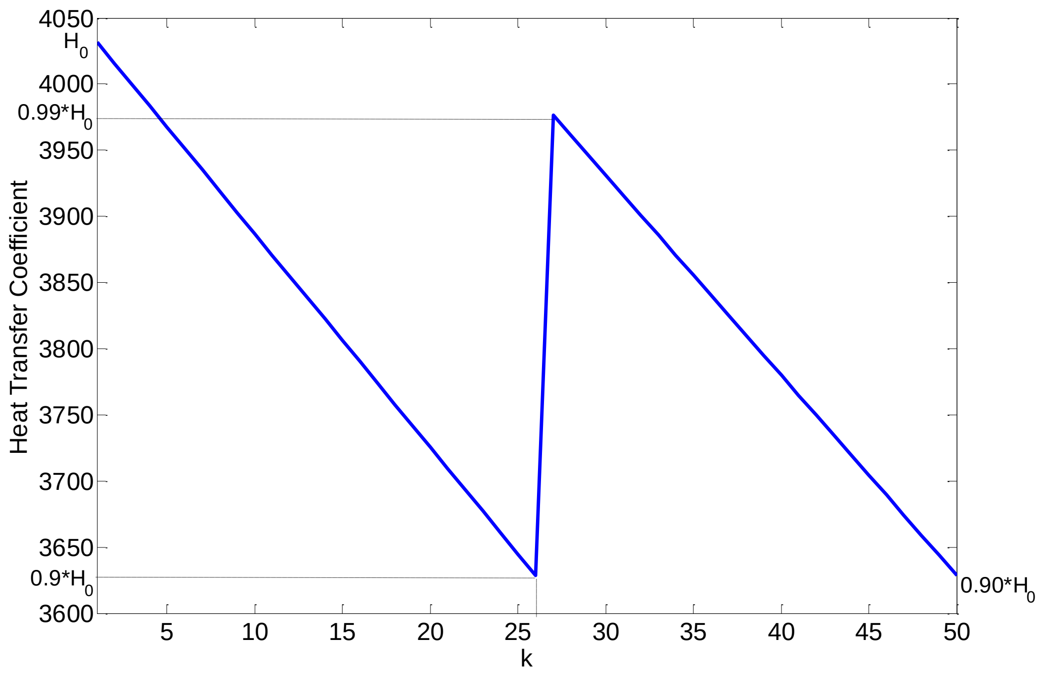

Figure 4. 12. Second scenario of hidden context.

4.2.4 Results

In the ILDW algorithm, GNB is employed as AC while the MC is selected to be SVM, and Tr and RP are set to be 60% and 2%, respectively. For initial training of the MC, the first dataset, X_1, is clustered with the CS automatic clustering method; the results for both scenarios are reported in Table 4. 6.

Table 4. 7 compares unsupervised ILDW and supervised IL for scenario1 and scenario 2. In scenario 1, $\overline{F1}$ of the ILDW is 0.34% less than IL, but $\overline{CPU}$ is improved up to 69.8% that result in 69% enhancement in regard to MGM score. Moreover, $\overline{NSV}$ in ILDW is improved up to 50.2%. In the scenario 2, $\overline{F1}$ of ILDW is 0.7 %less than IL, and $\overline{CPU}$ is improved up to 75% that imply 73% enhancement in MGM score. In addition, in ILDW algorithm $\overline{NSV}$ 59.53% is reduced.

Figure 4. 13, Figure 4. 14 and Figure 4. 15 compare performance of the IL and ILDW, for both scenarios, based on F1 score, CPU time, and NSV, respectively. Figure 4. 16 presents WI_k and its cardinality for ILDW algorithm in two applied scenarios.

Table 4. 6. Performance of the CS as unsupervised automatic clustering method.

	Scenario 1		Scenario 2	
Initial Dataset				
Cluster	Consisting of (%)		Consisting of (%)	
	Nr	F1	Nr	F1
Nr	98.3	1.7	97.3	2.7
F1	3.8	96.2	1.8	98.2
CA (%)	97.5		97.7	
CPU time (s)	2.8		2.7	

Table 4. 7. Performance comparison of the supervised IL algorithm and unsupervised ILDW algorithm.

Scenario	Method	$\overline{\mathbf{F1}}$ (%)	$\overline{\boldsymbol{CPU}}$ (s)	MGM score	$\overline{\mathbf{NSV}}$
1	Supervised IL	99.32	0.053	1	418.38
	Unsupervised ILDW	98.98	0.016	1.69	208.12
2	Supervised IL	98.30	0.036	1	532.68
	Unsupervised ILDW	97.56	0.009	1.73	215.56

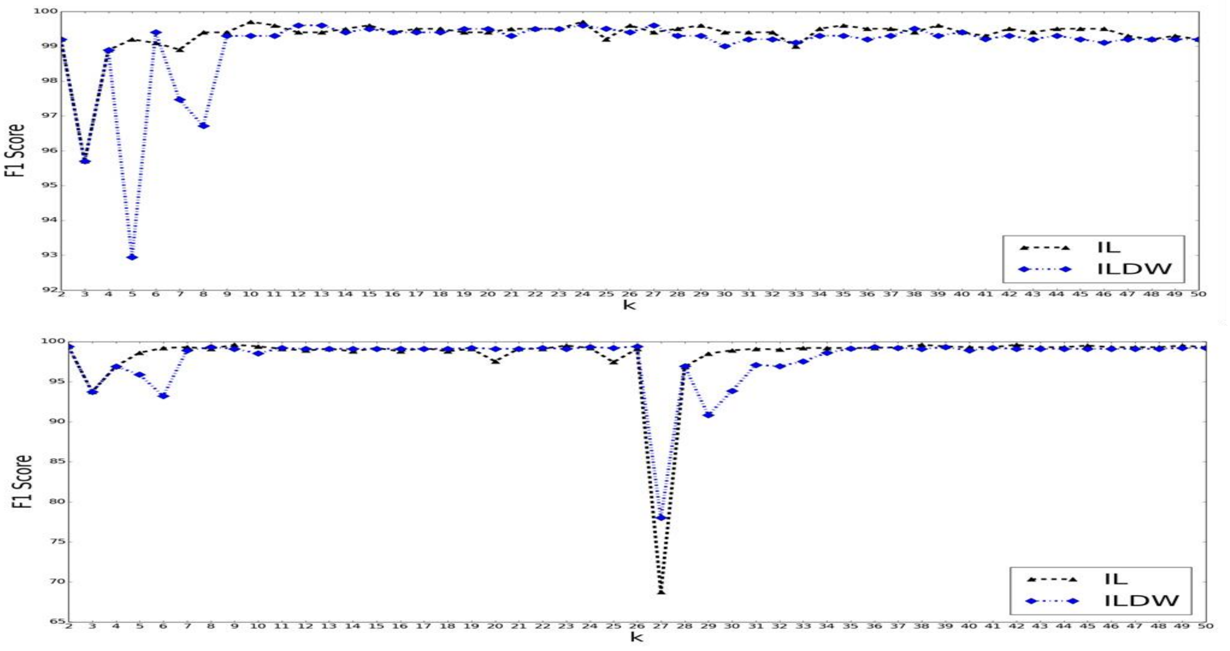

Figure 4. 13. Comparison of F1 score between supervised IL and unsupervised ILDW in scenario 1 (up) and scenario 2 (down).

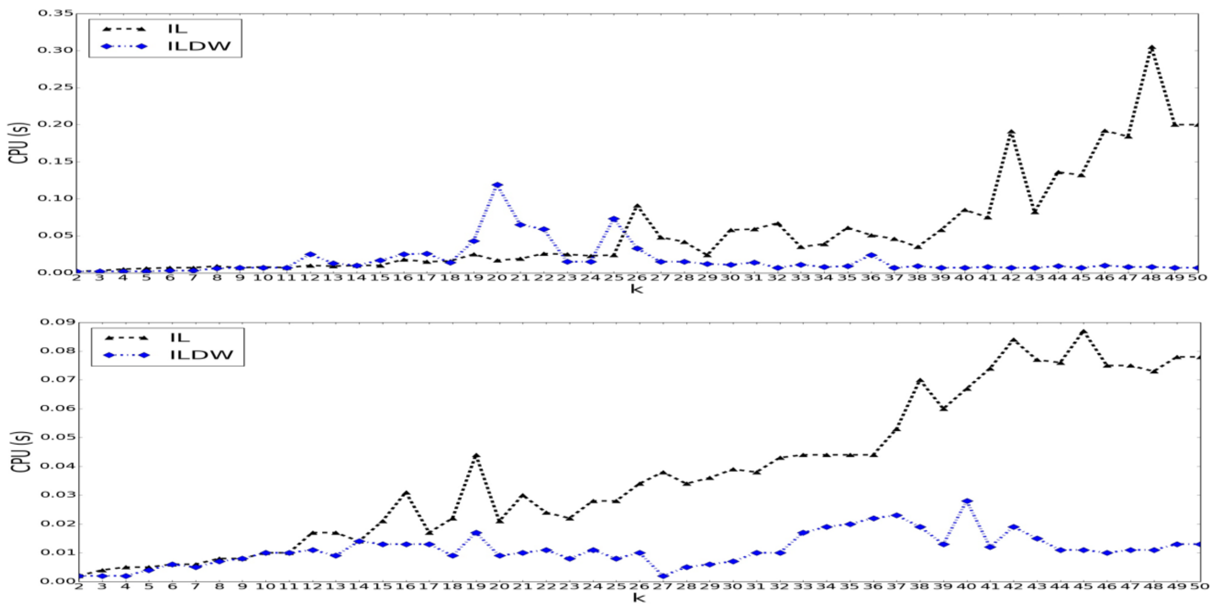

Figure 4. 14. Comparison of CPU time between supervised IL and unsupervised ILDW in scenario 1 (up) and scenario 2 (down).

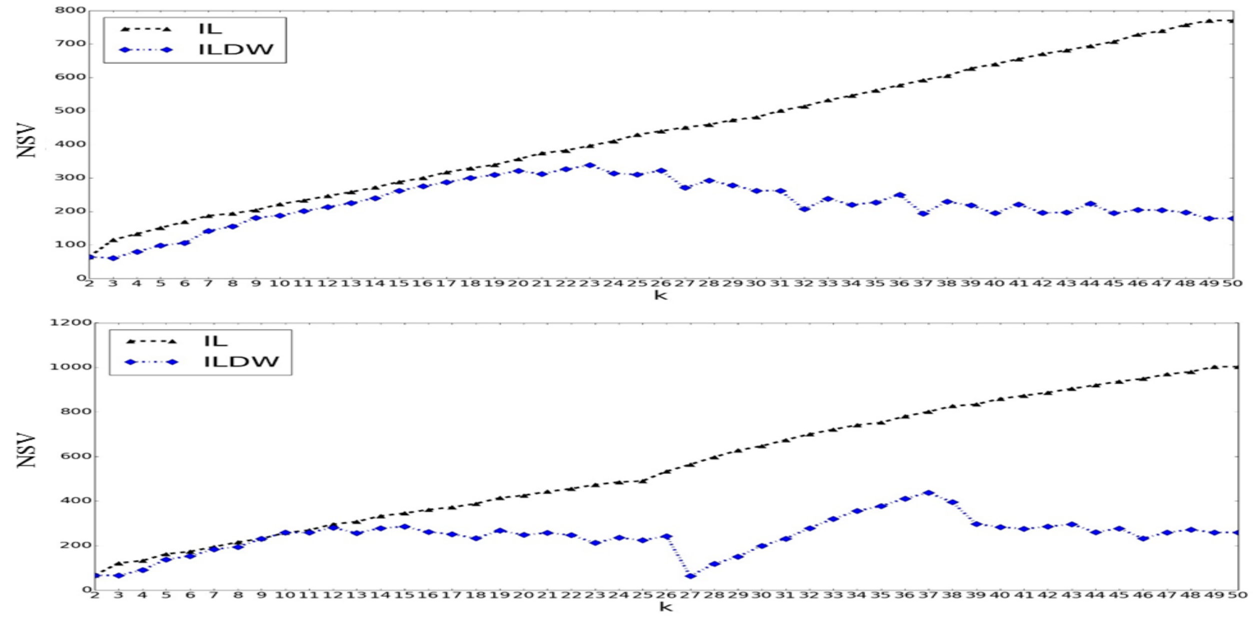

Figure 4. 15. Comparison of NSV between supervised IL and unsupervised ILDW in scenario 1 (up) and scenario 2 (down).

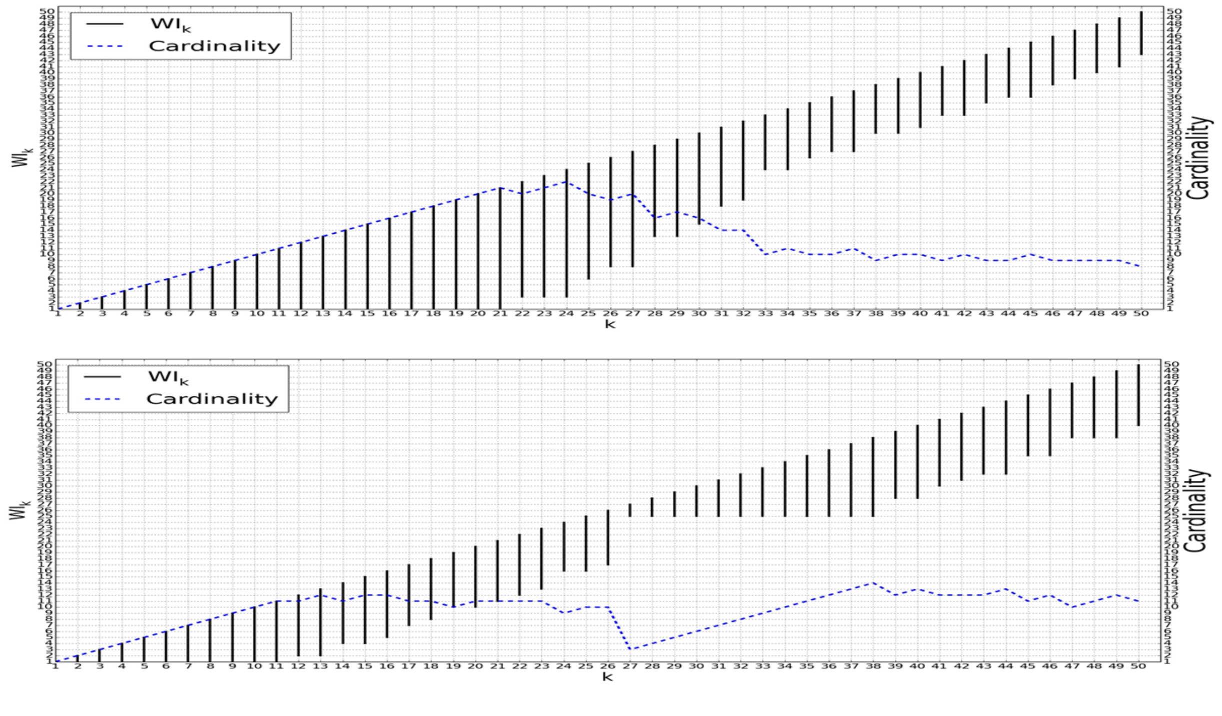

Figure 4. 16. WI_k and its cardinality for unsupervised ILDW in scenario 1 (up) and scenario 2 (down).

4.2.5 Conclusions

In order to address unsupervised FD updating for handling CD, ILDW algorithm in an unsupervised style is proposed. Results of updating for two hidden context scenarios prove that unsupervised ILDW algorithm by MGM score and NSV significantly surpasses the standard supervised IL procedures. For training supervised IL algorithm, completely correct labels are applied, but for unsupervised ILDW all the labels are provided by the proposed filtering algorithm. Thus, the supervised IL algorithm presents slight better performance in terms of F1 score. The F1 score reductions for unsupervised ILDW algorithm, in both applied scenarios, comparing with supervised IL, are less than 1%, and it is justifiable because of its filtering method that may keep some wrong labeled samples and remove some correct labeled samples.

Chapter 5: Conclusion

5.1 Conclusions and Contributions

By non stop condition monitoring, amount of available information gradually increases, and the new failure modes may appear. These failure modes could develop either as drift in existing classes or as new classes; therefore, those monitoring systems are required that can be updated, automatically and incrementally.

For FDD updating because of the new classes, the first step is to detect new failures, but typical FDD systems only recognize the failure modes that exist in the initial training phase. Moreover, suggested models that address ND are about only identifying new failures rather than present clear strategies for FDD updating.

For FDD updating because of the CD, primitive updating approaches integrate arriving and original samples and retrain FDD system. Updating with these traditional methods set off rapid growth of FDD model complexity and required CPU time. In order to cure these weaknesses, some promising studies have been reported although their performances need to be improved [179]. Furthermore, these methods address their applications, mainly, not for chemical engineering problems, but in text mining, video and image analysis, etc.

Therefore, this thesis develops supervised and unsupervised FDD updating frameworks for dealing with new classes/faults and CD while it is aimed for three objects. First, providing applicable frameworks that are efficient for meeting the goal (keeping the accuracy high) and cost (CPU time) of the FDD updating. Second, making information about CD magnitude to assist operators for decision making. Third, exploring data processing methods in order to improve FDD performance.

In chapter 2, and prior to studying FDD updating frameworks, data processing with three families of approaches, including imputation of missing values, feature selection, and feature extension are investigated. The purposes of the applied techniques are to improve the characteristics of data in different aspects. In the first family of approaches, application of three data driven techniques for dealing with noise, outliers and missing values are studied. Comparing with the other techniques, OK shows its reliability in tested scenarios while it assists classifier (SVM) to achieve even higher than 99.0 % accuracy in terms of F1 score. For feature selection, functionalities of the filter and wrapper methods with various criteria in FDD field are studied. The results confirm their influence over increasing FDD accuracy and reducing required CPU time, taking it into account that wrapper methods have better performance. For TE case study, the optimal features, which are selected by the best wrapper method, contrast with using all features improve FDD performance up to 196% on the scale of MGM score. For feature extension, the applicability of the error features, which are made by an observer, in the various scenarios of the process is investigated. The results prove that replacing the error features by the measured features enhance the FDD performance. As part of the results, replacing error features by measured features enhance $\overline{CAS}$ ($\overline{F1}$score) of the three classifiers, including SVM, GNB and DT up to 57.7%, 132.3% and 36.6%, respectively.

In chapter 3, supervised FD updating for coming up against CD in the process is studied. In the first section, DW algorithm is suggested; DW algorithm exploits the last available samples for providing data window whereas it has a strategy for forgetting redundant (old) samples in comparison with current concepts of the process. Furthermore, ILDW algorithm is proposed that is a hybrid approach combining the merits of the DW and IL algorithms. In four hidden context scenarios of CSTR benchmark, DW and ILDW outdo the IL in regard to MGM score, on average, by 44.5% and 43.6%, respectively. In the next section of the chapter, for tracing amount of CD in the process although implicitly, a framework is presented. It is based on non-automatic clustering, and it offers cda index with regard to "precision" definition in order to monitor CD amount changes in each class, separately. In the tested scenario, the framework detects 5% changes of hidden context during the operational time as 16% changes in cda_{k0} (index of normal class), and 32% changes in cda_{k1} (index of fault 1 class).

In chapter 4, unsupervised FDD updating is investigated. In the first section, an FDD updating framework versus novel faults is presented; the hybrid FDD updating framework is made up of automatic clustering and OCCs while FDD performance is enhanced by means of an observer. For the assessed static and dynamic scenarios through three tanks benchmark, results indicate 93.1% and 83.1% of success based on NPP for detecting samples in addition to performing FDD updating tasks, efficiently. In the next section of the chapter, and for unsupervised FD updating for handling CD, the ILDW algorithm is proposed in a different style. With the new ILDW scheme, the label of samples are predicted in an unsupervised way, and a strategy for filtering wrong labeled samples are applied. The applicability of this algorithm is verified by two scenarios in which ILDW comparing with the supervised IL algorithm improves MGM score up to 70.0%, in average.

5.2 Future Works

Considering thesis researches, it is found that the following topics have received less attention and deserved to be investigated. Here are summaries of them:

Integration of FDD with other process operations:

The majority of the studies in FDD are performed by ignoring the interactions of it with control and optimization systems. Obviously, these conditions make the applicability of the FDD methods far from real implementation. Designing the FDD framework while the roles and effects of control and optimization systems are considered is a topic that needs to be studied, deeply [7].

Designing advanced FDD updating system:

Most of the studies, which have addressed FDD updating problem, are developed to handle either new faults or CD. Studying a framework that could update the FDD whereas both types of new conditions occur in the process is a topic that must be explored [181] [111].

Improving implementation and algorithms of OCC:

OCCs are useful type of classifiers that have ND usages in FDD. In this thesis OCCs are applied, and based on the results, in the dynamic conditions of the process they have relatively unreliable performance. This weakness should be investigated in two lines; first, a new scheme for applying them just as tools that could compensate dynamic states of the process, and second improving their algorithms and kernel functions [69] [157].

5.3 Published Contributions

During the development of the thesis, the following contributions have been peer-reviewed and accepted for presentation and/or publication in different international journals and conference proceedings:

1) Shokry, M. H. Ardakani, G. Escudero, M. Graells, and A. Espuña, "Dynamic kriging based fault detection and diagnosis approach for nonlinear noisy dynamic processes,"

Computer. Chem. Eng., vol. 106, pp. 758–776, Nov. 2016. DOI: https://doi.org/10.1016/B978-0-444-63428-3.50014-X

2) M. H. Ardakani, G. Escudero, M. Graells, and A. Espuña, "Incremental Learning Fault Detection Algorithm Based on Hyperplane-Distance," Computer Aided Chemical Engineering, vol. 38, 2016, pp. 1105–1110. DOI: https://doi.org/10.1016/B978-0-444-63428-3.50189-2

3) M. H Ardakani, M. Askarian, A. Shokry, G. Escudero, M. Graells, and A. Espuña, "Optimal Feature Selection for Designing a Fault Diagnosis System," Computer Aided Chemical Engineering, vol. 38, pp. 1111–1116, 2016. DOI: https://doi.org/10.1016/B978-0-444-63428-3.50190-9

4) M. H Ardakani, A. Shokry, G. Saki, G. Escudero, M. Graells, and A. Espuña, "Imputation of Missing Data with Ordinary Kriging for Enhancing Fault Detection and Diagnosis," Computer Aided Chemical Engineering, vol. 38, 2016, pp. 1377–1382. DOI: https://doi.org/10.1016/B978-0-444-63428-3.50234-4

5) M. H. Ardakani, A. Shokry, G. Escudero, M. Graells, and A. Espuna, "A framework for Unsupervised Fault detection and diagnosis Based on Clustering assisted Kriging Observer," 3rd Conference on Control and Fault-Tolerant Systems (SysTol), 2016, pp. 183–188.

6) Shokry, M. H Ardakani, G. Escudero, M. Graells, and A. Espuña, "Kriging based Fault Detection and Diagnosis Approach for Nonlinear Noisy Dynamic Processes," Computer Aided Chemical Engineering, vol. 38, Elsevier, 2016, pp. 55–60. DOI: https://doi.org/10.1016/B978-0-444-63428-3.50014-X

7) M. H Ardakani, M. Askarian, G. Escudero, M. Graells, and A. Espuña, "Toward Online Explore of Concept Drift for Fault Detection of Chemical Processes," Computer Aided Chemical Engineering, vol. 40, Elsevier, 2017, pp. 1657–1662. DOI: https://doi.org/10.1016/B978-0-444-63965-3.50278-6

8) M. H Ardakani, G. Escudero, M. Graells, and A. Espuña, "Sliding Dynamic Data Window: Improving Properties of the Incremental Learning Methods" Computer Aided Chemical

Engineering, vol. 40, Elsevier, 2017 pp. 1663–1668. DOI: https://doi.org/10.1016/B978-0-444-63965-3.50279-8

9) M. H Ardakani, A. Shokry, G. Escudero, M. Graells, and A. Espuña, “Sliding Dynamic Data Window: Online Quantification of the Concept Drift Using Incremental Learned Classifier and Non-automatic Clustering” Computer Aided Chemical Engineering, vol. 41, Elsevier, 2018,

10) M. H Ardakani, A. Shokry, G. Escudero, M. Graells, and A. Espuña, “Unsupervised Automatic Updating of Classification Models of Fault Diagnosis” Computer Aided Chemical Engineering, vol. 41, Elsevier, 2018,

References

[1] S. Dash and V. Venkatasubramanian, “Challenges in the industerial applications of fault diagnostic systems,” *Comput. Chem. Eng.*, vol. 24, no. 2–7, pp. 785–791, Jul. 2000.

[2] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, “A review of process fault detection and diagnosis,” *Comput. Chem. Eng.*, vol. 27, no. 3, pp. 293–311, Mar. 2003.

[3] S. Al-dahidi, F. Di Maio, P. Baraldi, and E. Zio, “Ensemble Clustering for Fault Diagnosis in Industrial Plants,” *Chem. Eng. Trans.*, vol. 43, 2015.

[4] B. S. J. Costa, P. P. Angelov, and L. A. Guedes, “A new unsupervised approach to fault detection and identification,” *Proc. Int. Jt. Conf. Neural Networks*, pp. 1557–1564, 2014.

[5] X. Wang, H. Feng, and Y. Fan, “Fault detection and classification for complex processes using semi-supervised learning algorithm,” *Chemom. Intell. Lab. Syst.*, vol. 149, pp. 24–32, 2015.

[6] I. Monroy, R. Benitez, G. Escudero, and M. Graells, “A semi-supervised approach to fault diagnosis for chemical processes,” *Comput. Chem. Eng.*, vol. 34, no. 5, pp. 631–642, May 2010.

[7] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, “A review of process fault detection and diagnosis Part III: Process history based methods,” *Comput. Chem. Eng.*, vol. 27, no. 3, pp. 327–346, Mar. 2003.

[8] S. J. Qin, “Survey on data-driven industrial process monitoring and diagnosis,” *Annu. Rev. Control*, vol. 36, no. 2, pp. 220–234, Dec. 2012.

[9] V. Lemaire, C. Salperwyck, and A. Bondu, “A Survey on Supervised Classification on Data Streams,” in *Business Intelligence: 4th European Summer School, eBISS 2014, Berlin, Germany, July 6-11, 2014, Tutorial Lectures*, E. Zimányi and R.-D. Kutsche, Eds. Cham: Springer International Publishing, 2015, pp. 88–125.

[10] C. K. Lau, K. Ghosh, M. a. Hussain, and C. R. Che Hassan, “Fault diagnosis of Tennessee Eastman process with multi-scale PCA and ANFIS,” *Chemom. Intell. Lab. Syst.*, vol. 120, pp. 1–14, Jan. 2013.

[11] S. Yin, S. X. Ding, A. Haghani, H. Hao, and P. Zhang, “A comparison study of basic data-

driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process," *J. Process Control*, vol. 22, no. 9, pp. 1567–1581, Oct. 2012.

[12] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven Soft Sensors in the process industry," *Comput. Chem. Eng.*, vol. 33, pp. 795–814, 2009.

[13] T. Singh and M. Kantardzic, "On the reliable detection of concept drift from streaming unlab ele d data," *Expert Syst. Appl.*, vol. 82, pp. 77–99, 2017.

[14] S. Xu and J. Wang, "A fast incremental extreme learning machine algorithm for data streams classification," *Expert Syst. Appl.*, vol. 65, pp. 332–344, 2016.

[15] Z. Ouyang, Y. Gao, Z. Zhao, and T. Wang, "Study on the classification of data streams with concept drift," *2011 Eighth Int. Conf. Fuzzy Syst. Knowl. Discov.*, vol. 3, pp. 1673–1677, 2011.

[16] M. Markou and S. Singh, "Novelty detection: A review - Part 1: Statistical approaches," *Signal Processing*, vol. 83, pp. 2481–2497, 2003.

[17] M. Markou and S. Singh, "Novelty detection: A review - Part 2:: Neural network based approaches," *Signal Processing*, vol. 83, pp. 2499–2521, 2003.

[18] A. Lemos, W. Caminhas, and F. Gomide, "Adaptive fault detection and diagnosis using an evolving fuzzy classifier," *Inf. Sci. (Ny).*, vol. 220, pp. 64–85, Jan. 2013.

[19] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Comput. Appl.*, vol. 19, no. 2, pp. 263–282, 2010.

[20] V. Venkatasubramanian, "Abnormal events management in complex process plants: challenges and opportunities in intelligent supervisory control," *Proc. FOCAPO*, 2003.

[21] T. J. Rato and M. S. Reis, "Fault detection in the Tennessee Eastman benchmark process using dynamic principal components analysis based on decorrelated residuals (DPCA-DR)," *Chemom. Intell. Lab. Syst.*, vol. 125, pp. 101–108, Jun. 2013.

[22] A. Ben Ayed, M. Ben Halima, and A. M. Alimi, "Survey on clustering methods : Towards fuzzy clustering for big data," *Int. Conf. Soft Comput. Pattern Recognit.*, pp. 331–336, 2014.

[23] B. Malley, D. Ramazzotti, and J. T. Wu, “Data Pre-processing,” in *Secondary Analysis of Electronic Health Records*, Cham: Springer International Publishing, 2016.

[24] R. Isermann, “Model-Based Fault Detection and Diagnosis - Status and Applications,” *IFAC Proc. Vol.*, vol. 37, no. 6, pp. 49–60, 2004.

[25] P. M. Frank and X. Ding, “Survey of robust residual generation and evaluation methods in observer-based fault detection systems,” *J. Process Control*, vol. 7, no. 6, pp. 403–424, 1997.

[26] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. 1973.

[27] R. J. Patton, J. Chen, and T. M. Siew, “Fault Diagnosis in Nonlinear Dynamic Systems via Neural Networks,” *Proc. IEE Int.Conf. Control*, no. 389, pp. 1–6, 1994.

[28] A. Shokry, M. H. Ardakani, G. Escudero, M. Graells, and A. Espuña, “Dynamic kriging based fault detection and diagnosis approach for nonlinear noisy dynamic processes,” *Comput. Chem. Eng.*, 2017.

[29] L. H. Chiang, M. E. Kotanchek, and A. K. Kordon, “Fault diagnosis based on Fisher discriminant analysis and support vector machines,” *Comput. Chem. Eng.*, vol. 28, no. 8, pp. 1389–1401, Jul. 2004.

[30] F. Camci and R. B. Chinnam, “General support vector representation machine for one-class classification of non-stationary classes,” *Pattern Recognit.*, vol. 41, no. 10, pp. 3021–3034, Oct. 2008.

[31] J. Weston and C. Watkins, “Support Vector Machines for Mlulti C1ass Pattern Recognition,” in *Proceedings of ESANN99*, 1999.

[32] E. J. Bredensteiner and K. P. Bennett, “Multicategory Classification by Support Vector Machines,” *Comput. Optim. Appl.*, vol. 12, no. 1, pp. 53–79, Jan. 1999.

[33] D. M. Farid, L. Zhang, C. M. Rahman, M. A. Hossain, and R. Strachan, “Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks,” *Expert Syst. Appl.*, vol. 41, no. 4, Part 2, pp. 1937–1946, 2014.

[34] G. Cauwenberghs and T. Poggio, “Incremental and Decremental Support Vector Machine

Learning," in *Advances in neural information processing systems,* 2001, pp. 409–415.

[35] I. Yélamos, G. Escudero, M. Graells, and L. Puigjaner, "Performance assessment of a novel fault diagnosis system based on support vector machines," *Comput. Chem. Eng.*, vol. 33, no. 1, pp. 244–255, Jan. 2009.

[36] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast Kernel Classifiers with Online and Active Learning," *Mach. Learn. Res.*, vol. 6, pp. 1579–1619, 2005.

[37] F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, "On-line independent support vector machines," *Pattern Recognit.*, vol. 43, no. 4, pp. 1402–1412, Apr. 2010.

[38] M. Jia, H. Xu, X. Liu, and N. Wang, "The optimization of the kind and parameters of kernel function in KPCA for process monitoring," *Comput. Chem. Eng.,* vol. 46, pp. 94–104, Nov. 2012.

[39] H. Duan, X. Shao, W. Hou, G. He, and Q. Zeng, "An incremental learning algorithm for Lagrangian support vector machines," *Pattern Recognit. Lett.*, vol. 30, no. 15, pp. 1384–1391, Nov. 2009.

[40] Y. Chu, S. J. Qin, and C. Han, "Fault Detection and Operation Mode Identification Based on Pattern Classification with Variable Selection," *Framework*, pp. 1701–1710, 2004.

[41] J. R. Quinlan, "15 - LEARNING EFFICIENTCLASSIFICATION PROCEDURES AND THEIR APPLICATION TO CHESS END GAMES," in *Machine Learning*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. San Francisco (CA): Morgan Kaufmann, 1983, pp. 463–482.

[42] B. Özyurt, A. K. Sunol, M. C. Çamurdan, P. Mogili, and L. O. Hall, "Chemical plant fault diagnosis through a hybrid symbolic-connectionist machine learning approach," *Comput. Chem. Eng.*, vol. 22, no. 1, pp. 299–321, 1998.

[43] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, 2001.

[44] O. Addin, S. . Sapuan, M. Othman, and B. . Ahmed Ali, "Comparison Naïve bayes classifier with back Comparison of Naïve propagation neural network classifier based on f - folds

feature extraction algorithm for ball bearing fault diagnostic system," vol. 6, no. 13, pp. 3181–3188, 2011.

[45] F. Provost and V. Kolluri, "A Survey of Methods for Scaling Up Inductive Algorithms," *Data Min. Knowl. Discov.*, vol. 3, no. 2, pp. 131–169, Jun. 1999.

[46] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[47] S. Luca, D. A. Clifton, and B. Vanrumste, "One-class classification of point patterns of extremes," *J. Mach. Learn. Res.*, vol. 17, no. 191, pp. 1–21, 2016.

[48] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognit.*, vol. 40, pp. 863–874, 2007.

[49] S. Das, A. Abraham, and A. Konar, "Automatic Clustering Using an Improved Differential Evolution Algorithm," *IEEE Trans. Syst. man Cybern. A Syst. humans*, vol. 38, no. 1, pp. 218–237, 2008.

[50] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering : A Review," *ACM Comput. Surv.*, vol. 31, no. 3, 1999.

[51] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and Semi-supervised Clustering: A Brief Survey," *A Rev. Mach. Learn. Tech. Process. Multimed. Content, Rep. MUSCLE Eur. Netw. Excell.*, pp. 1–12, 2004.

[52] G. Ahalya and H. M. Pandey, "Data Clustering Approaches Survey and Analysis," *1st Int. Conf. Futur. trend Comput. Anal. Knowl. Manag.*, pp. 532–537, 2015.

[53] Q. P. He, S. J. Qin, and J. Wang, "A new fault diagnosis method using fault directions in Fisher discriminant analysis," *AIChE J.*, vol. 51, no. 2, pp. 555–571, 2005.

[54] Y. Zhang and S. J. Qin, "Improved nonlinear fault detection technique and statistical analysis," *AIChE J.*, vol. 54, no. 12, pp. 3207–3220, Dec. 2008.

[55] C.-H. Chou, M.-C. Su, and E. Lai, "A new cluster validity measure and its application to image compression," *Pattern Anal Applic*, vol. 7, pp. 205–220, 2004.

[56] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Trans. Pattern Anal.*

Mach. Intell., vol. 1, no. 2, pp. 224–227, 1979.

[57] F. Boukouvala, F. J. Muzzio, and M. G. Ierapetritou, "Dynamic data-driven modeling of pharmaceutical processes," *Ind. Eng. Chem. Res.*, vol. 50, no. 11, pp. 6743–6754, 2011.

[58] A. Shokry and A. Espuña, "Sequential dynamic optimization of complex nonlinear processes based on kriging surrogate models," *Procedia Technol.*, vol. 15, pp. 376–387, 2014.

[59] P. Wang, C. Yang, X. Tian, and D. Huang, "Adaptive Nonlinear Model Predictive Control Using an On-line Support Vector Regression Updating Strategy," *Chinese J. Chem. Eng.*, vol. 22, no. 7, pp. 774–781, Jul. 2014.

[60] R. Debnath, M. Muramatsu, and H. Takahashi, "An Efficient Support Vector Machine Learning Method with Second-Order Cone Programming for Large-Scale Problems," *Appl. Intell.*, vol. 23, no. 3, pp. 219–239, Dec. 2005.

[61] P. Laskov, C. Gehl, S. Kruger, and K.-R. Muller, "Incremental Support Vector Learning : Analysis , Implementation and Applications," *J. Mach. Learn. Res.*, vol. 7, pp. 1909–1936, 2006.

[62] Y. Zhang and Y. Zhang, "Fault detection of non-Gaussian processes based on modified independent component analysis," *Chem. Eng. Sci.*, vol. 65, no. 16, pp. 4630–4639, Aug. 2010.

[63] L. Gosselin, M. Tye-Gingras, and F. Mathieu-Potvin, "Review of utilization of genetic algorithms in heat transfer problems," *Int. J. Heat Mass Transf.*, vol. 52, no. 9–10, pp. 2169–2188, 2009.

[64] G. Jones, "Genetic and evolutionary algorithms," *Encycl. Comput. Chem.*, vol. 2, pp. 1127–1136, 1998.

[65] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances," *Inf. Sci. (Ny).*, vol. 355–356, pp. 127–151, 2016.

[66] D. Mylaraswamy and V. Venkatasubramanian, "A Hybrid Framework for Large Scale

Process Fault Diagnosis," *Comput. Chem. Eng.*, vol. 21, 1997.

[67] F. Pereira, T. Mitchell, and M. Botvinick, "Machine learning classifiers and fMRI: A tutorial overview," *Neuroimage*, vol. 45, no. 1, pp. S199–S209, 2009.

[68] T. Fawcett, "An introduction to ROC analysis," vol. 27, pp. 861–874, 2006.

[69] H. Chen, P. Tiňo, and X. Yao, "Cognitive fault diagnosis in Tennessee Eastman Process using learning in the model space," *Comput. Chem. Eng.*, vol. 67, pp. 33–42, Aug. 2014.

[70] T. Fawcett, "ROC Graphs : Notes and Practical Considerations for Researchers," *Mach. Learn.*, vol. 31, pp. 1–38, 2004.

[71] A. Bifet, J. Read, I. . Žliobait\.e, B. Pfahringer, and G. Holmes, "Pitfalls in Benchmarking Data Stream Classification and How to Avoid Them," in *Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 465–479.

[72] I. . Žliobait\.e, A. Bifet, J. Read, B. Pfahringer, and G. Holmes, "Evaluation methods and decision theory for classification of streaming data with temporal dependence," *Mach. Learn.*, vol. 98, no. 3, pp. 455–482, Mar. 2015.

[73] V. Lemaire, C. Salperwyck, and A. Bondu, "A survey on supervised classification on data streams," *Bus. Intell.*, pp. 88–125, 2015.

[74] E. R. Faria, I. J. C. R. Gonçalves, A. C. P. L. F. de Carvalho, and J. Gama, "Novelty detection in data streams," *Artif. Intell. Rev.*, vol. 45, no. 2, pp. 235–269, Feb. 2016.

[75] H. Chen, A. Kremling, and F. Allgöwer, "Nonlinear Predictive Control of a Benchmark CSTR," *Proc. Eur. Control Conf.*, no. 1, pp. 3247–3252, 1995.

[76] J. Liu, "Fault diagnosis using contribution plots without smearing effect on non-faulty variables," *J. Process Control*, vol. 22, no. 9, pp. 1609–1623, Oct. 2012.

[77] W. S. Yip and T. E. Marlin, "Multiple data sets for model updating in real-time operations optimization," *Comput. Chem. Eng.*, vol. 1354, no. 2, 2002.

[78] J. Liu and D.-S. Chen, "Fault isolation using modified contribution plots," *Comput. Chem. Eng.*, vol. 61, pp. 9–19, Feb. 2014.

[79] A. Kouadri, M. A. Aitouche, and M. Zelmat, “Variogram-based fault diagnosis in an interconnected tank system,” *ISA Trans.*, vol. 51, no. 3, pp. 471–476, 2012.

[80] M. R. Maurya, R. Rengaswamy, and V. Venkatasubramanian, “Fault diagnosis using dynamic trend analysis: A review and recent developments,” *Eng. Appl. Artif. Intell.*, vol. 20, no. 2, pp. 133–146, Mar. 2007.

[81] M. a. Bin Shams, H. M. Budman, and T. a. Duever, “Fault detection, identification and diagnosis using CUSUM based PCA,” *Chem. Eng. Sci.*, vol. 66, no. 20, pp. 4488–4498, Oct. 2011.

[82] J. Yu, “A particle filter driven dynamic Gaussian mixture model approach for complex process monitoring and fault diagnosis,” *J. Process Control*, vol. 22, no. 4, pp. 778–788, Apr. 2012.

[83] J. . J. Downs and E. F. Vogel, “A PLANT-WIDE INDUSTRIAL CONTROL PROBLEM PROCESS,” *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993.

[84] Y. Zhang, “Enhanced statistical analysis of nonlinear processes using KPCA, KICA and SVM,” *Chem. Eng. Sci.*, vol. 64, no. 5, pp. 801–811, Mar. 2009.

[85] F. Yang, Y. Zhang, F. Ciucci, Z. Wu, S. Wang, Y. Wang, and Z. Zhang, “Towards a consistent understanding of the metal hydride reaction kinetics : Measurement , modeling and data processing,” *J. Alloys Compd.*, vol. 741, pp. 610–621, 2018.

[86] Z. Ge, Z. Song, and F. Gao, “Review of Recent Research on Data-Based Process Monitoring,” *Am. Chem. Soc.*, vol. 52, no. 4, pp. 3543–3262, 2013.

[87] M. Askarian, G. Escudero, M. Graells, R. Zarghami, F. Jalali-Farahani, and N. Mostoufi, “Fault diagnosis of chemical processes with incomplete observations: A comparative study,” *Comput. Chem. Eng.*, vol. 84, pp. 104–116, 2016.

[88] K. Pelckmans, J. De Brabanter, J. A. K. Suykens, and B. De Moor, “Handling missing values in support vector machine classifiers,” vol. 18, pp. 684–692, 2005.

[89] G. Chechik, G. Heitz, G. Elidan, P. Abbeel, and D. Koller, “Max-margin Classification of Data with Absent Features,” *J. Mach. Learn. Res.*, vol. 9, pp. 1–21, 2008.

[90] S. Krause and R. Polikar, “An ensemble of classifiers approach for the missing feature problem,” in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, 2003, vol. 1, pp. 553–558 vol.1.

[91] J. L. Schafer and J. W. Graham, “Missing data: our view of the state of the art.,” *Psychol. Methods*, vol. 7, no. 2, pp. 147–177, 2002.

[92] K. Lakshminarayan, S. A. HARP, and T. SAMAD, “Imputation of Missing Data in Industrial Databases,” *Appl. Intell.*, vol. 11, pp. 259–275, 1999.

[93] Y. Zhang, C. Zhang, and W. Zhang, “Statistical Analysis of Nonlinear Processes Based on Penalty Factor,” vol. 2014, 2014.

[94] A. L. Blum and P. Langley, “Selection of relevant features and examples in machine learning,” *Artif. Intell.*, vol. 97, no. 1–2, pp. 245–271, 1997.

[95] G. John, R. Kohavi, and K. Pfleger, “Irrelevant Features and the Subset Selection Problem,” in *Icml*, 1994, pp. 121–129.

[96] S. H. Huang, “Supervised feature selection: A tutorial,” *Artif. Intell. Res.*, vol. 4, no. 2, 2015.

[97] I. Guyon and A. Elisseeff, “An Introduction to Variable and Feature Selection,” *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[98] D. Zhang, S. Chen, and Z. Zhou, “Constraint Score : A new filter method for feature selection with pairwise constraints,” *Pattern Recognit.*, vol. 41, pp. 1440–1451, 2008.

[99] R. Kohavi and H. John, “Artificial Intelligence Wrappers for feature subset selection,” *Artif. Intell.*, vol. 97, pp. 273–324, 1997.

[100] L. Yu and H. Liu, “Efficient Feature Selection via Analysis of Relevance and Redundancy,” *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, 2004.

[101] B. Auffarth, M. López, and J. Cerquides, “Comparison of redundancy and relevance measures for feature selection in tissue classification of CT images,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6171 LNAI, pp. 248–262, 2010.

[102] J. C. Rajapakse, L. Wong, and R. Acharya, “Pattern Recognition in Bioinformatics: An

Introduction," in *Pattern Recognition in Bioinformatics*, 2006, pp. 1–3.

[103] C. D. Hanchuan Peng, "Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1226–1238, 2005.

[104] N. Kwak and C.-H. Choi, "Input Feature Selection by Mutual Information Based on Parzen Window," vol. 24, no. 12, pp. 1667–1671, 2002.

[105] I. M. Chora, "An investigation on automatic systems for fault diagnosis in chemical processes," 2011.

[106] L. Guo, Y. Zhang, H. Wang, and J. Fang, "Observer-Based Optimal Fault Detection and Probability Distributions," vol. 54, no. 10, pp. 3712–3719, 2006.

[107] R. H. Chen, D. L. Mingori, and J. L. Speyer, "Optimal stochastic fault detection filter," *Automatica*, vol. 39, no. 3, pp. 377–390, 2003.

[108] A. Shokry, M. H. Ardakani, G. Escudero, M. Graells, and A. Espuña, "Dynamic kriging based fault detection and diagnosis approach for nonlinear noisy dynamic processes," *Comput. Chem. Eng.*, vol. 106, pp. 758–776, Nov. 2017.

[109] A. Xu and Q. Zhang, "Nonlinear system fault diagnosis based on adaptive estimation," *Automatica*, vol. 40, no. 7, pp. 1181–1193, 2004.

[110] Y. Yang, S. X. Ding, and L. Li, "Systems & Control Letters On observer-based fault detection for nonlinear systems," *Syst. Control Lett.*, vol. 82, pp. 18–25, 2015.

[111] M. H. Ardakani, A. Shokry, G. Escudero, M. Graells, and A. Espuna, "A framework for Unsupervised Fault detection and diagnosis Based on Clustering assisted Kriging Observer," in *3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, 2016, pp. 183–188.

[112] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, 1996.

[113] P. Kadlec, R. Grbić, and B. Gabrys, "Review of adaptation mechanisms for data-driven soft sensors," *Comput. Chem. Eng.*, vol. 35, no. 1, pp. 1–24, Jan. 2011.

[114] J. Gama, *Knowledge discovery from data streams*. Chapman and Hall/CRC Press, 2010.

[115] R. Polikar, L. Udpa, and V. Honavar, “Learn ++ : An Incremental Learning Algorithm for,” *Trans. Syst. man, Cybern. c Appl. Rev.*, vol. 31, no. 4, pp. 497–508, 2001.

[116] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, “Mining data streams: a review,” *SIGMOD Rec.*, vol. 34, no. 2, pp. 18–26, 2005.

[117] G. Dong, J. Han, P. S. Yu, L. V. S. Lakshmanan, J. Pei, and H. Wang, “Online Mining of Changes from Data Streams : Research Problems and Preliminary Results,” *ACM SIGMOD MPDS ’03 San*, pp. 11–13, 2003.

[118] J. C. Schlimmer and D. Fisher, “A Case Study of Incremental Concept Induction,” *Kehler, T., Rosenschein, S. (Eds.), Proc. Fifth Natl. Conf. Artificial Intell.*, vol. 1, pp. 496–501, 1986.

[119] P. E. Utgoff, “Incremental Induction of Decision Trees,” *Mach. Learn.*, vol. 4, no. 2, pp. 161–186, Nov. 1989.

[120] C. Domeniconi and D. Gunopulos, “Incremental support vector machine construction,” *Proc. 2001 IEEE Int. Conf. Data Min.*, pp. 589–592, 2001.

[121] J. C. Schlimmer and R. H. Granger, “Incremental learning from noisy data,” *Mach. Learn.*, vol. 1, no. 3, pp. 317–354, 1986.

[122] J. Sankaranarayanan, H. Samet, and A. Varshney, “A fast all nearest neighbor algorithm for applications involving large point-clouds,” *Comput. Graph.*, vol. 31, no. 2, pp. 157–174, 2007.

[123] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, “Models and Issues in Data Stream Systems,” *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, pp. 1–16, 2002.

[124] J. a Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. De Carvalho, and J. Gama, “Data stream clustering,” *ACM Comput. Surv.*, vol. 46, no. 1, pp. 1–31, 2013.

[125] M. Z. Hayat and M. R. Hashemi, “A DCT Based Approach for Detecting Novelty and Concept Drift in Data Streams,” in *In International conference of soft computing and*

pattern recognition, 2010, pp. 373–378.

[126] E. J. Spinosa, A. P. de L. F., D. Carvalho, and J. Gama, "OLINDDA : A cluster-based approach for detecting novelty and concept drift in data streams," in *Proceedings of the 2007 ACM symposium on applied computing*, 2007, pp. 448–452.

[127] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," *Ninth ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 2, no. 1, pp. 226--235, 2003.

[128] B. Krawczyk and M. Woźniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift," *Soft Comput.*, vol. 19, no. 12, pp. 3387–3400, 2015.

[129] C. C. Aggarwal, J. Han, J. Wang UIUC, and P. S. Yu, "On Demand Classification of Data Streams," *ACM Conf. Knowl. Discov. Data Min.*, pp. 503–508, 2004.

[130] K. Nishida and K. Yamauchi, "Detecting Concept Drift Using Statistical Testing," in *Springer-Verlag*, 2007, pp. 264–269.

[131] P. Sobhani and H. Beigy, "New Drift Detection Method for Data Streams," in *Springer-Verlag*, 2011, pp. 88–97.

[132] R. Klinkenberg and I. Renz, "Adaptive information filtering: Learning in the presence of concept drifts," *Work. Notes ICML/AAAI-98 Work. Learn. Text Categ.*, pp. 33–40, 1998.

[133] A. Bifet, R. Gavalda, and R. Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing.," *Sdm*, vol. 7, p. 2007, 2007.

[134] B. Brian, D. Mayur, and M. Rajeev, "Sampling From a Moving Window Over Streaming Data," 2000.

[135] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with Drift Detection," in *A. Bazzan & S. Labidi (Eds.), Advances in Artificial Intelligence – SBIA 2004. Lecture Notes in Computer Science (Vol. 3171, pp. 66–112). Berlin/ Heidelberg: Springer*, 2004.

[136] A. Bifet and R. Gavald, "Learning from Time-Changing Data with Adaptive Windowing a," in *Proceedings of the seventh SIAM international conference on data mining, SDM'07* .

Lake Buena Vista, Florida, USA: SIAM., pp. 443–448.

[137] S. H. Bach and M. A. Maloof, “Paired Learners for Concept Drift,” in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 23–32.

[138] C. Li, K. Liu, and H. Wang, “The incremental learning algorithm with support vector machine based on hyperplane-distance,” *Appl. Intell.*, vol. 34, no. 1, pp. 19–27, Apr. 2009.

[139] K. K. Syed, N.A., Liu, H., Sung, “Incremental Learning with Support Vector Machines,” *Proc. ACM SIGKDD Internat. Conf. Knowl. Discov. Data Min.*, 1999.

[140] R. Xiao, J. Wang, and F. Zhang, “An approach to incremental SVM learning algorithm,” *Proc. - ISECS Int. Colloq. Comput. Commun. Control. Manag. CCCM 2000*, vol. 1, no. 1, pp. 352–354, 2000.

[141] M. Baena-garc, J. GDel Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales Bueno, “Early Drift Detection Method,” in *International workshop on knowledge discovery from data streams, IWKDDS’06*, 2006, pp. 77–86.

[142] D. Rafael, D. L. Cabral, R. Souto, and M. De Barros, “Concept drift detection based on Fisher’s Exact test,” *Inf. Sci. (Ny).*, vol. 442–443, pp. 220–234, 2018.

[143] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, Apr. 1996.

[144] M. H. Ardakani, M. Askarian, G. Escudero, M. Graells, and A. Espuna, “Toward Online Explore of Concept Drift for Fault Detection of Chemical Processes,” in *Computer Aided Chemical Engineering*, 2017, vol. 40, pp. 1657–1662.

[145] B. J. Ohran, D. Munoz de la pena, and J. F. Davis, “Enhancing Data-based Fault Isolation Through Nonlinear Control,” *AIChE J.*, vol. 54, no. 1, pp. 223–241, 2008.

[146] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, “An experimental evaluation of novelty detection methods,” *Neurocomputing*, vol. 135, pp. 313–327, 2014.

[147] P. Perner, “Concepts for novelty detection and handling based on a case-based reasoning process scheme,” *Eng. Appl. Artif. Intell.*, vol. 22, no. 1, pp. 86–91, 2009.

[148] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, “A review of novelty

detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[149] H. j. Lee and S. J. Roberts, "On-line novelty detection using the Kalman filter and extreme value theory," in *2008 19th International Conference on Pattern Recognition*, 2008, pp. 1–4.

[150] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Comput. Surv.*, vol. 41, no. 3, p. 15:1--15:58, Jul. 2009.

[151] R. K. Pearson, "Outliers in process modeling and identification," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 1, pp. 55–63, 2002.

[152] V. J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, Oct. 2004.

[153] A. Rusiecki, "Robust Neural Network for Novelty Detection on Data Streams," in *Artificial Intelligence and Soft Computing*, 2012, pp. 178–186.

[154] D. M. Farid, L. Zhang, A. Hossain, C. M. Rahman, R. Strachan, G. Sexton, and K. Dahal, "An adaptive ensemble classifier for mining concept drifting data streams," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5895–5906, 2013.

[155] D. M. Farid and C. M. Rahman, "Novel class detection in concept-drifting data stream mining employing decision tree," in *2012 7th International Conference on Electrical and Computer Engineering*, 2012, pp. 630–633.

[156] T. M. Al-Khateeb, M. M. Masud, L. Khan, and B. Thuraisingham, "Cloud Guided Stream Classification Using Class-Based Ensemble," in *2012 IEEE Fifth International Conference on Cloud Computing*, 2012, pp. 694–701.

[157] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the Support of a High-Dimensional Distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, Jul. 2001.

[158] G. G. Cabral, A. L. I. Oliveira, and C. B. G. Cahú, "Combining nearest neighbor data description and structural risk minimization for one-class classification," *Neural Comput. Appl.*, vol. 18, no. 2, pp. 175–183, Feb. 2009.

[159] D. P. Filev, R. B. Chinnam, F. Tseng, and P. Baruah, "An Industrial Strength Novelty Detection Framework for Autonomous Equipment Monitoring and Diagnostics," *IEEE Trans. Ind. Informatics*, vol. 6, no. 4, pp. 767–779, 2010.

[160] S. Ntalampiras, I. Potamitis, and N. Fakotakis, "Probabilistic Novelty Detection for Acoustic Surveillance Under Real-World Conditions," *IEEE Trans. Multimed.*, vol. 13, no. 4, pp. 713–719, 2011.

[161] Y. Li, "Selecting training points for one-class support vector machines," *Pattern Recognit. Lett.*, vol. 32, no. 11, pp. 1517–1522, 2011.

[162] P. Juszczak, D. M. J. Tax, E. Pe¸kalska, and R. P. W. Duin, "Minimum spanning tree based one-class classifier," *Neurocomputing*, vol. 72, no. 7, pp. 1859–1869, 2009.

[163] G. G. Cabral and A. L. I. Oliveira, "A hybrid method for novelty detection in time series based on states transitions and swarm intelligence," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–8.

[164] L. Tarassenko, A. Nairac, N. Townsend, and P. Cowley, "Novelty detection in jet engines," in *IEE Colloquium on Condition Monitoring: Machinery, External Structures and Health (Ref. No. 1999/034)*, 1999, p. 4/1-4/5.

[165] A. X. Carvalho and M. A. Tanner, "Modelling nonlinear count time series with local mixtures of Poisson autoregressions," *Comput. Stat. Data Anal.*, vol. 51, no. 11, pp. 5266–5294, 2007.

[166] M. Svensén and C. M. Bishop, "Robust Bayesian mixture modelling," *Neurocomputing*, vol. 64, pp. 235–252, 2005.

[167] M. K. Albertini and R. F. de Mello, "A Self-organizing Neural Network for Detecting Novelties," in *Proceedings of the 2007 ACM Symposium on Applied Computing*, 2007, pp. 462–466.

[168] M. Z. Hayat and M. R. Hashemi, "A DCT based approach for detecting novelty and concept drift in data streams," in *2010 International Conference of Soft Computing and Pattern Recognition*, 2010, pp. 373–378.

[169] M. M. Masud, T. M. Al-Khateeb, L. Khan, C. Aggarwal, J. Gao, J. Han, and B. Thuraisingham, “Detecting Recurring and Novel Classes in Concept-Drifting Data Streams,” in *2011 IEEE 11th International Conference on Data Mining*, 2011, pp. 1176–1181.

[170] E. R. de Faria, A. C. de Leon Ferreira Carvalho, and J. Gama, “MINAS: multiclass learning algorithm for novelty detection in data streams,” *Data Min. Knowl. Discov.*, vol. 30, no. 3, pp. 640–680, May 2016.

[171] I. Katakis, G. Tsoumakas, and I. Vlahavas, “Tracking recurring contexts using ensemble classifiers: an application to email filtering,” *Knowl. Inf. Syst.*, vol. 22, no. 3, pp. 371–391, Mar. 2010.

[172] T. S. Sethi, M. Kantardzic, and H. Hu, “A grid density based framework for classifying streaming data in the presence of concept drift,” *J. Intell. Inf. Syst.*, vol. 46, no. 1, pp. 179–211, Feb. 2016.

[173] M. Askarian, R. Benítez, M. Graells, and R. Zarghami, “Data-based fault detection in chemical processes : Managing records with operator intervention and uncertain labels,” *Expert Syst. Appl.*, vol. 63, pp. 35–48, 2016.

[174] F. O. de França and A. L. V Coelho, “A biclustering approach for classification with mislabeled data,” *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5065–5075, 2015.

[175] C. Zhang, D. Li, J. Yang, and A. Yong, “Human cognitive paradigm and its application in semi-supervised learning,” *Opt. - Int. J. Light Electron Opt.*, vol. 125, no. 3, pp. 1178–1184, 2014.

[176] B. Frenay and M. Verleysen, “Classification in the Presence of Label Noise: A Survey,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 5, pp. 845–869, 2014.

[177] M. J. Piovoso, S. Member, K. A. Kosanovich, and J. P. Yuk, “Process Data Chemometrics,” in *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT*, 1992, vol. 41, no. 2, pp. 262–268.

[178] A. Raich and A. Çinar, “Diagnosis of process disturbances by statistical distance and angle measures,” *Comput. Chem. Eng.*, vol. 21, no. 6, pp. 661–673, 1997.

[179] P. M. G. Jr, S. G. T. de C. Santos, R. S. M. Barros, and D. C. L. Vieira, “A comparative study on concept drift detectors,” *Expert Syst. Appl.*, vol. 41, pp. 8144–8156, 2014.

[180] Š. Raudys, “How good are support vector machines?,” *Neural Networks*, vol. 13, no. 1, pp. 17–19, Jan. 2000.

[181] M. Ardakani, G. Escudero, M. Graells, and A. Espuña, “Incremental Learning Fault Detection Algorithm Based on Hyperplane-Distance,” in *Computer Aided Chemical Engineering*, vol. 38, 2016, pp. 1105–1110.