Machine Learning for Early Detection of Distillation Column Flooding

by

© **Opeoluwa Adebayo**

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Engineering.

Department of Process Engineering

Memorial University

February 2025

St. John's, Newfoundland and Labrador, Canada

Abstract

Distillation column significantly affects the overall energy efficiency of a process plant. Poor performance of a column can result from faults, such as reflux failure, change in tray efficiency, change in feed temperature, etc. Flooding is one of the severe consequences of the faults attributed to a distillation column. During flooding, products go off-specification, and there is a tendency for a complete shutdown of the production process. In recent years, machine learning (ML) methods have been widely employed in process engineering for their ability to discover important patterns in data. One of the applications of ML is in predicting distillation column flooding. The supervised ML methods can predict flooding by forecasting the pressure drop across the column. The challenges of applying supervised ML methods for predicting flooding in distillation columns include a lack of large volumes of flooding data, the potential for overfitting, and long training time in some cases. A large amount of flooding data combined with normal operation data is needed to train the supervised ML algorithms for flooding detection. Therefore, it is important to identify flooding data sets from the operational data. However, flooding data sets are rare compared to normal data sets, which leads to an imbalanced data set. In this research, we address the data scarcity issue surrounding the application of supervised ML for flooding prediction by utilizing time-series generative adversarial networks, a framework that uses deep learning algorithms to generate synthetic data by preserving the temporal order in the original data. Additional flooding data sets are generated using this framework. Supervised ML algorithms are trained and tested to forecast the pressure drop of the column. Classification of the column data (i.e., flooding or not flooding) is done using clustering. This method is compared with predicting flooding using popular unsupervised ML methods such as principal component analysis (PCA) and autoencoders; these are unaffected by the data imbalance. Results show that by applying supervised ML algorithms to the sensor data of the distillation column, flooding conditions can be detected 19 minutes in advance and up to 60 minutes before it fully develops. This outperforms the PCA and autoencoders, which are popular unsupervised ML methods.

To my dear parents, for their constant love and support. Thanks for making me who I am today.

Acknowledgements

Firstly, I would like to thank the Almighty God for his guidance and blessing, which made it possible for me to complete the research successfully.

I would like to express my deep and sincere gratitude to my research supervisors, Dr. Syed Imtiaz and Dr. Salim Ahmed, for giving me the opportunity to do this research and providing invaluable guidance throughout my program at the University. I would also like to thank them for their financial support throughout my program at the University. It was never possible to succeed without their help and support throughout this research program.

I appreciate the advice and knowledgeable contributions of my fellow students in process systems engineering research group. I learned a lot from them during my research program.

I would also like to express my sincere gratitude to my spouse, Oluwatosin Adebayo, for her love, prayers, and mental support throughout my research program.

Finally, I thank my parents for their love, prayers, support, and sacrifices towards my education.

Contents

List of Tables

List of Figures

List of abbreviations

DFR	Distillate flowrate
RF	Reflux flowrate
SNR	Signal-to-noise ratio
TimeGAN	Time-series generative adversarial networks
VFR	Vapor flowrate.

Chapter 1

Introduction

1.1 Background

Distillation column is an important unit operation in chemical and process industries, playing a pivotal role in separating complex mixtures into their individual components or fractions [1]. According to Nicholas Cheremisinoff [1], the distillation process is very energy intensive and can contribute to more than 50% of plant operating costs. Thus, there is great interest in operating distillation columns as efficiently and reliably as possible [2].

Poor performance of a distillation has a high effect on the overall performance of a process plant. This poor performance is a result of process faults such as reflux failure, change in tray efficiency, change in feed temperature, etc. These faults affect the column purity, temperature, and pressure profiles of a distillation column [3].

A fault is an anomaly that causes the system to deviate from the normal operating conditions [4]. Faults can cause a loss of efficiency during operation and, in the worst cases, a loss of asset availability. Flooding is one of the severe consequences of the faults attributed to a distillation process. Flooding is an abnormal situation in which the distillation column stops generating a separation due to build-up of liquid in the column, consequently leading to the products going off specification [5]. Once fully developed, flooding causes a significant increase in the differential pressure across the distillation column as a result of liquid accumulation and a reduction in the separation performance of the distillation column. When unchecked, flooding can disrupt the entire production process of a chemical plant by causing limited production operation or complete shutdown of the production process [6].

To avoid the degradation in the performance of a distillation column, there is a need to anticipate and predict flooding so that corrective measures can be taken to avoid its full development. This can be done by providing early warnings of flooding risk to process operators some minutes before the completely developed flooding event.

Machine Learning (ML) algorithms are gaining ground in process engineering applications due to the large amount of industry-generated data, and these data are available from physical sensors [7]. Data collected from the physical sensors can be combined and processed into meaningful data through machine learning algorithms. The processed data can be used for training data-driven models to capture the process conditions [8]. Application of machine learning algorithms to process industry include but are not limited to online prediction, process monitoring, and process fault detection [9]. Examples of these applications are but are not limited to support vector machine (SVM) based on experimental data to predict flooding in packed-columns [10], data-based gaussian process combined with empirical flooding equations to calculate flooding curves [11], a combination of clustering and change point algorithms using refinery data for real-time fault detection [12], usage of bayesian recurrent neural networks (RNN) to detect foaming in amine distillation column [13] and application of transfer learning using a neural network (NN) trained with data from a first-principle dynamic simulator to detect anomalies on industrial data [14].

1.2 Problem Statement

Flooding in distillation columns is a critical operational issue that can impair separation efficiency and safety. Most traditional methods for detecting flooding often lack the precision and adaptability required for real-time monitoring. These traditional detection methods are also insufficient in dynamic environments where process conditions frequently change. As a result, there is a need for more advanced and reliable techniques, such as ML to predict and mitigate the risk of flooding in distillation columns.

A significant challenge in applying supervised ML methods to flooding detection is the data imbalance caused by insufficient flooding data, which reduces the model's accuracy and reliability. Without a balanced dataset of normal operational data and flooding data, supervised ML models struggle to identify flooding conditions effectively, limiting their utility in industrial fault detection. This research addresses this gap by generating synthetic flooding data to balance the dataset, enabling the practical application of supervised learning techniques for reliable flooding detection in distillation columns.

1.3 Motivation

Detecting flooding in distillation columns can significantly enhance operational safety and efficiency. However, real-world flooding events are rare and challenging to simulate repeatedly under controlled conditions, leading to limited data availability for training ML models. This data scarcity creates a bottleneck for adopting supervised ML techniques, as unbalanced datasets often result in biased models with poor generalizability. By generating synthetic flooding data to counteract this imbalance, this research aims to open new avenues for deploying machine learning in industrial process monitoring, contributing to safer and more efficient distillation operations.

1.4 Research Objectives

One of the significant effects of flooding in a distillation column is an increase in the pressure drop between the distillation column stages due to the accumulation of liquid in the column. This research aims to detect flooding early in a distillation column by monitoring the pressure drop through supervised learning methods. Using supervised learning methods for flooding detection requires sufficient flooding data and normal operational data so that the supervised learning methods can accurately capture the dynamics of the distillation column during normal operation and before flooding occurs in the column. However, the lack of sufficient flooding regime data leads to a case of data imbalance between normal operation data and flooding data, with flooding data being the minority, thereby making supervised learning methods unsuitable for such tasks. However, unsupervised learning methods such as principal component analysis (PCA) and Autoencoders are not affected by data imbalance problems caused by insufficient flooding data. Hence, they are more suitable for this task. This work focuses on detecting flooding in a distillation column in case of insufficient flooding regime data. The following are the objectives of this research:

- detect flooding in a distillation column using data-driven approaches such as PCA and Autoencoders. Such methods do not suffer from data imbalance caused by insufficient flooding regime data,
- overcome the data imbalance problem by generating synthetic flooding regime data,
- develop supervised learning methods for early flooding detection in a distillation column through pressure drop forecast,
- develop additional variables that can be used as indicators to better detect flooding in a distillation column.

1.5 Thesis Structure

The structure of this thesis is shown in Fig.1.1. This thesis consists of five chapters. The first chapter presents the background of this study to justify the need to conduct this research. It also summarizes what this research aims to achieve by giving the objectives of the study. Also, it describes the software used to carry out this study. The rest of the thesis is organized as follows: Chapter 2 provides an extensive literature review of fault detection and diagnosis in process systems. This includes various methods of detecting and diagnosing faults in process systems. An extensive review of flooding in a distillation column is also presented in this chapter. This includes the concept of flooding, the causes of flooding, and various ways of detecting flooding in a distillation column. An overview of machine learning is also presented in this chapter. This includes an overview of different machine learning methods and the mathematical formulation of the machine learning methods used in this work. Chapter 3 presents the research methodology used in this work. This includes approaches used in this research to detect flooding in a distillation column and the appropriate mathematical formulations associated with the approaches. The case study considered in this research and the results obtained are presented in Chapter 4. Finally, Chapter 5 concludes this thesis by summarizing the findings and highlighting the contributions of this research. Recommendations for future works are also given in this last chapter.

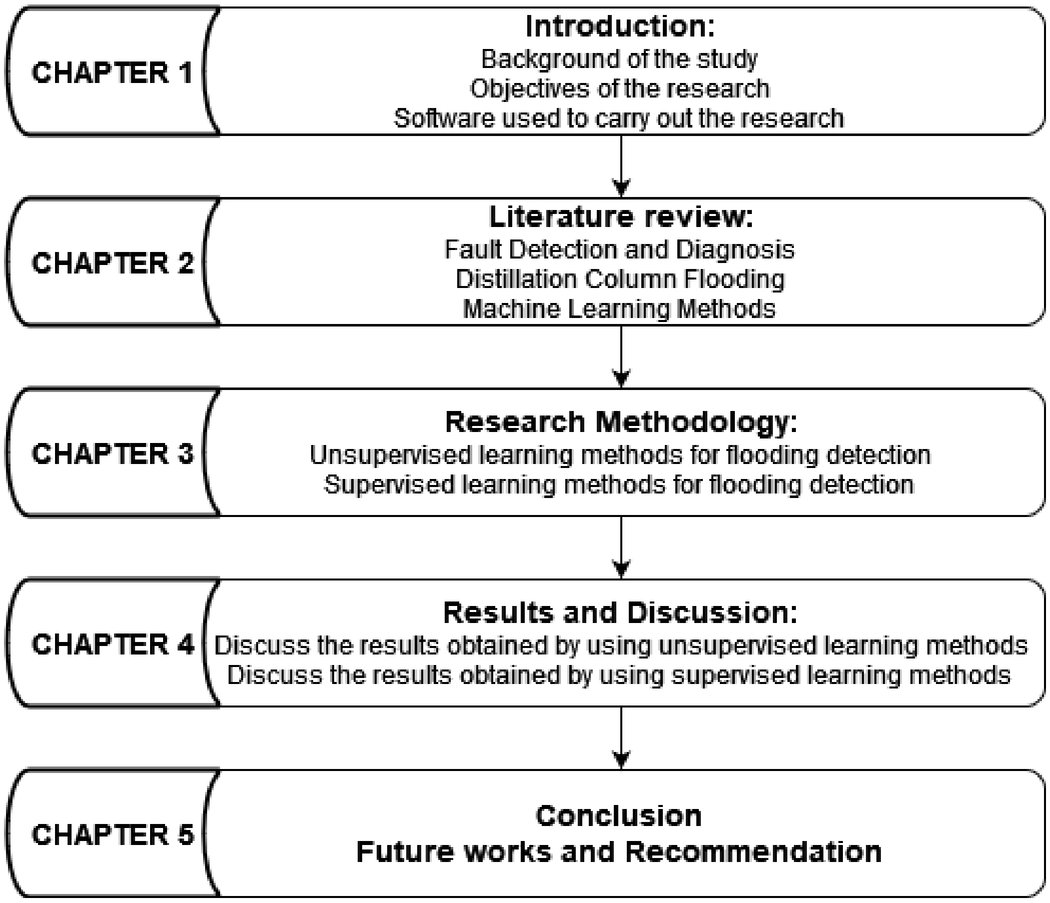

CHAPTER 1
Introduction:
Background of the study
Objectives of the research
Software used to carry out the research
CHAPTER 2
Literature review:
Fault Detection and Diagnosis
Distillation Column Flooding
Machine Learning Methods
CHAPTER 3
Research Methodology:
Unsupervised learning methods for flooding detection
Supervised learning methods for flooding detection
CHAPTER 4
Results and Discussion:
Discuss the results obtained by using unsupervised learning methods
Discuss the results obtained by using supervised learning methods
CHAPTER 5
Conclusion
Future works and Recommendation

Figure 1.1: Summary of the structure of the thesis.

1.6 Software Used

The algorithms used in this research and their applications have been demonstrated using real-life and generated data using Python [15], a high-level, interpreted, open-source programming language. Multiple packages and libraries have been incorporated during the analysis, such as numpy [16] to implement multidimensional arrays and matrices, along with mathematical functions to operate on them. Pandas [17], for data manipulation and analysis through the provision of data structures and operations. Scitkit-learn [18] for predictive data analysis using efficient tools. Scipy [19], which contains modules for optimization, linear algebra, integration, and signal processing. The visualization of the results of this study is presented using matplotlib [20], a comprehensive library for creating static, animated, and interactive visualizations.

Chapter 2

Literature Review

This chapter aims to provide a comprehensive overview of existing literature related to this study. This chapter presents an overview of process fault detection and diagnosis. Followed by an overview of flooding in a distillation column, its causes, and various ways of detection. Subsequently, an overview of machine learning methods is presented. Discussion about different machine learning methods used in this research and their mathematical formulations are also presented. Finally, this chapter also introduces a time-series generative adversarial network. It is a deep-learning framework for generating synthetic time series data by preserving the temporal order of the data. The mathematical formulations of this framework are also presented.

2.1 Fault Detection & Diagnosis

Modern control systems have grown exceedingly complicated as a result of the integration of many functions and components to meet advanced performance requirements [21]. Systems become prone to faults in their daily operations due to this complexity. The reliance on human operators to handle such unexpected occurrences and emergencies is getting increasingly problematic owing to various variables [22]. It is problematic since the diagnostic activity encompasses multiple malfunctions such as process unit failures, deterioration, parameter drifts, etc. The scale and complexity of contemporary industrial plants complicate matters even further. For example, the number of process variables in a large processing plant may exceed a few thousand variables collected repeatedly every few seconds [23]. Also, catastrophic accidents can occur due to human operators' poor decisions based on inadequate information from sensor failures or bias. 76.1% of accidents were caused by human error [24]. Therefore, fault detection and diagnostics (FDD) tools are important.

A typical process monitoring and management system comprises fault detection, fault identification, fault diagnosis, and process recovery. Fault detection determines whether there is an unpermitted deviation of at least one characteristic property or parameter of the process from the standard condition (i.e., fault) and the time of detection of the fault. Whereas fault identification determines a fault's size and time-variant behavior, followed by the kind of fault, its location, and time of detection. Also, fault diagnosis is used to isolate the fault and extract additional knowledge like fault type, size, and root cause. Process recovery is related to removing the effect of the fault and returning the process to normal functioning [25]. Fault identification is often considered partly under detection and partly under fault diagnosis, and the entire process is referred to commonly as FDD as illustrated in Fig.2.1.

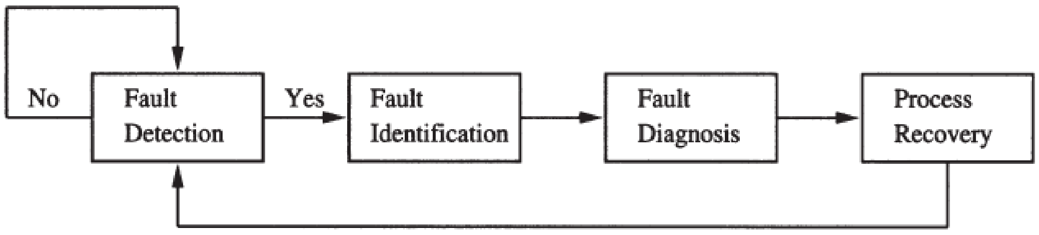


Figure 2.1: Schematic illustration of a process monitoring loop.

The fault detection step indicates whether a process variable is in the normal operating range. Early detection of an operational deviation increases the ability to diagnose faults before they reach a critical stage. This mitigates the risk and ensures the safety of an operation. The same applies to flooding in a distillation column. Fault diagnostic focuses on classifying the faults and inferring the root cause. A quick and correct diagnosis makes it easier for operators to get to the root of the problem, thus eliminating any cascading events that may lead to accidents. Many FDD methods have been developed throughout the last decades. Each technique has advantages and disadvantages that are dictated by factors such as domain-specific knowledge required, historical data availability, reliability, generality, and computing complexity [22].

2.2 Methods for Fault Detection & Diagnosis

Fault detection in process industries is a challenging task. It requires timely detection of anomalies, which can be a result of failures or malfunction of sensors or actuators, including changes in process variables within a system. Fault detection has been classified broadly into three major classes: model-based, data-based, and knowledge-based approaches [26].

2.2.1 Model-Based Methods

The model-based methods use a model of the system that mathematically describes the relationship among the various variables in the process. Deriving an accurate model of a complex industrial system can be difficult and time-consuming; this is a downside to the model-based approach.

2.2.1.1 Residual Generation

In model-based approaches, there is a need for the generation of residual signals from the mathematical model of the system as illustrated in Fig.2.2, which is then used as a fault indicator [27]. Residual is based on a deviation between measurements and model-equation-based computations.

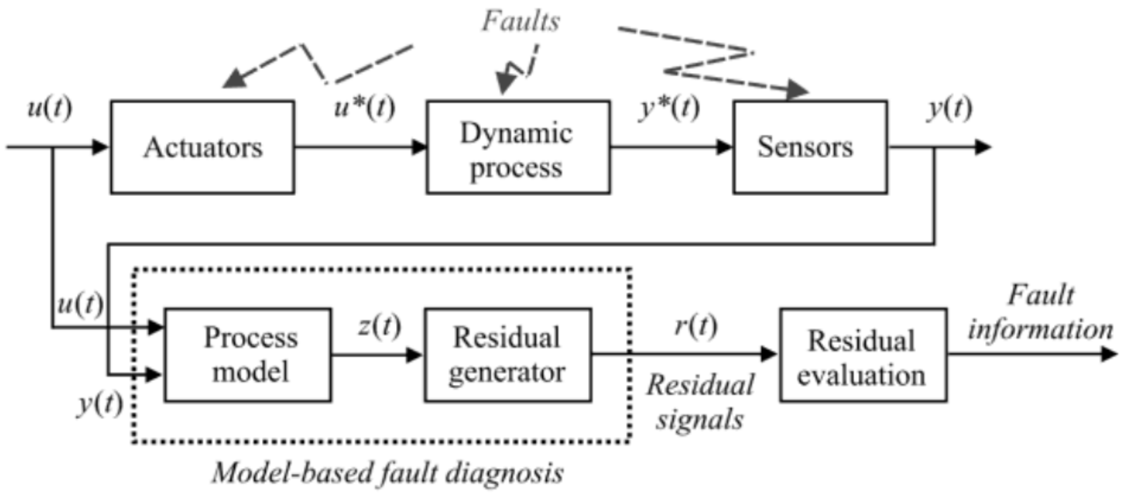


Figure 2.2: Schematic illustration of model-based approach [28].

Fig.2.2 represents a schematic illustration of this approach, consisting of two parts: residual generation and evaluation. The residual generator compares the system's data to the results of the process model and publishes the discrepancies as residuals. The residual evaluator gets the residuals and decides if the status of the process is faulty or normal [29].

The true relationship among process variables may be nonlinear. However, most model-based methods assume linear relationships among process variables. The plant model is a mathematical representation of the system. It is often developed based on first principles, for example, the conservation of mass, energy, and momentum. Running the model and the real system with the same input, the output should be the same, and the residual must be zero. In practice, however, the residual vector may always have a non-zero value due to noise or model-plant mismatch. A threshold is specified to indicate the faulty region. Different tools have been proposed for FDD using the constructed model to generate residual signals, including parameter estimation, observer-based, and parity-relations approaches [30]. Generally, the faults are either additive faults or multiplicative faults [31]. The additive faults, such as sensors or actuators bias, can be modeled as follows:

$$x(t+1) = A\mathbf{x}(t) + B\mathbf{u}(t) + B_f\mathbf{f}(t) + B_d\mathbf{d}(t) + B_n\mathbf{n}(t) \tag{2.1}$$

$$y(t) = C\mathbf{x}(t) + D\mathbf{u}(t) + D_f\mathbf{f}(t) + D_d\mathbf{d}(t) + D_n\mathbf{n}(t) \tag{2.2}$$

Where A, B, C, and D are the system state space matrices associated with the state matrix $\mathbf{x}(t)$ and output matrix $\mathbf{u}(t)$. f is associated with the fault matrices, d for disturbance matrices and n for noise matrices. On the other hand, other faults are represented by multiplicative faults, and they are modeled as follows:

$$x(t+1) = (A + \Delta A)\mathbf{x}(t) + (B + \Delta B)\mathbf{u}(t) \tag{2.3}$$

$$y(t) = (C + \Delta C)\mathbf{x}(t) + (D + \Delta D)\mathbf{u}(t) \tag{2.4}$$

2.2.1.2 Parameter Estimation

Process faults usually result in changes in the model parameters or the state variables. The parameter estimation approach is acceptable if the process failures are related to changes in model parameters. This means that it is appropriate for multiplicative fault scenarios. The residuals are calculated using the nominal model parameters and estimated model parameters. The parameters are estimated using standard techniques [32–34], and developed based on a recursive concept to reduce the computational burden. If the changes in physical parameters are greater than those detected in training data, faults are indicated. Least squares methods provide a powerful tool by monitoring the parameter estimates online [35].

2.2.1.3 Observer-based Methods

The Observer-based technique is appropriate if the faults are related to changes in actuators or sensors, i.e., it is particularly useful for identifying and isolating additive faults. The observer-based method reconstructs the output of the system from the measurements or a subset of the measurements with the aid of observers. The difference between the measured outputs and the estimated outputs is used as the vector of residuals. Reconstruction of the unmeasured states from the measurable input and output can be done using Luenberger observer or Kalman Filter [36]. Kalman filters, or observers, are widely used for state estimation. The main concern of the observer-based technique is the generation of a set of residuals that detect and uniquely identify different faults. These residuals should be robust in the sense that the decisions are not corrupted by such unknown inputs as unstructured uncertainties like process and measurement noise and model uncertainties [27]. Hence, the residuals are determined as the difference between the estimated and measured plant output.

2.2.1.4 Parity Relation Methods

The parity relation methods are popular for residual generation in the field of model-based FDD. Parity equations are rearranged and usually transformed variants of the input-output or state-space models of the plant. The essence is to check the parity (consistency) of the plant models with sensor outputs (measurements) and known process inputs. In the parity space, residual generation, the dynamics of the residual signals regarding the faults and unknown inputs are presented in the form of algebraic equations. Hence, most of the problem solutions are achieved in the framework of linear algebra [37]. They are used to estimate the residuals of a linear regression model, and they have been shown to produce unbiased estimates with minimal variance [38].

In the open loop process, the parity-relation vector is formed from a linear combination of sensor outputs and applied inputs, and the values are interpreted from measures using the observers. The residuals should be zero in normal operating conditions where the system shows no fault. However, The existence of noise and model uncertainty prohibit the residual from becoming zero. Many approaches have been developed to address the parity relations for FDD, such as dynamic parity relations [39], building parity relations using the state-space model [40], etc. One of the key assumptions of this technique is that the model is linear. As a result, this technique becomes less suited for monitoring batch operations since operating conditions vary constantly, and non-linearity is common in large-scale production plants.

The major advantage of model-based methods is clearly capturing the process's dynamics. In addition, costly hardware redundancy is eliminated. However, these approaches rely on the accuracy of the model. Furthermore, working with non-linear systems makes analysis and modeling more complex. Moreover, the process of creating a model may be time- and resource-consuming. Hence, the model-based category becomes less suitable in large-scale process monitoring and existing systems [27]. In the absence of an explicit model of a system and if measurement signals are the only resources, the data-driven implicit models are suitable [26].

2.2.2 Data-Based Methods

Data-driven techniques are widely applied in the process industry for process monitoring and diagnosis purposes [41]. Data-driven methods involve data analysis and machine learning techniques to identify system anomalies, defects, or faults. These methods do not require a thorough knowledge of the process dynamics. Hence, they leverage historical and real-time data to detect deviations from normal operating conditions, often providing early warnings of potential issues. Process data acquired under normal and abnormal operating circumstances is utilized for fault detection and diagnosis. As a result, the efficacy and accuracy of these methodologies are determined by the availability and quality of the supplied data. The advancements in control and data acquisition systems allow for the collection of vast amounts of process data. Data is extracted, loaded, and transformed for process monitoring, fault detection, and further analysis. These approaches are classified as qualitative, such as expert systems and trend analysis, and quantitative, which includes statistical methods, artificial neural networks (ANN), and support vector machine (SVM) [27]. The statistical methods depend on statistical parameters such as the mean and standard deviation of observations. There are two types of methods: univariate and multivariate.

2.2.2.1 Univariate Methods

Univariate methods are applicable for monitoring a single process variable. The Shewhart Chart ($\bar{x}$ Chart) is the most popular approach used for single variable control [42]. The upper and lower control limits are estimated as follows:

$$UCL = \mu + T\sigma$$

$$LCL = \mu - T\sigma$$

where μ and σ are the mean and the standard deviation of the process variable during the normal operation. A threshold of T is used to define the normal operating range, as it is usually set to 3. Exceeding these limits indicates a process fault. Other control charts for a single process variable, such as Exponential Moving Average (EWMA) [43] and Cumulative Sum (CUSUM) [44], also employ several time frames to increase fault detection efficiency and minimize false alarm and missed alert rates. CUSUM accumulates deviations from a target value, enhancing sensitivity to small shifts in process parameters while EWMA applies weights to data points, emphasizing recent observations, useful for detecting small and gradual changes. While useful in many scenarios, univariate statistical methods have several disadvantages, particularly when applied to complex systems with multiple interrelated variables. A single control chart is required to monitor each variable, neglecting the correlation effect between process variables. Therefore, they are inappropriate for monitoring the modern complex and dynamic processes. They do, however, increase complexity and need additional processing time. Also, they require proper tuning for the hyper-parameter. Univariate methods are preferred because they are relatively easy to implement. However, the methods cannot account for multicollinearity, where two or more variables are highly correlated. They can become cumbersome and less effective as the number of variables increases. In systems with many variables, monitoring each individually becomes impractical. Multivariate methods provide a more scalable solution by summarizing the information from all variables into a few components or indices.

2.2.2.2 Multivariate Methods

Multivariate approaches are essential for monitoring and diagnosing systems where multiple interrelated variables must be considered simultaneously. These methods analyze the relationships between variables, capturing complex patterns and interactions that univariate methods might miss. They are more resistant to false alarms than univariate methods. They also display the process data in a reduced dimensional space. Both of those offer reliable fault detection and process monitoring at lower computing costs. Examples of these methods are principal component analysis (PCA) [45, 46], partial least squares (PLS) [21, 47, 48], Fisher discriminant analysis (FDA) [49], and independent component analysis (ICA) [50]. The popularity of these algorithms is based on their ease of implementation and capability of tackling some problems, which include dimensionality reduction of data and extraction of critical features from the data. PCA reduces the dimensionality of data by transforming the original variables into a smaller set of uncorrelated variables called principal components. These components capture most of the variance in the data. Detection indices such as T^2-statistic and Q-statistic are used to monitor the variability captured by the principal components and in the residuals, respectively. PLS is a regression technique that models the relationship between input and output variables by extracting latent variables that explain both sets. ICA separates a multivariate signal into additive, statistically independent components. It's particularly useful when the underlying source signals are independent but not necessarily uncorrelated. FDA, also known as Linear Discriminant Analysis (LDA), is a statistical method for pattern recognition, classification, and dimensionality reduction. It is particularly effective in scenarios where there are two or more classes, and the goal is to separate them by finding a linear combination of features that best separates the classes.

Although data-driven process monitoring systems are simple to use and effective in detecting faults early, the diagnosis is imperfect. This is due to the difficulty of interpreting measured variable contributions in a larger process with multiple variables. A system may be unable to detect a small magnitude fault [51]. Also, a smearing effect due to matrix multiplication to calculate variables' contribution can lead to ambiguity in FDD [52]. This adds more complexity to the operator to accurately detect the cause of the process fault. Additionally, since data-driven methods typically only use one data source, these methods lose a lot of information derived from various sources, like the event log or expert knowledge. Therefore, to overcome these limitations, researchers focus more on feature extraction, knowledge integration, and methods integration. Also, data-driven methods rely heavily on the availability and quality of historical and real-time data. Poor quality, incomplete, or noisy data can significantly reduce the accuracy and reliability of fault detection models. Missing data or errors in data collection can lead to incorrect fault detection, either missing actual faults (false negatives) or incorrectly identifying normal behavior as faults (false positives). While data-driven methods aim to reduce the need for extensive domain knowledge, they still often require significant expertise for data preprocessing, feature selection, and model tuning [53].

2.2.3 Knowledge-Based Methods

Knowledge-based methods for fault detection leverage human expertise, rules, and domain-specific knowledge to identify faults and anomalies in systems. These methods are often implemented using expert systems, rule-based systems, and model-based approaches. A comprehensive description of the process, expert knowledge, a causal model, or fault-symptom scenarios may all be used to create qualitative models for knowledge-based approaches [21]. When a thorough mathematical model is unavailable, and a system has relatively few inputs, outputs, and states, these approaches are appropriate [21]. According to [54], the most common knowledge-based methods in FDD are qualitative simulation (QS), expert systems, fault tree analysis (FTA), signed digraphs (SDG), and Bayesian networks (BN). Qualitative simulation is a method used for fault detection that focuses on understanding the behavior of a system through qualitative descriptions rather than precise numerical data. This approach is particularly useful when quantitative models are difficult to obtain, or the system is too complex for detailed quantitative analysis. They can accurately anticipate how the system would behave under normal conditions and when there are various faults, which can be vital diagnostic information. Steady-state qualitative simulation (QSIM) [55] is a common method of QS. QSIM interprets a system's dynamic behavior using qualitative differential equations. By value and direction, it expresses the qualitative state of a variable. Value refers to an ordered collection of remarkable values that generally describe variables' values, whereas direction denotes a change in direction [55]. This technique has certain drawbacks, such as intrinsic ambiguity and the absence of temporal information. [56] introduced a fuzzy QSIM algorithm that enabled the inclusion of more quantitative information, such as the rate-of-change of variables and the relative strengths of qualitative relationships.

2.2.3.1 Expert System

An expert system is an organized knowledge system that imitates a human expert to address issues in a particular field [57]. Expert systems use a set of rules and facts to mimic the decision-making abilities of human experts. Over time, plant operators gain theoretical and practical knowledge to determine the origin of a probable issue and recommend appropriate repair procedures. An expert system can be used to automate this procedure. Four elements are often found in an expert system: knowledge base, inference engine, knowledge management, and user interface [58]. Rule-based systems are a type of expert system that uses a set of predetermined rules to detect faults. These rules are typically derived from domain knowledge and expert insights. A rule-based expert system for fault detection in chemical processes was proposed by [59]. [60] developed an expert system for FDD in a power station. An expert system for a refinery's cracking unit was implemented with the inclusion of static rules, time-varying rules, and bidirectional heuristics by [58]. The aforementioned expert systems used rule-based diagnostic knowledge, which is simple to create and does not need intricate quantitative domain knowledge. However, it could have drawbacks, including limited resolution, fusion explosion, and the inability to capture knowledge effectively about processes that change in space and time. Also, rule management can become complex as the number of rules grows.

2.2.3.2 Fault Tree Analysis

Fault tree analysis (FTA) is another popular method for fault detection and reliability assessment. FTA is a top-down, deductive failure analysis technique that uses a tree-like model of the various logical relationships between system failures and their causes. Basic and intermediate events are propagated graphically and hierarchically to the top event representing the hazard. Various issues, such as equipment breakdown, component contamination, or human factors, are considered the basic events. An aberrant symptom is the intermediate events. Different logic gates (e.g., AND, OR, and XOR) are used to conduct the propagation, providing a more adaptable representation of causal information. A general fault tree analysis consists of the following four steps: (i) system definition, (ii) fault tree construction, (iii) qualitative evaluation, and (iv) quantitative evaluation [61]. Prior to the construction of the fault tree, a complete understanding of the system is required. In fact, a system description is also a part of the analysis documentation[61]. The fault tree construction usually starts by asking questions such as what could cause a top-level event. In answering this question, other events connected by logic nodes are generated. The tree is expanded in this manner till one encounters events (primary events) that need not be developed further [62]. After constructing the fault tree, the next step in the analysis is evaluating the fault tree. A top-down analysis may be used to determine the underlying cause of an undesirable occurrence, whereas a down-to-up analysis can be used to determine the effects of a fundamental event. Qualitative evaluation is concerned with the development of minimal cut sets, defined as a collection of primary failures, all of which are necessary and sufficient to cause the system failure by the minimal cut-set in question [27]. Converting the fault tree into the smallest cut sets and ranking their consistency is a typical strategy. The set of primary events required for the occurrence of the top event is known as the minimal cut set [63]. A quantitative version of the FTA may be created if the historical statistics data is provided [64]. Although this approach is simple to develop, it requires a significant amount of effort and is prone to mistakes.

2.2.3.3 Signed Directed Graphs

Given the structural and functional aspects of a process, the behavior of the process can be derived. Simulation is concerned with this form of derivation. Diagnosis is the inverse of simulation. Diagnosis is concerned with deducing structure from the behavior. This kind of deduction needs reasoning about the cause-and-effect relationships in the process. Cause-effect relations or models can be represented in the form of signed directed graphs (SDGs). SGD is another popular qualitative technique for process fault diagnostics. A digraph is a graph with directed arcs between the nodes, while an SDG is a graph in which the directed arcs have a positive or negative sign attached to them [27]. They employ nodes and directed links to describe the process's variables, as well as the causal linkages between them. Each node has a sign to describe its status. The direction of the link represents the cause-effect relation. The tail is connected to the cause, while the tip points toward the effect. The massive knowledge represented in SDG is obtained by deep mathematical models, including differential and algebraic equations [65]. The use of differential equations to develop SDGs for fault diagnosis was proposed by [66]. SDGs are an extremely effective visual representation of qualitative models. They are the most extensively used method for causal knowledge-based process fault identification [67]. Knowledge-based models can identify the underlying cause of a process issue immediately after it has been identified.

2.2.3.4 Bayesian Network

Bayesian Network (BN) is another widely used method for representing knowledge and reasoning. It has both qualitative and quantitative components. The qualitative section is a directed acyclic graph (DAG), with nodes representing random variables and links representing causation relationships between them. A link connects a parent node to a child node. The term "leaf node" describes a node without any children, while the term "root node" describes a node without any children. Root nodes are assigned marginal probabilities, while the rest of the network is assigned conditional probabilities. The Bayes theorem, shown below, states that the posterior probabilities for unobservable nodes can be estimated by propagating observable observations throughout the network once they become available.

$$P(X_E|X_O) = \frac{P(X_1, \cdots, X_n)}{\sum_O P(X_1, \cdots, X_n)} \tag{2.5}$$

where X_O denotes the observed nodes, while X_E denotes the nodes that need to be estimated. The main advantage of this approach is the ability to diagnose faults accurately even though various uncertainties are presented [68]. Also, its robustness is not affected by the quality or availability of data. It enables the incorporation of prior process information in qualitative and quantitative forms, such as causal relationships and probabilities. BN for fault detection and diagnosis was proposed earlier by [69]. However, the time dependency essential for describing how aberrant events originate and propagate was not considered. The dynamic BN was used to overcome this limitation. [70] proposed a dynamic BN that incorporates the hidden Markov Model for fault diagnosis. A dynamic Bayesian network (DBN) approach has been developed for systems with missing data [71]. It is important to note that the majority of the aforementioned algorithms call on conditional probability distributions of node states under relevant fault circumstances, which are difficult to provide. [72] created new indices to assess the abnormality probability at each node and anticipated that nodes in actual fault propagation paths would have a greater faulty likelihood. As a result, the fault propagation path may be found by looking for nodes across the network that have a high risk of being anomalous.

2.3 Distillation Column Flooding

2.3.1 Distillation Column Description

In a typical distillation column, the feed stream, consisting of a mixture of components to be separated, is fed into a tall cylindrical vessel standing vertically. This vessel is popularly referred to as the "distillation column." Inside the vessel are structures designed to cause intimate radial mixing, i.e., mixing at any given vertical level and the contact area between a stream of vapor flowing up and a stream of liquid flowing down. These structures can take the form of trays or stages, or they can be packed. The function of these internal structures is to make possible vapor/liquid contacting and mass transfer [2].

Fig.2.3 gives the illustration of a distillation column. The part of the distillation column above the feed point is commonly called the "Enriching Section" or the "Rectification Section." The part of the distillation column below the feed point is known as the "Stripping Section." Liquid flows down the column, exits the bottom of the stripping section, and flows into the reboiler. The reboiler is a special type of heat exchanger that uses steam or some other heat transfer fluid to heat the liquid in the reboiler to its boiling point. The vapor generated by this boiling liquid exits the reboiler and is fed back into the stripping section of the column. Excess liquid in the reboiler overflows a weir and exits the process as the "Bottoms Product," sometimes referred to as the "Bottoms." The vapor from the reboiler flows up the column, countercurrent to the liquid flowing down the column.

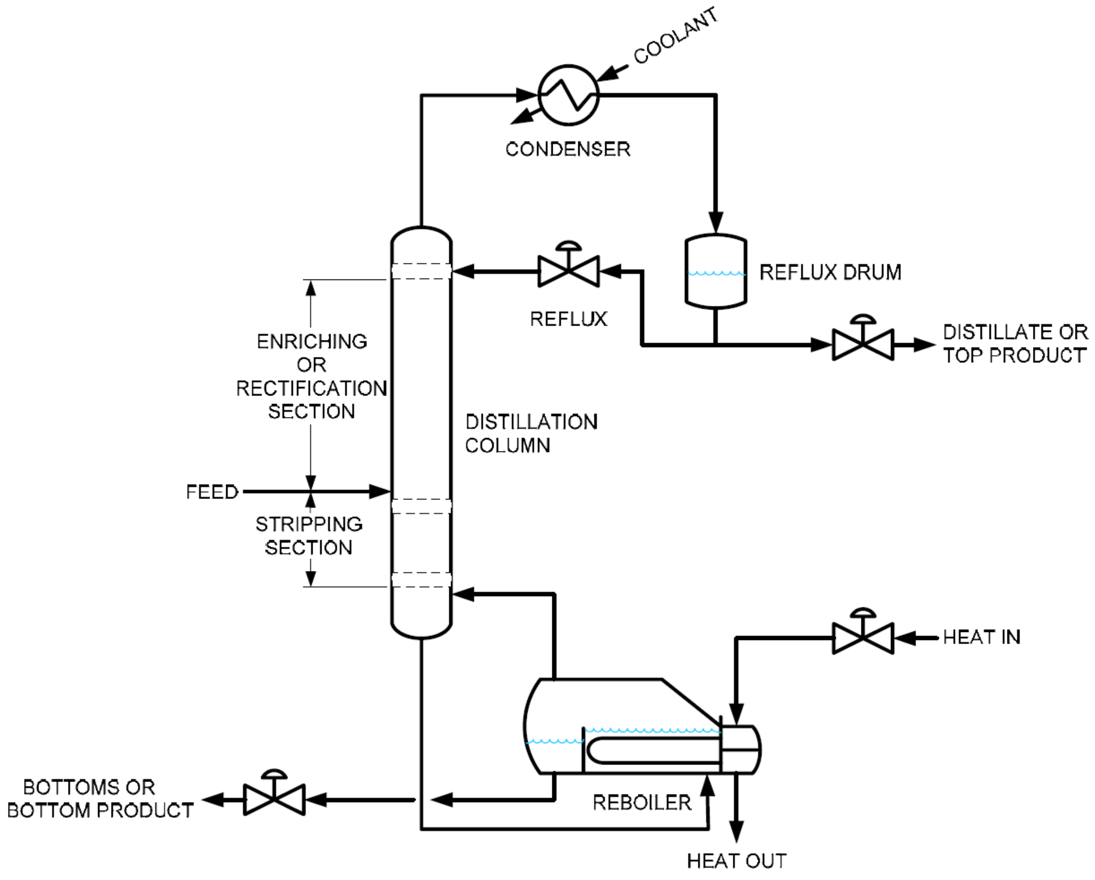


Figure 2.3: Schematic illustration of a distillation column [2].

The components in the feed stream are separated according to their relative boiling points. Components with a lower boiling point tend to become enriched in the vapor traveling up the column. Components with a higher boiling point tend to become enriched in the liquid traveling down the column. Eventually, the vapor enriched in low boiling components exits the rectification Section on the top of the distillation column. This vapor is condensed back to a liquid by cooling in a condenser heat exchanger. The condensed liquid is collected in the reflux drum. As described previously, a portion of the condensed liquid is fed back into the rectification section to become the liquid flowing down the column. The rest of the liquid exits the process as the “Top Product” or the “Distillate.”

2.3.2 Concept of Flooding

Flooding is a common problem that can occur in a distillation column. Liquid flows downward over the structured packing in a distillation column countercurrent to the upward-flowing vapor. The vapor must follow a tortuous path, but the void space in the packing is predominantly filled with vapor. The vapor is said to be in a continuous phase. The upward flow of the vapor exerts an aerodynamic drag on the falling liquid. This drag force acts in opposition to the force of gravity and slows the flow of the falling liquid [5]. When the relative flowrates of the vapor and liquid are such that the drag force is greater than or equal to the gravity force, then the liquid stops flowing down the column. This condition is called flooding. Flooding can begin at any vertical location in the column.

Flooding is generally defined as the condition of column inoperability due to excessive retention of liquid inside the column [73]. Flooding is an inherently unstable condition. Once the column reaches the flood condition, continued steady operation becomes impossible.

2.3.3 Symptoms of Flooding

Flooding is the most common capacity limitation in distillation. It is characterized by the accumulation of liquid in the column. This accumulation propagates upward from the lowest flooded region. Accumulating liquid backs up into the tray (or packed section) above, and so on, until the whole column fills with liquid or until an abrupt change in tray design or flow conditions (e.g., feed point) is reached. Flooding may or may not propagate above that point. Flooding can be recognized by one or more of the following symptoms:

1. Excessive column differential pressure
2. Sharp rise in column differential pressure
3. Reduction in bottom stream flow rate
4. Rapid rise in entrainment from column top tray
5. Loss of separation (as can be detected by temperature profile or product analysis)

The high-pressure drop observed during flooding is caused by the accumulation of liquid, which characterizes flooding [74, 75]. The high-pressure drop indicates liquid accumulation. The pressure drop may not significantly rise when the liquid accumulation is small. Typical scenarios include flooding near the top of the tower (only a few trays or a short packing length accumulates liquid), flooding in vacuum-packed towers (accumulation is channeled, and the vapor bypasses the accumulation region), and flooding at low liquid rates (slow liquid accumulation). Hence, a sharp rate of rise of pressure drop with vapor rate may be an even more sensitive flooding indicator than the magnitude of pressure drop [73]. As vapor loads are raised, so does the tray pressure drop. Upon flooding, the pressure drop rise accelerates due to liquid accumulation. In many cases, the pressure drop will rise once it starts, even when vapor loads are not further raised.

Reduction of bottom flow is a common indicator of flooding [76, 77]. Liquid accumulates in the column upon flooding, so less reaches the bottom. This can be seen by a fall in the bottom level. Most frequently, the bottom level is controlled by manipulating the bottom flow rate so the level stays constant, but the bottom flow rate will decline. While a reduction in bottom flow indicates flooding, many distillation columns may flood without a significant decline in bottom flow. For example, if flooding occurs in the rectifying section, while most of the feed is liquid, the bottom section may continue to operate normally without a significant decline in the bottom flow rate. Also, if the flood point is well above the bottom, there may be a significant delay from the onset of flooding to the time the bottom flow is significantly reduced, which makes accurate measurements of the flooding conditions difficult. Generally, a reduction of the bottom stream flow rate is a good indicator of flooding in columns that flood near the bottom and in columns that are relatively short [77], particularly if flooding occurs between the feed point and the bottom.

A rapid rise in entrainment is another common flooding indicator [76, 78, 79]. As liquid accumulates in the column, it builds up to the top and is entrained in the column overhead stream. In towers whose overhead stream goes to a knockout drum or to the bottom of another tower, this entrainment can be recognized as a buildup or rapid rise of a liquid level in the drum or bottom of the downstream column. In most distillation columns, the tower overhead goes to a condenser, and the condenser outlet stream continues to a reflux drum. The reflux drum usually has a level control that manipulates either the distillate or the reflux rate. When the drum level controls the distillate rate, the entrainment rise is often indicated as a significant increase in the distillate rate for no apparent reason. When the drum level controls the reflux rate, the entrainment is often indicated as a rise in reflux for no corresponding increase in boil-up rate and/or an increase in reflux flow rate that does not result in an increase in heat input required to maintain the same bottom column temperatures. The increased reflux is unable to descend down the tower due to the flooding near the top, so it entrains back into the overhead, returns as more additional reflux, and never reaches the bottom of the column. The reflux valve often opens widely due to the recirculation of entrainment around the tower overhead loop [80]. This indicator is particularly useful when the pressure drop rise is not sharp. However, this indicator may fail to indicate a stripping section flood that does not propagate to the rectifying section.

As flooding approaches, the rate of liquid entrained by the vapor sharply rises. As the entrainment accelerates, efficiency and separation plunge. The loss of separation is best recognized from laboratory analyses of column products. Another good indicator of separation loss is the column temperature profile. Liquid accumulation is often indicated as a temperature rise above the flooded tray because the accumulating liquid is richer in heavy and because the flooded trays no longer achieve an efficient separation. A rise in temperature may also occur below the flooded section because the reduced downflow of liquid from the flooded section leads to the heating up of this section and because the higher pressure drop increases the boiling point of the liquid [73]. For best results, the application of this method requires a good knowledge of the normal and flooded temperature profiles under similar feed conditions [81].

2.3.4 Causes of Flooding

Flooding in a distillation column is characterized by liquid accumulation in the column. There are different mechanisms that can cause this liquid accumulation in the column.

1. Entrainment: An increase in the vaporization rate of the reboiler will cause more vapor to be boiled up, which increases vapor flow upward the column and subsequently carry-over of liquid from the tray below to the tray above [73]. This causes the liquid to be entrained in the vapor. Upon further increase in

the vapor flow rate, massive entrainment of the liquid begins causing liquid accumulation and subsequently flooding in the tray above. The schematic of the flow on a tray of a distillation column is shown in Fig.2.4.

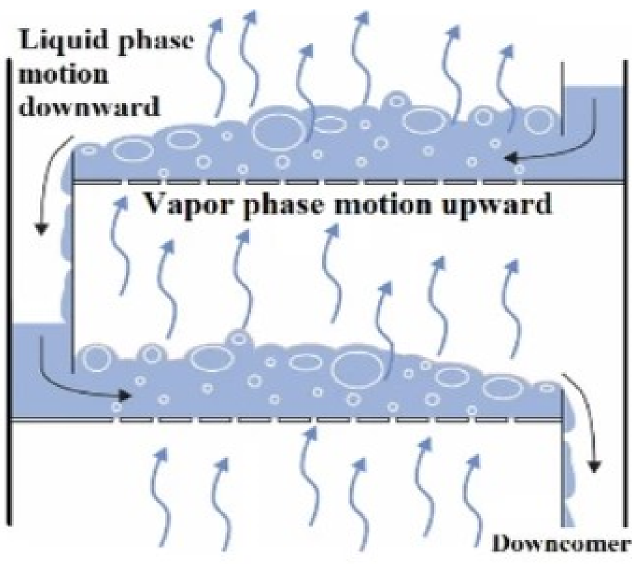


Figure 2.4: Schematic of a distillation column tray.

2. Downcomer restriction: The downflow of liquid in a distillation column can be impeded by restrictions in the downcomers [82]. A downcomer must be sufficiently large to transport all of the liquid downflow. Excessive friction losses in the downcomer entrance and/or excessive flow rate of vapor venting from the downcomer in counter-flow will impede liquid downflow, initiating liquid accumulation (termed downcomer choke flooding) on the tray above [73]. Also, aerated liquid backs up in the downcomer because of tray pressure drop, liquid height on the tray, and frictional losses in the downcomer apron. All of these increase with increasing liquid rate. Tray pressure drop also increases as the vapor rate rises. When the backup of aerated liquid exceeds the (tray spacing + weir height), i.e., fills up the downcomer, liquid accumulates on the tray above, causing downcomer backup flooding.

3. Internal damage: Damages to trays or defective internal components can also lead to liquid accumulation on distillation column trays, which can result in loss of active areas.

2.4 Flooding Detection

Based on the literature, previous research on ways of detecting flooding focused on four main aspects: Visual detection [80], liquid holdup measurements [73], acoustic signal analysis [83], and pressure monitoring [84]. Other approaches involve using machine learning methods [85, 86].

2.4.1 Visual Detection

Flooding can be detected by visually observing the buildup of liquid on the surface of distillation column packings. This approach is a simple flooding monitoring method if the column used is transparent [87]. Sight glasses can also be used to visually indicate flooding [76, 80]. Sight glasses are expensive, increase the leakage potential, and may lead to a chemical release if the glass breaks. Supplying a light source that will permit observation can also be an issue. For these reasons, this technique is not commonly used in commercial columns. It is mainly used when the column processes non-hazardous material at near ambient pressure [73]. The problem with visual observation is that there is often a delay in reaction, and by the time flooding is noticed, damage or loss has already taken place. In addition, hysteresis effects have been observed to delay column recovery [88, 89]. With hysteresis, flooding persists until the flowrate is reduced to a level much lower than the critical flowrate, making restoration of normal column operation more difficult.

2.4.2 Liquid holdup Measurements

Flooding can also be detected by noticing the increased liquid holdup using a gamma or x-ray scan. Gamma scanning is one technique particularly suitable for flooding detection. It is powerful in diagnosing flooding, identifying the flooded regions, and often also providing insight into the nature of the flood [73]. Gamma scanning is a procedure whereby a process column is non-disruptively examined by moving a sealed radioactive source emitting gamma-ray in conjunction with a radiation detector along the exterior of the interposed column. This is illustrated in Fig.2.5.

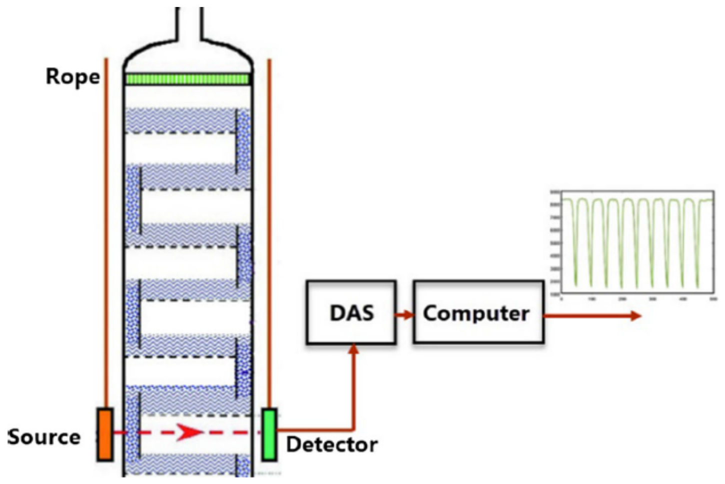


Figure 2.5: Distillation column scanning with gamma rays [90].

Throughout the investigation, the radioactive material remains permanently encapsulated within a special source housing and makes no contact either with the column or the process material. A source holder with an appropriate collimator is used to direct the radiation beam to the column. The absorbed or transmitted gamma ray intensity indicates the real quantity and nature of the material that exists between the detector and the source. Gamma scanning of distillation columns employs radioactive sources in the 500- to 2500-keV range [91]. The relation that describes the gamma rays transmitted through a material [91–94] is

$$I = I_0 e^{-\mu\rho x} \tag{2.6}$$

where I is the radiation intensity in keV, as seen by the detector; I_0 is the radiation intensity of the source in keV; ρ is the density of the medium; x is the thickness of the medium, and μ is the absorption coefficient, which depends on the gamma-ray source and the medium material.

By using a constant distance between the source and the detector during the scanning process, the measured radiation intensity will vary only when the internal material density changes, where the relation between the radiation intensity and the absorber material density is inversely proportional. The intensity of transmitted radiation is stored graphically on a computer through a data acquisition system (DAS). Radioactive sources used for distillation column investigations should be capable of penetrating the wall thickness of the column and the medium of interest. Hence, the radioactive sources used are normally cobalt-60 and cesium-137 [73].

Gamma scanning has been used to detect flooding, diagnose faults in a distillation column, and troubleshoot, optimize, and maintain the operation of distillation columns [90, 95, 96]. However, the purchase and maintenance costs are relatively high [6].

2.4.3 Acoustic Signal Detection

Flooding in a distillation column can also be detected using acoustic signal analysis. This basically involves attaching audio or sound recorders to the outside of the distillation column, the sound waves or acoustic signals generated due to propagations in the column are extracted, analyzed using different methods, and classified to know the state of the column. An illustration of this approach is shown in Fig.2.6.

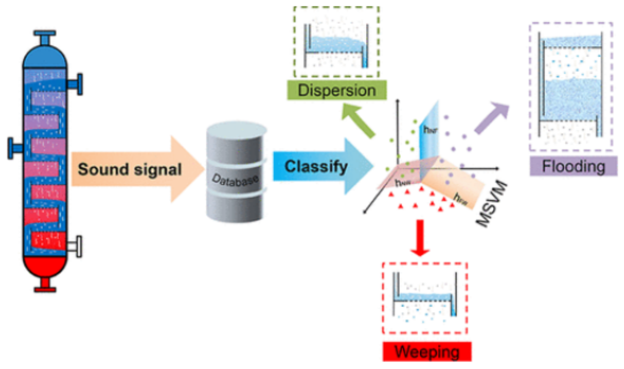


Figure 2.6: Schematic of acoustic signal analysis on a distillation column [97].

Hansuld et al. [83] proposed using microphones as an inexpensive, non-intrusive, online method of detecting flooding onset. This was done by attaching piezoelectric microphones to the outside of the column to monitor operations. Sound waves produced from the propagations of pressure imbalances in the column through fluid media deformed the piezoelectric material, thereby generating voltage signals [98]. The sound waves were analyzed using advanced signal analysis, such as standard deviation and entropy. The entropy and standard deviation results validated that acoustic signals can be used non-intrusively to monitor column operations and detect flooding.

Zhang et al. [97] also used acoustic signals to determine fluid flow states in distillation columns. This was done by performing an audio recording on three sieve tray distillation columns. Firstly, acoustic signals that are related to different fluid flow states were recorded in various columns. Then, the characteristic parameters of these acoustic signals were extracted and fused from the time, frequency, and cepstrum domains. Finally, a multi-classification support vector machine (MSVM) model was coupled with the characteristic parameters to recognize the fluid flow states in distillation columns. The results show that acoustic signals can recognize various fluid flow states in a distillation column, including flooding states.

Another application of using acoustic signals to detect flooding in a distillation column was implemented by Wang et al. [99] by combining acoustic signal analysis with k-nearest neighbor (KNN) classification algorithm to discover the running states of a distillation column. The acoustic signals were collected under normal and abnormal operations in the column. A dual-domain feature extraction method was used to extract features from the signals, which were then analyzed and compared in a generic way. The classification of the acoustic signals was done using the KNN model. The results obtained show high flooding identification accuracy.

Based on the different ways of using acoustic signals to detect flooding in the distillation column, the downside of the method is that acoustic signals are vulnerable to the surrounding environment, especially in industrial columns [6].

2.4.4 Pressure Monitoring

Flooding in distillation columns is characterized by an increase in the differential pressure across the column [75]. Hence, flooding can be detected by monitoring the pressure drop of a distillation column. A distillation column operates within a certain pressure range under normal conditions. This pressure is influenced by factors such as the feed flow rate, temperature, and composition. Trays or packing materials are where the liquid and vapor phases come into contact for separation. A typical pressure drop across these internals for a normal operating condition exists. If flooding occurs in the column, it disrupts the normal flow patterns. This can lead to a significant increase in pressure drop across the trays or packing. As a result, the pressure at certain points in the column, such as above the flooded section, would be higher than expected. Pressure sensors are strategically placed at different points along the distillation column. These sensors continuously monitor the pressure at their respective locations. The pressure readings from the sensors are fed into a control system or an alarm system. It triggers an alarm if the pressure readings exceed predefined thresholds or deviate significantly from expected values. When an alarm is triggered, operators are alerted to investigate the issue. They can then take corrective actions, such as adjusting the feed rates, temperatures, or reflux ratios, to alleviate the flooding and restore normal operation.

In the literature, pressure drop monitoring for flooding detection can also be done differently. Such as Parthasarathy et al. [84] used the differential pressure drop across a distillation column and neural network model to develop a flooding indicator. The approaches involve determining whether a flooding model could be developed for the column unit with the available data. Consequently, model development and online implementation were done while special tests were conducted on the process to evaluate the models' performance and select the best predictor model. However, using only a differential pressure model would not detect flooding in a reliable manner.

Also, Pihlaja and Miller [100] verified the pressure monitoring approach in a distillation column equipped with a specific pressure sensor. This was done by sensing a differential pressure signal along a distillation flow path. This is followed by filtering the differential pressure signal, where the filtered signal is responsive to a phase inversion along the flow path. A flooding indicator was generated as a function of the filtered differential pressure signal. The flooding indicator is responsive to the onset of a flooding condition based on phase inversion such that a change in the flooding indicator indicates the onset of a flooding condition.

2.4.5 Machine Learning Approach

Recently, machine learning (ML) methods are currently being used to detect flooding in distillation columns. This involves using machine learning algorithms to learn patterns and distributions of column data during the onset of flooding. This approach builds on the usage of the flooding indicators to detect flooding in a distillation column.

Oeing et al. [85] tested different supervised ML algorithms to detect flooding in a spinning band distillation column (SBDC). The SBDC consists of a DN25 glass column with a solid rotating internal (spinning band) manufactured by Normag-Pfaudler, Ilmenau, Germany. The SBDC has a thin gap for the countercurrent liquid-vapor flow between the spinning band and the wetted glass wall of the column. The rotation of the spinning band is used to induce an intensified mass transport between liquid and vapor to achieve a better separation efficiency with a higher rotation speed. Increasing the band speed increases pressure drop due to higher loading and higher vapor velocities in the column. Depending on the band speed, liquid loads, and vapor velocity at a certain point, the liquid accumulates, which results in flooding. The accumulation of liquid at a specific location negatively affects the desired separation process. In the case of the SBDC, the pressure drop is massively influenced by the spinning band speed. The complex hydrodynamics of the system with rotating internal and two-phase flow is arduous to model. Hence, they predict the pressure drop and classify the current operating point with the help of a trained ML algorithm with historical data. This made it possible to get information about certain parameter sets and classifications even before the undesired state, like the flooding point, is in operation. The prediction of the pressure drop was done using different supervised ML algorithms, and the classification of the operating point was done using a clustering algorithm. Linear regression, random forest, extra trees, adaboost, gradient boosting, and a combination of adaboost and extra trees are the supervised ML algorithms that were used to predict the pressure drop. The result shows that the gradient boost algorithm outperforms other algorithms in predicting the pressure drop and also in predicting flooding in the SBDC.

Also, Ochoa-Estopier et al. [86] applied a trained binary classification random forest model to predict the risk of reaching a pre-flooding operation state in a TotalEnergies refinery located in France. This approach does not rely on direct pressure measurements of the flooded sections to detect flooding. Instead, it relies on real-time measurements such as flowrates, liquid levels, temperatures, top and bottom column pressure, and domain indicators. These measurements and other developed flooding indicators used as additional variables are used to train the binary classification random forest model. The process engineers developed two flooding indicators as additional variables for the training. The first indicator was formulated based on flooding correlation. It was developed as a binary variable. The second flooding indicator was developed in the refinery by the process experts. It is a variable obtained empirically by correlating the evolution of pressures and temperatures at selected column points during a 30-minute period preceding the flooding event. Although sensitive details about the indicator were not disclosed, the formulation was based on the temperature-pressure relationship. Data labeling was done using a labeling procedure to classify the data collected into Normal, Flooding, Pre-flooding, and Post-flooding. Once the label Flooding is assigned at time t, the Pre-flooding label is consequently assigned to all available instances between $t - 60$ min and t. Similarly, the Post-flooding label is assigned to all available instances between time t and $t + 8$ h. Their objective is to build a model that can warn the operator of the risk of flooding (i.e., pre-flooding conditions) following a period of normal operation. Hence, data that belong to Post-flooding and Flooding classes were not included in the model and thus were removed from the dataset. A binary numerical variable, called flooding index FI, was introduced to treat the classes as numerical values within the model. FI is equal to zero for the points labeled as Normal and equal to one for points labeled as Pre-flooding. This led to a significant imbalance between the Normal and Pre-flooding classes. In this case, the Pre-flooding class is the minority class and the most important class as well. A resampling procedure was used to tackle class imbalance between normal and pre-flooding by duplicating the Pre-flooding data points n times while the Normal data points were selected randomly until a desired ratio was achieved. The binary classification random forest model was trained and tested, and the results obtained show that flooding events can be detected in advance while keeping a low number of false negative predictions.

2.5 Machine Learning

Most of the aforementioned methods are traditional methods for detecting and preventing flooding. Though they can provide some flooding indications, they often lack the precision and adaptability required for real-time monitoring. These conventional methods can be insufficient where process conditions frequently change, such as in a dynamic industrial environment. As a result, there is a growing need for more advanced and reliable techniques to predict and mitigate the risk of flooding in distillation columns. Machine learning (ML), a subset of artificial intelligence [101], provides promising solutions to this problem. ML methods are intelligent solutions in process monitoring and fault detection. These methods aid in shifting from reliance on the human element, which is prone to mistakes, to automated and contemporary functioning. ML algorithms are gaining ground in process engineering applications due to the large amount of data generated by the industry, and these data are available from physical sensors [7]. Data collected from the physical sensors can be combined and processed into meaningful data through ML algorithms. The processed data can be used for training data-driven models so as to capture the process conditions [8]. ML can be classified as supervised, unsupervised, or reinforcement learning [102].

2.5.1 Supervised Learning

In supervised learning, the goal is to learn a mapping from inputs to outputs given a set of labeled input-output pairs of data [103]. Supervised learning is mainly used for classification or prediction, where the model is trained on well-labeled input data. Then, the model is able to classify unseen data into the desired label or forecast the future value. Examples of the algorithms to achieve this task of regression or classification are linear regression [104], support vector machines (SVMs) [105], and decision tree-based models such as random forest, AdaBoost or gradient boosting regressor [106–108].

Linear regression is straightforward and easy to implement. It can also be regularised to prevent overfitting and can be updated easily using new data through stochastic gradient descent [109]. However, linear regression performs poorly in the presence of nonlinear relationships and cannot capture complex patterns due to lack of flexibility [110].

SVMs can handle both classification and regression problems. SVMs use a mechanism called kernel to calculate the distance between two observations and then find a decision boundary that maximizes the distance between the closest members of separate classes. In this method, a hyperplane, which is the decision boundary, needs to be defined. SVMs can model nonlinear decision boundaries [109]. They have a lot of kernels to choose from and are fairly robust against overfitting. However, SVMs are memory intensive and tricky to tune due to selecting the right kernel. Also, they don't scale well with large datasets.

The tree ensembles have the ability to handle large amounts of data and can provide good accuracy due to the combination of multiple estimators [106]. The regressor trees can be combined with bagging or boosting techniques to control overfitting and improve accuracy by building a group of estimators [111]. They are fairly robust to outliers and can learn nonlinear relationships.

One high-impact area of progress in supervised learning in recent years involves deep networks, which are multilayer networks of threshold units, each of which computes some simple parameterized function of its inputs [112, 113]. Deep learning systems make use of gradient-based optimization algorithms to adjust parameters throughout such a multilayered network based on errors at its output [114].

2.5.2 Unsupervised Learning

Unsupervised learning is that in which only inputs are known to the learning algorithm. The data is unlabelled, meaning it has not been organized into groups or factors. Unsupervised learning involves finding interesting patterns in the input data [103].

Principal component analysis (PCA) and partial least square (PLS) regression are popular unsupervised learning algorithms as they are easy to implement and can be combined with regression models for online predictions of high dimensional data [9]. The popularity of these algorithms is based on their ease of implementation and capability of tackling some problems, which include dimensionality reduction of data and extraction of key features from the data. The major disadvantages of PCA include loss of information if the number of principal components in PCA is not carefully chosen, the independent variables become less interpretable since the original dataset will be transformed into principal components, and data normalization is a must in PCA; hence, it tends to be biased towards features with high variance.

Clustering is also a popular method to find patterns in unlabeled data based on their similarity. Clustering can be hierarchical (finding successive clusters using previously established clusters) or partitional (determining all clusters at a time) [115]. Agglomerative clustering is an example of hierarchical clustering, while k-means and k-medoids algorithms are examples of partitional clustering. K-means is the most popular clustering algorithm because it is easy to implement, fast, and flexible. However, it requires the user to specify the number of clusters a priori, and it is sensitive towards outliers. K-means is limited to spherical clusters only, it cannot handle overlapping or arbitrary clusters [109] hence, DBSCAN, which is a density-based clustering algorithm, is being used to discover arbitrary-shaped clusters in data [116].

2.5.3 Reinforcement Learning

Reinforcement learning (RL) is a type of ML where an agent learns to make decisions by performing actions in an environment to maximize cumulative reward [117]. Unlike supervised learning, RL does not require labeled input/output pairs and learns from the consequences of actions. This means that instead of training examples that indicate the correct output for a given input, the training data in reinforcement learning are assumed to provide only an indication as to whether an action is correct or not; if an action is incorrect, there remains the problem of finding the correct action [118]. In reinforcement learning, an agent takes incremental actions, assesses the reward for that action, and either continues forward or changes direction as a consequence of the reward. These techniques are a form of human learning emulation, modeling how a person learns through trial and error using a limited framework of knowledge, like how a toddler learns to balance upright and walk by experiencing movements that work and do not work.

Other popular ML algorithms include probability-based techniques such as Gaussian processes [119] and artificial neural networks (ANN). For ANN, Gated recurrent units (GRU) and long short-term memory units (LSTM) have proven effective for time series data. LSTM has the ability to handle long-term dependencies and capture complex patterns in sequential data. LSTM cells can avoid the vanishing or exploding gradient problem, allowing them to learn from longer sequences without losing or amplifying the information [120]. The drawback of LSTM is that it is computationally expensive, and as a result of the extra parameters in LSTM, it requires more time to train. LSTM cells are more prone to overfitting, necessitating regularization techniques such as dropout, weight decay, or early stopping. Due to the vast choices of ML models, it is a common practice to test different models and compare their performances.

ML models rely heavily on good-quality data for accurate predictions, and the scarcity of samples seriously harms the performance of these models. According to Zhuo and Ge [121], the approach to handle a situation like this can be classified into two aspects: algorithm and data. For classification tasks, algorithm approaches are to design classifiers that can handle few samples, and semi-supervised learning such as semi-supervised ladder network [122] can be used to learn precise data distribution of the unlabeled samples. Also, as a means to trade off the minority class and gain better accuracy across all classes, an imbalance learning algorithm [123] and cost-sensitive gated neural networks [124] can be implemented. Data approaches deal with generating synthetic or virtual samples that can be used for augmenting the dataset, the augmented dataset can then be used to train classifiers. One of the ways of generating synthetic samples is over-sampling; an example can be seen in [125]. For regression tasks where a continuous target variable is being considered, Torgo et al. [126] proposed a resampling strategy that uses synthetic minority over-sampling technique (SMOTE) related approach for under-sampling and oversampling within a regression context for a case of rare extreme value prediction. However, other approaches for more effective data augmentation are deep learning-based, and a representative is the generative adversarial networks (GAN), which was first proposed in 2014 for image generation [127]. GAN has been proven to generate better samples used for data augmentation in fault detection [128, 129].

One of the applications of ML is time series forecasting, where the variables are time-stamped or dependent on time order. Generating synthetic samples for time series data requires an algorithm that will take the temporal order of that data into consideration, i.e., the temporal dynamics should be preserved. Moniz et al. [130] proposed a resampling strategy for imbalanced time series data where an interval of the data is important to the user but severely underrepresented in the training data. This resampling strategy changed the original data distribution to meet the user criteria. An effective form of generating virtual time series samples is proposed by Yoon et al. [131], where GAN is used to generate time series data through a framework called time-series generative adversarial networks (TimeGAN).

2.6 Choice of Machine Learning Algorithms

Due to the availability of different ML algorithms, several algorithms must be tested to choose the best for a specific objective. The ML algorithms considered in this work are described below with their mathematical formulations.

2.6.1 Linear Regression

Linear Regression is considered in this work because of its easy way of implementation. It provides the simplest form of a regression problem. The implementation of a linear regression model is given below according to [132]:

Consider data $\mathcal{D} = \{(y_i, x_i) : i = 1, 2, \ldots, n\}$ where y_i is the ith response, measured on a continuous scale; $\mathbf{x}_i = (x_{i1}, ..., x_{ip})^t \in \mathbb{R}^p$ is the associated predictor vector; and n ($\gg$ p) is the sample size. The linear model is specified as

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i \quad \text{with} \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2) \tag{2.7}$$

$$\text{for} \quad i = 1, 2, \ldots, n$$

In matrix form,

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I) \tag{2.8}$$

where $\mathbf{y} = [y]_{n\times1}$ is the n-dimensional response vector; $\mathbf{X} = (x_{ij})_{n\times(p+1)}$ with $x_{i0} = 1$ is often called the design matrix; $\epsilon = [\epsilon_i]_{n\times1}$ is a random error component matrix; and $\beta = [\beta_i]_{1\times(p+1)}$ is the unknown constant matrix that must be estimated

There are several estimation methods available for linear models, including least squares, maximum likelihood, bayesian approach, robust estimation, ridge regression, and so on [104] to estimate the unknown parameters β and σ^2 with the most popular method being the least square method.

The least-squares criterion, which minimizes the distance from the observed response to the predicted values is given by

$$\begin{aligned} S(\beta) &= \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \right)^2 \\ &= (\mathbf{y} - \mathbf{X}\beta)^t (\mathbf{y} - \mathbf{X}\beta) \end{aligned} \tag{2.9}$$

Differentiating with respect to β gives

$$\frac{\partial S(\beta)}{\partial \beta} = -2(\mathbf{X^t y} - \mathbf{X^t X}\beta) \tag{2.10}$$

Setting Eq. 2.10 to 0 yields the normal equation

$$\mathbf{X^t y} = \mathbf{X^t X}\beta$$

Let us assume that $\mathbf{X}$ is of full column rank p. Thus, the Gram matrix $\mathbf{X^t X}$ must be positive definite (p.d.). The least squares estimator (LSE) $\hat{\beta}$ exists as a unique solution to the normal equation and is given by

$$\hat{\beta} = (\mathbf{X^t X})^{-\mathbf{1}}\mathbf{X^t y} \tag{2.11}$$

Subsequently, the vector of fitted values; $\hat{\mathbf{y}}$ is

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X^t X})^{-\mathbf{1}}\mathbf{X^t y} = \mathbf{H y} \tag{2.12}$$

where $\mathbf{H} = \mathbf{X}(\mathbf{X^t X})^{-\mathbf{1}}\mathbf{X^t}$ is often called the hat matrix or the projection matrix. The difference between the observed value y_i and the corresponding fitted value $\hat{y}_i$ is the residual $e_i = y_i - \hat{y}_i$. The n residuals may be conveniently written in matrix notation as

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{H y} = (\mathbf{1} - \mathbf{H})\mathbf{y} \tag{2.13}$$

Moreover, the minimized least-squares criterion leads to

$$S(\hat{\beta}) = \|e\|^2$$

which is often referred to as the residual sum of squares or the sum of squares for error (SSE), a natural unbiased estimator of σ^2 is given by

$$\hat{\sigma}^2 = \frac{SSE}{n - (p+1)} \tag{2.14}$$

2.6.2 Random Forests

Random Forest is an ensemble method that builds multiple decision trees and averages their predictions [133]. Random forest as defined in [134] is a generic principle of classifier combination that uses tree-structured based $\{h(\mathbf{x}, \Theta_k), k = 1, \ldots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors, and each tree casts a unit vote for the most popular class at input **x**. Random forests can be used for both regression and classification tasks [134].

Every Decision Tree is made by randomly selecting the data from the available data. For example, a Random Forest for each Decision Tree (as in Random Subspaces) can be built by randomly sampling a feature subset and/or by the random sampling of a training data subset for each Decision Tree (the concept of Bagging) as illustrated in Fig.2.7.

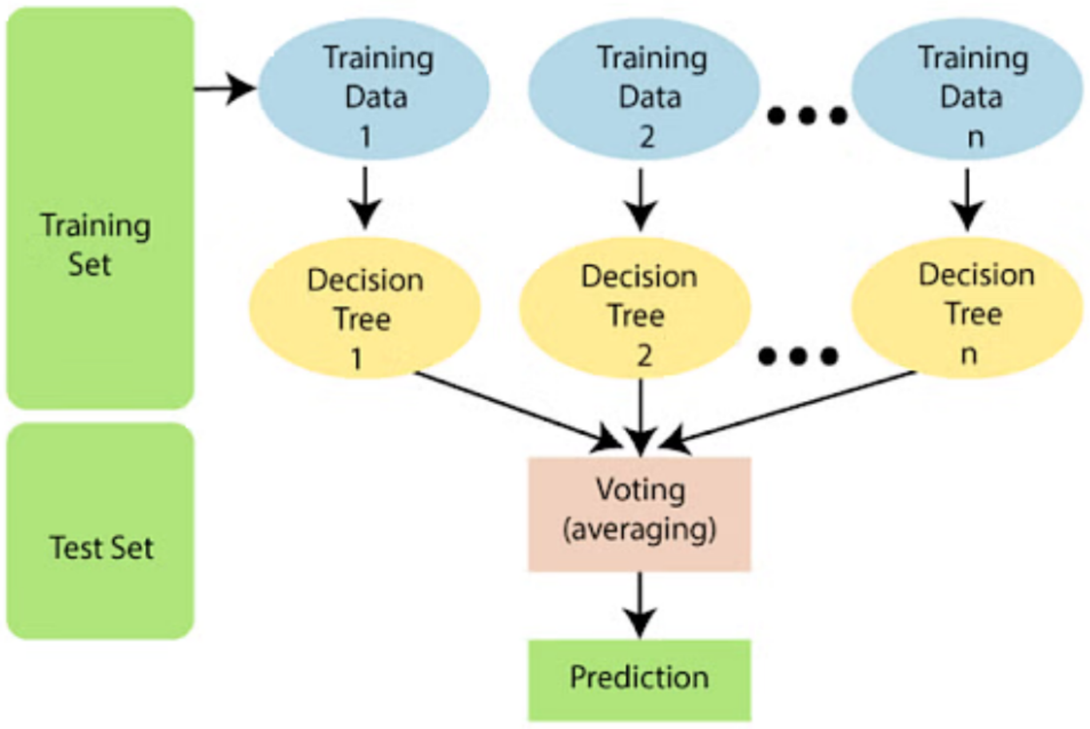


Figure 2.7: Illustration of random forests algorithm.

.

In a Random Forest, the features are randomly selected in each decision split. The correlation between trees is reduced by randomly selecting the features that improve the prediction power and result in higher efficiency. The advantages of random forests include [133]

- Overcoming the problem of overfitting
- Training data are less sensitive to outliers

- Variable importance and accuracy is generated automatically
- Eliminates the need for pruning the trees since parameters can be set easily.

The steps below explain the implementation of the random forests algorithm as given in [133]. let $T_K(x)$ be the k-th tree in the forest.

1. By Sampling N randomly, If the number of cases in the training set is N but with replacement from the original data. This sample will be used as the training set for growing the tree
2. For M number of input variables, the variable m is selected such that $m \gg M$ is specified at each node, m variables are selected at random out of the M, and the best split on this m is used for splitting the node. During the forest growing, the value of m is held constant.
3. Each tree is grown to the largest possible extent. No pruning is used.
4. Voting will take place by averaging the decision tree such that

$$RF(x) = \frac{1}{N} \sum_{k=1}^{N} T_K(x)$$

5. Finally, select the most voted prediction result as the final prediction result.

Random Forest generally exhibits a significant performance improvement as compared to a single tree classifier. The generalization error rate that it yields compares favorably to Adaboost. However, it is more robust to noise [133]

2.6.3 Extra Trees

Extremely Randomized Trees or Extra Trees (ET) algorithm is an ensemble approach based on a large number of decision trees [135]. The idea behind the ensemble technique is to combine the decisions of distinct models and make a judgment based on that combination, which essentially results in better performance compared to the achievements of a single decision or model. The ensemble technique is used in a vast number of applications for classification and regression tasks [136].

The Decision trees-based ensemble technique can achieve high performance when the base learners are independent, and that can be attained through randomization. When growing the trees, randomization entails better tree diversity and facilitates reducing the correlation [137]. One might say that ensemble learning methodologies work on the principles of the divide-and-conquer approach (or the wisdom of the crowd) to achieve enhanced performance. In supervised ML tasks, we can get a stable and more robust classifier (model) with precise predictions using an ensemble technique because it reduces the factors, i.e., noise, bias, and variance. However, an ensemble learner can cause a notable rise in computational costs due to the need to train a number of individual classifiers.

The ET algorithm consists of a number of Decision trees, where each tree is composed of a root node, child/split nodes, and leaf nodes, as shown in Fig.2.8. Given a dataset X at the root node, ET selects a split rule based on a random subset of features and a partially random cut point. In each child node, this procedure is repeated until reaching a leaf node. Furthermore, the three most important parameters of ET can be outlined as the number of trees in the ensemble (k), the number of attributes/features to select randomly (f), and the minimum number of samples/instances required to split a node (n_{min}).

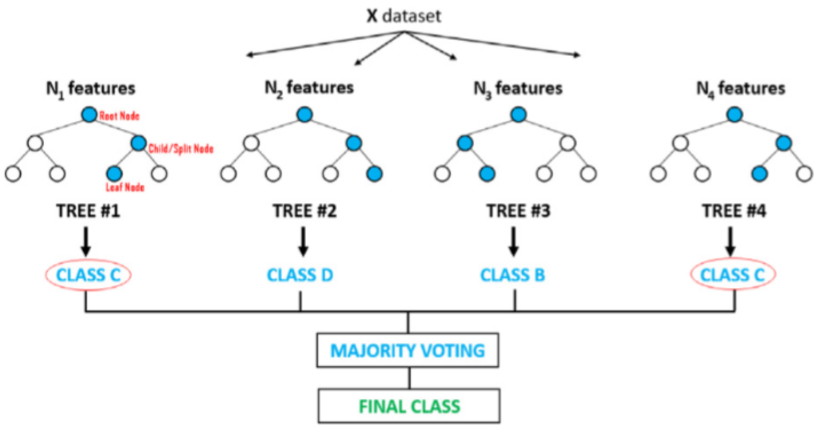


Figure 2.8: Illustration of the Extra Trees Algorithm.

As an ensemble of individual trees, the ET algorithm is similar to the regular Random forests, but with two key differences. First, the entire learning sample is used to train each tree instead of training a bootstrap sample. Second, the top-down splitting of nodes in the tree is with completely random splits, not the best splits

[135]. A random cut-point is used instead of calculating the locally optimal cut-point for each attribute being considered based on gini impurity or information gain. This value is selected from a uniform distribution within the attribute's empirical range (in the training set of the tree). Subsequently, the split that produces the highest score of all the randomly generated splits is selected for splitting the node. Since finding the best split at every node for each attribute or feature is highly time-consuming when growing a Decision tree, the process of ET makes it much faster to train than an ordinary Random forest algorithm. Also, ET outperforms Random Forests when there are noisy points in the data [138].

Furthermore, in the testing process, a test sample proceeds through each of the Decision Trees and to each child node, choosing the best splits and forwarding the test sample to the tree's right/left child node before a leaf node is reached. The leaf node determines the class for the test sample in any Decision Tree, and the final prediction is called the majority of votes by the (k) decision trees of the ET algorithm (Fig. 2.8).

The generalization error of the ML model can be declared as the sum of unique errors, i.e., bias and variance. A high bias can give rise to underfitting, which can be calculated as the ability to generalize unseen data accurately. In other circumstances, a high variance can arouse overfitting, which is provoked by the intense sensitivity of the model to inconsequential variations in the training set. The ET algorithm has the ability to strongly reduce bias and variance error better than any other randomization method, i.e., random forest. The variance is minimized by the selection of the cut-point and the explicit randomization of the subset of attributes, whereas the bias is minimized due to the complete use of the original training set to learn the individual Decision Tree [135].

Furthermore, a major advantage of ET during implementation is that it does not need immense concentration towards the selection of hyperparameter values. The ET model is quite robust to noise from an individual DT such that, typically, there is no need to prune. The general working steps of the ET algorithm [138] are summarized in Table 2.1. In practice, the number of trees k (step 4) is considered to be the single parameter that needs to be taken care of while constructing the ET mode.

Table 2.1: Steps towards Extra Trees Algorithm

1:	Construct a training set of size S
2:	Randomly select n learning samples without replacement from training set S(Bootstrap=False)
3:	Build a tree from the entire learning sample. At each node:
	3.1: Randomly select f features without replacement.
	3.2: Split the node by random cut-points.
4:	Repeat, k times, steps 2-3
5:	Aggregate the results of each tree to assign the respective class (majority voting)

2.6.4 Gradient Boosting

The common ensemble techniques like random forests rely on simple averaging of models in the ensemble [134]. The family of boosting methods is based on a different, constructive strategy of ensemble formation. The main idea of boosting is to add new models to the ensemble sequentially. At each particular iteration, a new weak, base-learner model is trained with respect to the error of the whole ensemble learned so far [139].

In gradient boosting machines, or simply GBMs, the learning procedure consecutively fits new models to provide a more accurate estimate of the response variable. The principle idea behind this algorithm is to construct the new base learners to be maximally correlated with the negative gradient of the loss function associated with the whole ensemble. The loss functions applied can be arbitrary, but to give a better intuition, if the error function is the classic squared-error loss, the learning procedure would result in consecutive error-fitting [140]. In general, the choice of the loss function is up to the researcher, with both a rich variety of loss functions derived so far and the possibility of implementing one's own task-specific loss [139].

For a dataset $(x, y)_{i=1}^{N}$, where $x = (x_1, \dots, x_d)$ refers to the explanatory input variable and y to the corresponding label of the response variable. The goal is to reconstruct the unknown functional dependence $x \xrightarrow{f} y$ with estimate $\hat{f}(x)$ such that some specified loss function $\psi(y, f)$ is minimised:

$$\hat{f}(x) = y$$

$$\hat{f}(x) = \underset{f(x)}{\operatorname{argmin}}\, \psi(y, f) \tag{2.15}$$

If we rewrite the estimation problem in terms of expectations, the equivalent formulation would be to minimize the expected loss function over the response variable $E_y(\psi[y, f(x)])$, conditioned on the observed explanatory data x:

$$\hat{f}(x) = \underset{f(x)}{\operatorname{argmin}} \; \underbrace{E_x[\overbrace{E_y(\psi[y, f(x)])}^{expected\ y\ loss}\,|x]}_{expected\ over\ the\ whole\ dataset} \tag{2.16}$$

The response variable y can come from different distributions. This naturally leads to a specification of different loss functions ψ. In particular, if the response variable is binary, i.e., $y \in 0, 1$, one can consider the binomial loss function. If the response variable is continuous, i.e., $y \in \mathrm{R}$, one can use the classical L_2 squared loss function or the robust regression Huber loss.

To make the problem of function estimating tractable, we can restrict the function search space to a parametric family of functions $f(x, \theta)$. This would change the function optimization problem into the parameter estimation one:

$$\hat{f}(x) = f(x, \hat{\theta}) \tag{2.17}$$

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \, E_x[E_y(\psi[y, f(x, \theta)])|x] \tag{2.18}$$

Typically, the closed-form solutions for the parameter estimates are not available. To perform the estimation, iterative numerical procedures are considered. In practice, given some specific loss function $\psi(y, f)$ and/or a custom base-learner $h(x, k)$, the solution to the parameter estimates can be difficult to obtain [139]. To deal with this, it was proposed to choose a new function $h(x, \theta_t)$ to be the most parallel to the negative gradient $\{g_t(x_i)\}_{i=1}^N$ along the observed data:

$$g_t(x) = E_y \left[\frac{\partial \psi(y, f(x)}{\partial f(x)} |x \right]_{f(x)=\hat{f}^{t-1}(x)} \tag{2.19}$$

Instead of looking for a general solution for the boost increment in the function space, one can simply choose the new function increment to be the most correlated with $-g_t(x)$. This permits the replacement of a potentially very hard optimization task with the classic least-squares minimization one:

$$(\rho, \theta_t) = \underset{\rho,\,\theta}{\operatorname{argmin}} \sum_{i=1}^{N} [-g_t(x_i) + \rho h(x_i, \theta)]^2 \tag{2.20}$$

To summarize, the complete formulation of the gradient boosting algorithm as originally proposed by Friedman [140] is given in Table 2.2.

Table 2.2: Gradient Boost Algorithm

```
Inputs:
• input data $(x, y)_{i=1}^{N}$
• number of iterations $M$
• choice of loss-function $\psi(y, f)$
• choice of the base-learner model $h(x, k)$
Algorithm:
1: initialize $\hat{f}_0$ with a constant
2: for $t = 1$ to $M$ do
3:    compute the negative gradient $g_t(x)$
4:    fit a new base learner function $h(x, \theta_t)$
5:    find the best gradient descend step-size $\rho_t$
      $\rho_t = \text{argmin}_\rho \sum_{i=1}^{N} \psi[y_i, \hat{f}_{t-1}(x_i) + \rho h(x_i, \theta_t)]$
6:    update the function estimate:
      $\hat{f}_t \leftarrow \hat{f}_{t-1} + \rho_t h(x, \theta_t)$
7: end for
```

2.6.5 Long Short-Term Memory (LSTM)

Recurrent neural networks (RNNs) with long short-term memory (LSTM) have emerged as an effective and scalable model for several learning problems related to sequential data [141]. Earlier methods for attacking these problems have either been tailored toward a specific problem or did not scale to long-term dependencies. LSTMs, on the other hand, are both general and effective at capturing long-term temporal dependencies. They do not suffer from the optimization hurdles that plague simple recurrent networks (SRNs) [142]. LSTMs have been used to advance the state of the art for many difficult problems such as handwriting recognition [143], acoustic modeling of speech [144], and analysis of audio [145], among others.

The central idea behind the LSTM architecture is a memory cell, which can maintain its state over time, and nonlinear gating units, which regulate the information flow into and out of the cell. A variety of LSTM cell configurations have been described since the first LSTM introduction in 1997 [146]. A simple RNN cell (Fig.2.9a) was extended by adding a memory block, which is controlled by input and output multiplicative gates. Fig.2.9b demonstrates the LSTM architecture of the jth cell c_j. The heart of a memory block is a self-connected linear unit s_c, also called a "constant error carousel" (CEC). CEC protects LSTM from vanishing and exploding gradient problems of traditional RNNs. An input and output gates consist of corresponding weight matrices and activation functions. The input gate with weighted input net_in and output y^{in} is capable of blocking irrelevant data from entering the cell. Similarly, the output gate with weighted input net_out and y^{out} shapes the output of the cell y^c. Overall, it can be concluded that LSTM cells consist of one input layer, one output layer, and one self-connected hidden layer. The hidden unit may contain "conventional" units that can be fed into subsequent LSTM cells.

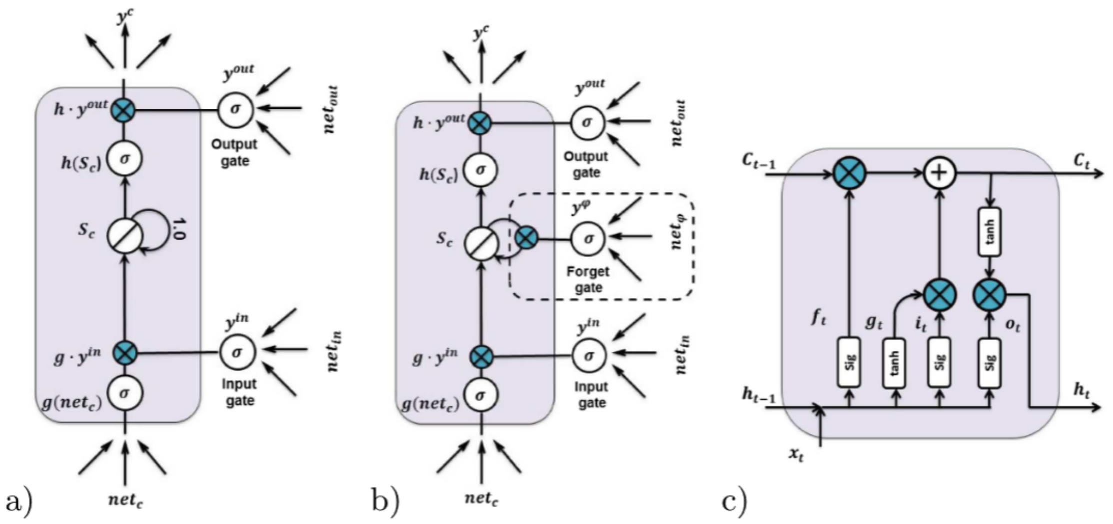


Figure 2.9: (a) An original LSTM unit architecture: a memory cell and two gates; (b) LSTM cell with forget gate; (c) modern representation of LSTM with forget gate.

Nevertheless, a standard LSTM cell also met some constraints due to the linear nature of s_c. It was identified that its constant growth may cause saturation of the function h and convert it into an ordinary unit. Therefore, an additional forget gate layer was included [147]. A new gate allows unneeded information to be erased and forgotten. Fig.2.9b shows a new cell architecture. In addition, in late works, gates are included in the cell, and LSTM architecture of frequent occurrence is demonstrated in Fig.2.9c.

The feedforward behaviour of the most commonly used configuration can be described by Eqs. (2.21 - 2.26). The cell input x_t at time t concatenates with an output of a cell h_{t-1} at the previous time step $t-1$. The resulting vector goes through the input node (g_t) and forgets (f_t), input (i_t), and output gates (o_t).

$$g_t = \tilde{C}_t = \tanh W^{(g)} x_t + U^{(g)} h_{t-1} + b^{(g)} \tag{2.21}$$

$$f_t = \sigma(W^{(f)} x_t + U^{(f)} h_{t-1} + b^{(f)}) \tag{2.22}$$

$$i_t = \sigma(W^{(i)} x_t + U^{(fi)} h_{t-1} + b^{(i)}) \tag{2.23}$$

$$o_t = \sigma(W^{(o)} x_t + U^{(o)} h_{t-1} + b^{(o)}) \tag{2.24}$$

Then, the forget gate decides whether to keep cell data C_{t-1} from a previous time step or block it. The current cell memory state and output of the cell are defined by:

$$C_t = g_t \odot i_t + f_t \odot C_{t-1} \tag{2.25}$$

$$h_t = o_t \odot \tanh C_t \tag{2.26}$$

where $U^{(*)}$ is the input weight matrix, $W^{(*)}$ is the hidden layer weight matrix, σ is an activation function and symbol $\odot$ denotes a pointwise or Hadamard multiplication. Weight increment during backpropagation can be found using the equation below:

$$W^{new} = W^{old} - \lambda \cdot \delta W^{old} \tag{2.27}$$

where λ is a Stochastic Gradient Descent (SGD) coefficient and deltas $\delta W = \sum_{t=1}^{T} \delta gateS_t \cdot x_t$, $\delta U = \sum_{t=1}^{T} \delta gateS_{t+1} \cdot h_t$, $\delta b = \sum_{t=1}^{T} \delta gateS_{t+1}$. Deltas of $gateS_t$ are to be found using the following equations [148]:

$$\delta h_t = \Delta_t + \Delta h_t \tag{2.28}$$

$$\delta C_t = \delta h_t \odot o_t \odot (1 - \tanh^2(C_t)) + \delta C_{t+1} \odot f_{t+1} \tag{2.29}$$

$$\delta g_t = \delta C_t \odot i_t \odot (1 - g_t^2) \tag{2.30}$$

$$\delta i_t = \delta C_t \odot g_t \odot (1 - i_t) \tag{2.31}$$

$$\delta f_t = \delta C_t \odot C_{t-1} \odot f_t \odot (1 - f_t) \tag{2.32}$$

$$\delta o_t = \delta h_t \odot \tanh C_t \odot o_t \odot (1 - o_t) \tag{2.33}$$

$$\delta x_t = W^T \cdot \delta gateS_t \tag{2.34}$$

$$\delta h_{-1} = U^T \cdot \delta gateS_t \tag{2.35}$$

2.6.6 K-Means Clustering

Clustering is a generic tool for finding groups or clusters in multivariate data based on their similarities [149]. Several algorithms have been proposed in the literature for clustering. The k-means clustering algorithm is the most commonly used because of its simplicity [150]. K-Means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a prior [151].

The main idea is to define *k* centroids, one for each cluster. These centroids should be placed in a cunning way because different location causes different results. So, the better choice is to place them as far away from each other as possible. The next step is to take each point belonging to a given data set and associate it with the nearest centroid. When no point is pending, the first step is completed, and an early group is done. At this point, it is necessary to re-calculate *k* new centroids as centers of the clusters resulting from the previous step. After these *k* new centroids, a new binding has to be done between the same data points and the nearest new centroid. A loop has been generated. As a result of this loop, it may be noticed that the *k* centroids change their location step by step until no more changes are done. In other words, centroids do not move anymore.

K-means clustering algorithm aims at minimizing a squared error objective function (this is the sum of squared distances of samples to their closest cluster, i.e., inertia) [151]:

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} \| y_i - c_k \|^2 \tag{2.36}$$

where S is a K-cluster partition of the entity set represented by vector y_i ($i \in I$) in the M-dimensional feature space consisting of non-empty non-overlapping clusters S_k each with a centroid c_k ($k = 1, \ldots, K$).

The algorithm is composed of the following steps:

1. Place k points in the space represented by the objects that are being clustered. These points represent the initial group of centroids.

2. Assign each object to the group that has the closest centroid

3. When all objects have been assigned, recalculate the positions of the k centroids.

4. Repeat Step 2 and 3 until the centroids no longer move.

There have been a number of different proposals in the literature for choosing the right K after multiple runs of K-Means. The oldest method for determining the true number of clusters in a dataset is the elbow method [152]. It is a visual method. The idea is that Start with $K = 2$, and keep increasing it in each step by 1, the number of clusters reached just before additional clusters start leading to smaller change in inertia (i.e. an elbow where the distortion in inertia goes down slowly) is the appropriate number of clusters.. This is the K value.

2.7 Time-series Generative Adversarial Networks

Generative adversarial networks (GAN) have become one of the most popular technologies in the field of deep learning. GAN was proposed in 2014 by Ian J. Goodfellow et al. [127]. The basic principle of GAN is illustrated in Fig.2.10; it consists of two network components: a generator (G) and a discriminator (D).

G is a network that generates samples; it receives a random noise z and generates samples using noise denoted as $G(z) = X_{synthetic}$. D is a network that determines whether a sample is real or not. An output of 1 must be a real sample, while an output of 0 means a synthetic sample. The real sample X_{real} and the synthetic sample $X_{synthetic}$ are fed to the D together during the training. The goal of G is to generate real samples to deceive D, while the goal of D is to separate the samples generated

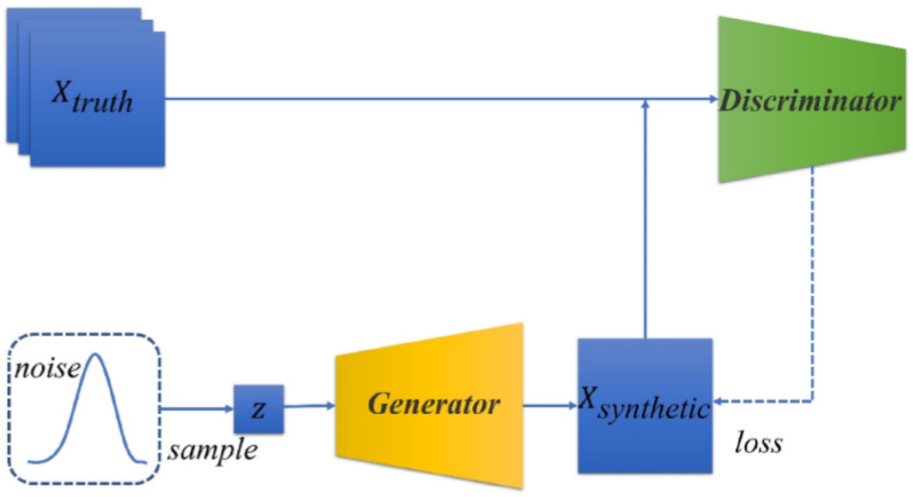


Figure 2.10: GAN architecture

by G from the real samples as much as possible. Therefore, G and D constitute a dynamic game playing, whose training process can be expressed as Eq. 2.37

$$\min_{G}\max_{D} V(D,G) = E_{X\sim p_{data}(X)}[\log D(X)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{2.37}$$

where D and G represent generator and discriminator; $V(D,G)$ represents the value function of generator and discriminator; E represents mathematic expectation; z and X represent noise and real data; $p_z(z)$ and $p_{data}(X)$ represent the distribution of noise and real data.

In the best case, G will produce a sample $G(z)$ that looks like a real sample. For D, it is difficult to determine whether the sample generated by G is real or not, so its output is 0.5. So far, a generative model has been built, which can be used to generate a synthetic sample like a real one.

Time-series Generative Adversarial Networks (TimeGAN) is a generative model variant based on GAN for time-series data [153]. TimeGAN was proposed in 2019 by Yoon et al. [131]. TimeGAN is trained adversarially and jointly via a learned embedding space with supervised and unsupervised losses. TimeGAN consists of four network components: an embedding function, a recovery function, a sequence generator, and a sequence discriminator. Hence, TimeGAN has two parts: one is an autoencoder, and the other is GAN, as shown in Fig.2.11. Based on the guidance of

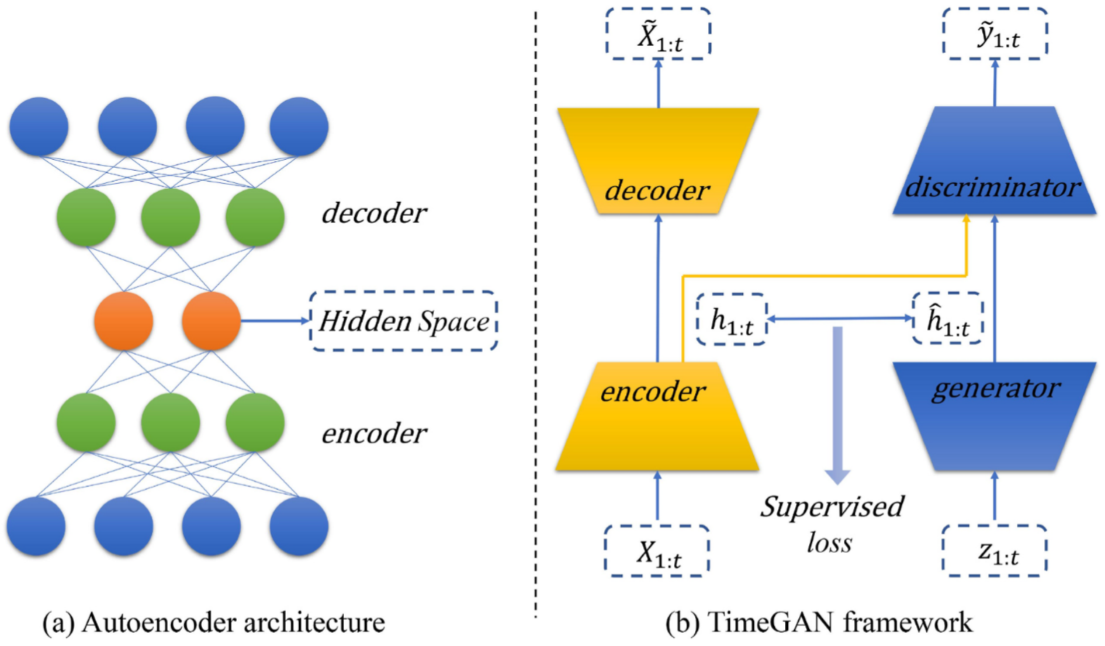


Figure 2.11: The diagram of TimeGAN [153].

the discriminator, the generator can generate a synthetic sample that achieves real feature distribution. TimeGAN combines this architecture with an autoencoder, which reduces the dimensionality of data into a hidden space and then recreates the data using hidden features. Autoencoder allows GAN to learn temporal dynamics in smaller dimensions. The hidden features generated by the generator and the hidden features generated by the encoder from the real data constitute supervised learning, enabling the generator to capture the temporal dynamics on the real data, i.e. $p(X_t|X_{1:t-1})$. In this way, a generative model is obtained that can explicitly learn both the temporal characteristics and the relationships among multiple features. The implementation of the TimeGAN described below was proposed by Yoon et al. [131].

2.7.1 Embedding and Recovery Functions

The embedding and recovery functions provide mappings between feature and latent space, allowing the adversarial network to learn the underlying temporal dynamics of the data via lower-dimensional representations. Let $\mathcal{H}_\mathcal{S}$ and $\mathcal{H}_\mathcal{X}$ denote the latent space vectors corresponding to feature space $\mathcal{S}$ and $\mathcal{X}$. The embedding function: $e : \mathcal{S} \times \prod_t \mathcal{X} \rightarrow \mathcal{H}_\mathcal{S} \times \prod_t \mathcal{H}_\mathcal{X}$ takes static and temporal feature to their latent code

$h_{\mathcal{S}}, h_{1:T} = e(s, x_{1:T})$. e can be implemented via a recurrent network

$$h_{\mathcal{S}} = e_{\mathcal{S}}(s) \quad h_t = e_{\mathcal{X}}(h_s, h_{t-1}, x_t) \tag{2.38}$$

where $e_{\mathcal{S}} : \mathcal{S} \rightarrow H_{\mathcal{S}}$ is an embedding network for static features and $e_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times \mathcal{X} \rightarrow H_{\mathcal{X}}$ is a recurrent embedding network for temporal features. In the opposite direction, the recovery function $r : \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}} \rightarrow \mathcal{S} \times \prod_t \mathcal{S}$ takes static and temporal codes back to their feature representations $\tilde{s}, \tilde{x}_{1:T} = r(h_s, h_{1:T})$. r can be implemented through a feedforward network at each step

$$\tilde{s} = r_{\mathcal{S}}(h_s), \quad \tilde{x}_t = r_{\mathcal{X}}(h_t) \tag{2.39}$$

where $r_{\mathcal{S}} : \mathcal{H}_{\mathcal{S}} \rightarrow \mathcal{S}$ and $r_{\mathcal{X}} : \mathcal{H}_{\mathcal{X}} \rightarrow \mathcal{X}$ are recovery networks for static and temporal embeddings.

2.7.2 Sequence Generator and Discriminator

The Generator first outputs into the embedding space instead of producing synthetic output directly in the feature space. Let $\mathcal{Z}_{\mathcal{S}}, \mathcal{Z}_{\mathcal{X}}$ denote vector spaces over which known distributions are defined, and from which random vectors are drawn as input for generating into $\mathcal{H}_{\mathcal{S}}, \mathcal{H}_{\mathcal{X}}$. The generating function $g : \mathcal{Z}_{\mathcal{S}} \times \prod_t \mathcal{Z}_{\mathcal{X}} \rightarrow \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}}$ takes a tuple of static and temporal random vectors to synthetic latent codes $\hat{h}_{\mathcal{S}}, \hat{h}_{1:T} = g(z_{\mathcal{S}}, Z_{1:T})$. g can be implemented through a recurrent neural network.

$$\hat{h}_{\mathcal{S}} = g_{\mathcal{S}}(Z_{\mathcal{S}}), \quad \hat{h}_t = g_{\mathcal{X}}(\hat{h}_{\mathcal{S}}, \hat{h}_{t-1}, z_t) \tag{2.40}$$

where $g_{\mathcal{S}} : Z_{\mathcal{S}} \rightarrow H_{\mathcal{S}}$ is an generator network for static features and $g_{\mathcal{X}} : \mathcal{H}_{\mathcal{S}} \times \mathcal{H}_{\mathcal{X}} \times Z_{\mathcal{X}} \rightarrow H_{\mathcal{X}}$ is a recurrent genertor for temporal features. Random vector $z_{\mathcal{S}}$ can be sampled from a distribution of choice, and z_t follows a stochastic process such as Gaussian distribution and Wiener process.

Finally, the discriminator also operates from the embedding space. The discrimination function $d : \mathcal{H}_{\mathcal{S}} \times \prod_t \mathcal{H}_{\mathcal{X}} \rightarrow [0, 1] \times \prod_t [0, 1]$ receives the static and temporal codes, returning classifications $\tilde{y}_{\mathcal{S}}, \tilde{y}_{1:T} = d(\tilde{h}_{\mathcal{S}}, \tilde{h}_{1:T})$. The $\tilde{h}_*$ notation denotes either real (h_*) or synthetic ($\tilde{h}_*$) embeddings; similarly, the $\tilde{y}_*$ notation denotes classifications of either real (y_*) or synthetic ($\tilde{y}_*$) data. d can be implemented via a bidirectional recurrent network with a feedforward output layer.

$$\tilde{y}_{\mathcal{S}} = d_{\mathcal{S}}(\tilde{h}_{\mathcal{S}}), \quad \tilde{y}_t = d_{\mathcal{X}}(\overleftarrow{u}_t, \vec{u}_t) \tag{2.41}$$

where $\vec{u}_t = \vec{c}_{\mathcal{X}}(\hat{h}_{\mathcal{S}}, \hat{h}_t, \vec{u}_{t-1})$ and $\overleftarrow{u}_t = \overleftarrow{c}_{\mathcal{X}}(\hat{h}_{\mathcal{S}}, \hat{h}_t, \overleftarrow{u}_{t+1})$ respectively denote the sequences of forward and backward hidden states, $\vec{c}_{\mathcal{X}}, \overleftarrow{c}_{\mathcal{X}}$ are recurrent functions and $d_{\mathcal{S}}, d_{\mathcal{X}}$ are output layer classification functions. Similarly, there are no restrictions on architecture beyond the generator being autoregressive.

2.7.3 Joint Training

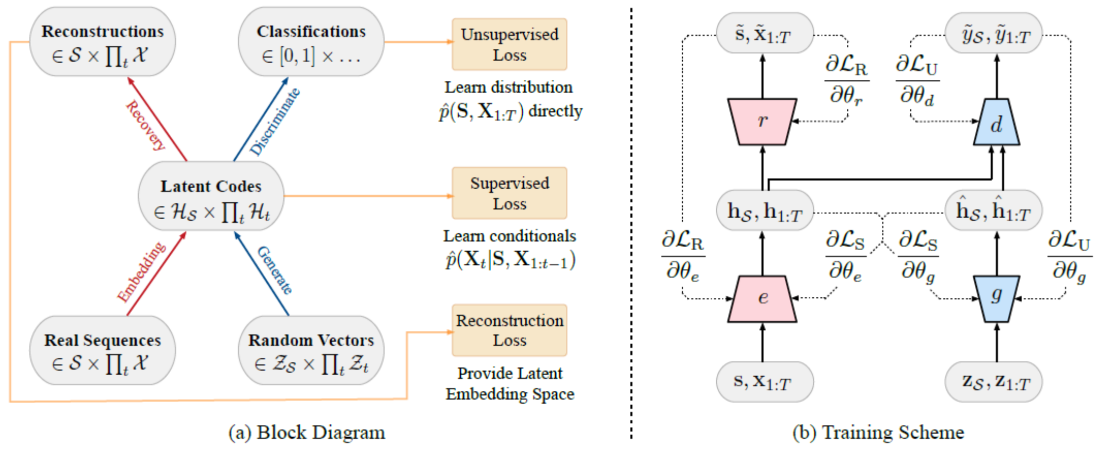
As a reversible mapping between feature and latent spaces, the embedding and recovery functions should enable accurate reconstructions $\tilde{s}, \tilde{x}_{1:T}$ of the original data $s, x_{1:T}$ from their latent representations $h_{\mathcal{S}}, h_{1:T}$. Therefore, the first objective function is the reconstruction loss given by

$$\mathcal{L}_{\mathcal{R}} = \mathbb{E}_{s, x_{1:T} \sim p}[\|s - \tilde{s}\|_2 + \sum_t \|x_t - \tilde{x}_t\|_2] \tag{2.42}$$

In TimeGAN, the generator is exposed to two types of inputs during training, as shown in Fig.2.12. First, in pure open-loop mode, the generator which is autoregressive receives synthetic embeddings $\tilde{h}_{\mathcal{S}}, \tilde{h}_{1:t-1}$ (i.e., its own previous outputs) in order to generate the next synthetic vector $\tilde{h}_t$. Gradients are then computed on the unsupervised loss to allow maximizing (for the discriminator) or minimizing (for the generator) the likelihood of providing correct classifications $\hat{y}_{\mathcal{S}}, \hat{y}_{1:T}$ for both the training data $h_{\mathcal{S}}, h_{1:T}$ as well as for synthetic output $\hat{h}_{\mathcal{S}}, \hat{h}_{1:T}$ from the generator,

$$\mathcal{L}_{\mathcal{U}} = \mathbb{E}_{s, x_{1:T} \sim p}[\log y_{\mathcal{S}} + \sum_t \log y_t] + \mathbb{E}_{s, x_{1:T} \sim \hat{p}}[\log(1 - \hat{y}_{\mathcal{S}}) + \sum_t \log(1 - \hat{y}_t)] \tag{2.43}$$

Relying solely on the discriminator's binary adversarial feedback may not be sufficient incentive for the generator to capture the stepwise conditional distributions in the data. An additional loss was introduced to further discipline learning to achieve this more efficiently. In an alternating fashion, we can also train in closed-loop mode, where the generator receives sequences of embeddings of actual data $h_{1:t-1}$ (i.e., computed by the embedding network) to generate the next latent vector. Gradients



Reconstructions $\in \mathcal{S} \times \prod_t \mathcal{X}$
Classifications $\in [0,1] \times \dots$
Unsupervised Loss
Learn distribution $\hat{p}(\mathbf{S}, \mathbf{X}_{1:T})$ directly
Recovery
Discriminate
Latent Codes $\in \mathcal{H}_\mathcal{S} \times \prod_t \mathcal{H}_t$
Supervised Loss
Learn conditionals $\hat{p}(\mathbf{X}_t|\mathbf{S}, \mathbf{X}_{1:t-1})$
Embedding
Generate
Real Sequences $\in \mathcal{S} \times \prod_t \mathcal{X}$
Random Vectors $\in \mathcal{Z}_\mathcal{S} \times \prod_t \mathcal{Z}_t$
Reconstruction Loss
Provide Latent Embedding Space
(a) Block Diagram
$\tilde{\mathbf{s}}, \tilde{\mathbf{x}}_{1:T}$
$\tilde{y}_\mathcal{S}, \tilde{y}_{1:T}$
$\frac{\partial \mathcal{L}_\text{R}}{\partial \theta_r}$
$\frac{\partial \mathcal{L}_\text{U}}{\partial \theta_d}$
r
d
$\mathbf{h}_\mathcal{S}, \mathbf{h}_{1:T}$
$\hat{\mathbf{h}}_\mathcal{S}, \hat{\mathbf{h}}_{1:T}$
$\frac{\partial \mathcal{L}_\text{R}}{\partial \theta_e}$
$\frac{\partial \mathcal{L}_\text{S}}{\partial \theta_e}$
$\frac{\partial \mathcal{L}_\text{S}}{\partial \theta_g}$
$\frac{\partial \mathcal{L}_\text{U}}{\partial \theta_g}$
e
g
$\mathbf{s}, \mathbf{x}_{1:T}$
$\mathbf{z}_\mathcal{S}, \mathbf{z}_{1:T}$
(b) Training Scheme

Figure 2.12: (a) Block diagram of component functions and objectives. (b) Training scheme; solid lines indicate forward propagation of data, and dashed lines indicate backpropagation of gradients [131].

can now be computed on a loss that captures the discrepancy between distributions $p(\mathbf{H_t}|\mathbf{H}_\mathcal{S}, \mathbf{H_{1:t-1}})$ and $\hat{p}(\mathbf{H_t}|\mathbf{H}_\mathcal{S}, \mathbf{H_{1:t-1}})$. Applying maximum likelihood yields the familiar supervised loss:

$$\mathcal{L}_\mathcal{S} = \mathbb{E}_{s, x_{1:T} \sim p}[\sum_t \|h_t - g_\mathcal{X}(h_\mathcal{S}, h_{t-1}, z_t)\|_2] \tag{2.44}$$

Where $g_\mathcal{X}(h_\mathcal{S}, h_{t-1}, z_t)$ approximates $\mathbb{E}_{z_t \sim \mathcal{N}}[\hat{p}(\mathbf{H_t}|\mathbf{H}_\mathcal{S}, \mathbf{H_{1:t-1}}, \mathbf{z_t})]$ with one sample z_t—as is standard in stochastic gradient descent. In sum, at any step in a training sequence, the difference between the actual next-step latent vector (from the embedding function) and synthetic next-step latent vector (from the generator—conditioned on the actual historical sequence of latent) is assessed. While $\mathcal{L}_\mathcal{U}$ pushes the generator to create realistic sequences (evaluated by an imperfect adversary), $\mathcal{L}_\mathcal{S}$ further ensures that it produces similar stepwise transitions (evaluated by ground-truth targets).

Chapter 3

Research Methodology

This chapter presents the justification for using machine learning methods to detect flooding in a distillation column, followed by the methodology employed in this work for flooding detection. This involves the approaches for preprocessing the collected sensor data of the distillation column. Subsequently, the approaches and mathematical formulations used for early flooding detection in a distillation column are presented. Due to the data imbalance between the normal operating data and flooding regime data, with flooding regime data being the minority as stated in the objectives statement in Section 1.4, unsupervised machine learning methods such as PCA and Autoencoders are considered for flooding detection. The data imbalance created by the insufficient flooding regime data does not affect these methods. Also, supervised learning methods are considered for flooding detection by forecasting the pressure drop of the column so that with accurate classification of the forecasted pressure drop, possible risk of flooding can be detected before it happens, leaving ample time for operators to make engineering decisions before the full development of flooding. However, sufficient flooding regime data is needed to use supervised learning for flooding prediction. This data imbalance problem is solved using Time-series generative adversarial networks (TimeGAN) to generate synthetic flooding regime data to balance the data.

In summary, three methods are considered for early flooding detection in the distillation column. PCA and Autoencoder are considered unsupervised ML methods, while pressure drop forecasting is considered an approach for different supervised ML methods. The overview of the research methodology is illustrated in Fig.3.1.

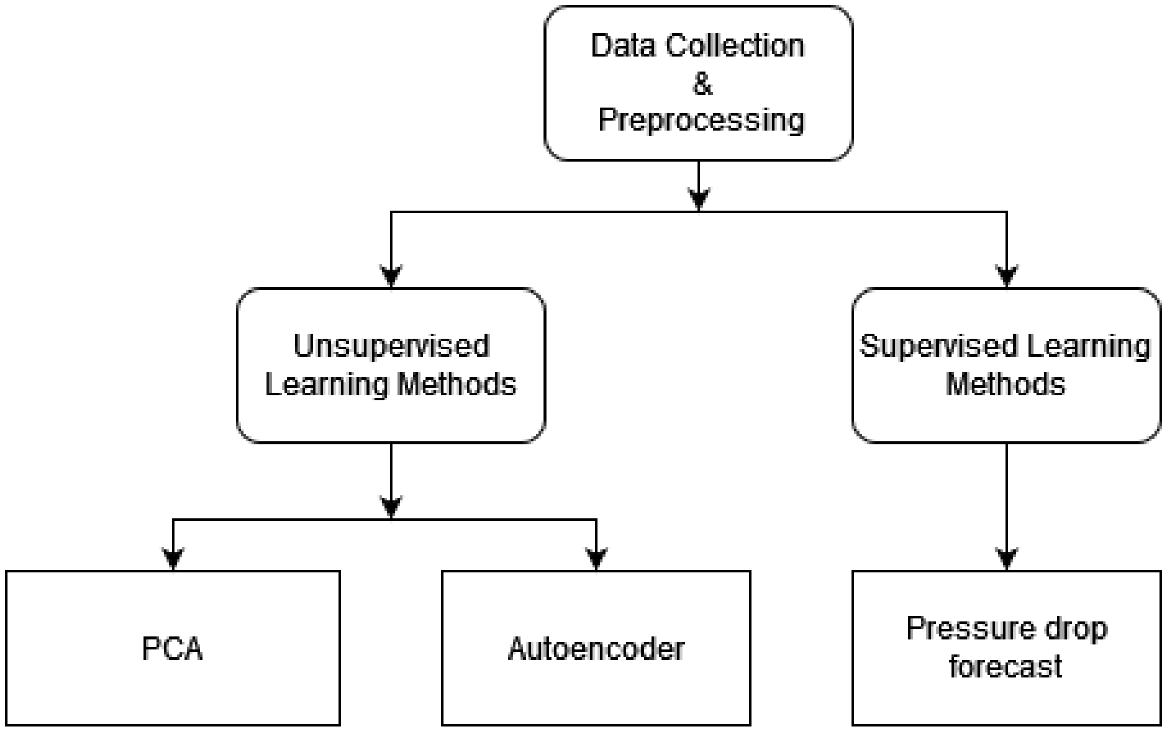


Figure 3.1: Overview of the research methodology for flooding detection using machine learning methods.

3.1 Rationale for Machine Learning Approaches

3.1.1 Challenges of Traditional Detection Method

Traditional flooding detection methods in distillation columns have commonly relied on static thresholds or physical observation, often based on factors such as liquid holdup or pressure differentials. While effective to an extent, these conventional techniques pose limitations, especially in real-time or dynamic operational environments where conditions may fluctuate rapidly. Static threshold-based approaches can struggle to adapt to variations in process parameters, leading to delays in detection and potential safety risks. Additionally, manual monitoring is labor-intensive and susceptible to human error, particularly in high-demand industrial settings. These limitations underscore the need for an automated, adaptive detection method capable of accurately identifying flooding events in real time.

3.1.2 Advantages of Machine Learning for Flooding Detection

ML offers unique advantages for flooding detection, addressing many of the constraints posed by traditional approaches. ML algorithms excel in recognizing complex patterns and correlations within large datasets, enabling them to detect early indicators of flooding that may not be evident through conventional methods. Unlike fixed thresholds, ML models can adapt to the specific operating conditions of a distillation column, learning from historical and real-time data to provide more reliable and timely predictions. This adaptability is crucial in industrial settings where variability in feed composition, temperature, and pressure can significantly impact process stability. By leveraging ML, the detection system can respond dynamically to changes, offering a robust and efficient solution for early flooding detection.

3.1.3 Addressing Data Imbalance through Machine Learning

A critical challenge in applying ML to flooding detection is the scarcity of flooding event data, leading to a significant class imbalance in the dataset. This imbalance hinders the performance of supervised learning algorithms, as models trained on imbalanced data may exhibit bias toward the majority class (non-flooding events), reducing their ability to correctly identify flooding occurrences. To address this issue, synthetic flooding data generation has been incorporated into the data preparation process, allowing for a balanced dataset that improves model training and performance. The addition of synthetic data enables the supervised ML model to learn from a wider variety of flooding scenarios, enhancing its capability to accurately detect rare flooding events. Through techniques such as using advanced methods like Generative Adversarial Networks (GANs) or data augmentation, the dataset can achieve a balanced representation of flooding and non-flooding cases, allowing the supervised ML model to deliver consistent, reliable detection.

3.1.4 Selection of Specific Machine Learning Algorithms

The selection of specific ML algorithms further contributes to the effectiveness of flooding detection in this study. Given the complexity of distillation column operations, models that handle regression tasks well—such as random forest, boosting algorithms, and neural networks are particularly suitable. Random forests, for example, are interpretable and perform well with large data, making them an excellent choice for initial testing and baseline comparisons. Neural networks, particularly deep learning models, offer powerful capabilities in feature extraction and complex pattern recognition, which are invaluable in identifying subtle trends and pre-flooding conditions. By choosing algorithms with proven robustness in handling complex patterns and time dependencies in data, this study ensures that the selected ML models can effectively manage the intricacies of flooding detection while maximizing accuracy and responsiveness.

3.2 Data Collection & Preprocessing

The first step is data collection and preprocessing. This involves the collection of data from the sensors of an industrial distillation column. These data give information about the mass and energy balances in the distillation column and are deemed important to the distillation process. The data must characterize the temperature measurements in the column trays, flow rates, differential pressure across the column, heat duties, and other necessary measurements. Data preprocessing is important for any data-based analysis to ensure that data truly represent the different events of the process. Plant shutdowns occur in the process industry for different reasons, maybe planned reasons, such as repairs and maintenance, or unplanned reasons, such as hazardous accidents or chemical spillage. In this analysis, the shutdown period of the plant is determined, and data related to this period are removed since they don't represent the dynamics of the plant during operation.

Also, missing data are filled using the forward fill method, which simply is a method for handling missing or incomplete data. The forward fill method involves carrying forward the last observed value to fill the missing gaps in the incomplete data. This filling method was used due to its ease of implementation and because there is little missing data in the considered features. These missing data are random and exist sporadically in the dataset.

An exponentially weighted moving average (EWMA) is applied to the data to ensure the data is free from sensor noise. The EWMA can be viewed as a dynamic control mechanism to help keep a process mean on target whenever discrete data are sequentially available [154]. It is also a popular method for estimating the long-term trend of a time series while giving more weight to recent observations [85]. It is implemented according to Eqn.3.1 below

$$\hat{y}_t = \alpha y_t + (1 - \alpha)\hat{y}_{t-1} \tag{3.1}$$

where $\hat{y}_t$ is the EWMA at time t, y_t is the observation at time t, and α is the smoothing parameter, typically a value between 0 and 1. It determines the weight of the current observation versus the previous EWMA value. A higher value of α gives more weight to recent observations, while a lower value gives more weight to historical observations, $\hat{y}_{t-1}$ is the previous EWMA value.

3.3 Flooding Detection using Unsupervised Learning Methods

In this distillation column, PCA and autoencoders are the unsupervised learning methods considered for early flooding detection.

3.3.1 Flooding Detection using PCA

3.3.1.1 PCA Formulation

Principal component analysis (PCA) is a linear dimensionality reduction technique that produces a lower-dimensional representation in a way that preserves the correlation structure between the process variables and is optimal in terms of capturing the variability in the data [155]. It determines a set of orthogonal vectors called loading vectors, ordered by the amount of variance explained in the loading vectors' direction. The formulation of PCA used in this work to detect flooding in a distillation column is described below.

Given a training set of n observations and m process variables stacked into a matrix X as shown below

$$X = \begin{bmatrix} x_{11} & x_{12} & .. & x_{1m} \\ x_{21} & x_{22} & .. & x_{2m} \\ : & : & .. & : \\ x_{n1} & x_{n2} & .. & x_{nm} \end{bmatrix} \tag{3.2}$$

The loading vectors are calculated by singular value decomposition (SVD) [156].

$$\frac{1}{\sqrt{n-1}} X = U \Sigma V^T \tag{3.3}$$

where $U \in \mathcal{R}^{n \times n}$ and $V \in \mathcal{R}^{m \times m}$ are unitary matrices, and the matrix $\Sigma \in \mathcal{R}^{n \times m}$ contains the non-negative real singular values of decreasing magnitude along its main diagonal ($\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$), and zero off-diagonal elements. The loading vectors are the orthonormal column vectors in the matrix V, and the variance of the training set projected along the i^{th} column of V is equal to σ_i^2. Solving Eqn 3.3 is equivalent to solving an eigenvalue decomposition of the sample covariance matrix S,

$$S = \frac{1}{\sqrt{n-1}} X^T X = V \wedge V^T \tag{3.4}$$

where the diagonal matrix $\wedge = \Sigma^T \Sigma \in \mathcal{R}^{m \times m}$ contains the non-negative real eigenvalues of decreasing magnitude ($\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$) and the i^{th} eigenvalue equals the square of the i^{th} singular value (i.e $\lambda_i = \sigma_i^2$).

In order to optimally capture the variations of the data while minimizing the effect of random noise corrupting the PCA representation, the loading vectors corresponding to the a largest singular values are typically retained. Selecting the columns of the loading matrix $P \in \mathcal{R}^{M \times a}$ to correspond to the loading vectors associated with the first a singular values, the projection of the observations in X into the lower-dimensional space is contained in the score matrix,

$$T = XP \tag{3.5}$$

And the projection of T back into the m-dimensional observation space,

$$\hat{X} = TP^T \tag{3.6}$$

The difference between X and $\hat{X}$ is the residual matrix E:

$$E = X - \hat{X} \tag{3.7}$$

The residual matrix captures the variations in the observation space spanned by the loading vectors associated with the $m - a$ smallest singular values. The subspace spanned by $\hat{X}$ and E are called the score space and residual space, respectively. The subspace in the matrix E has a small signal-to-noise ratio, and removing this space from X can produce a more accurate representation of the process $\hat{X}$.

Hence, the measured vector matrix X can be expressed using PCA as the sum of two orthogonal parts, approximated vector $\hat{X}$, and residual vector E as illustrated in Fig.3.2.

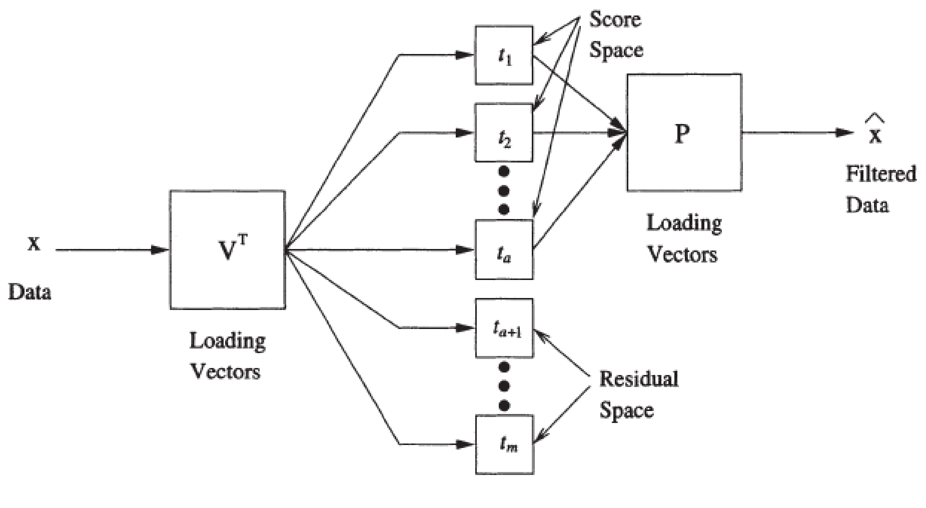


Figure 3.2: Representation of an observed data X as a sum of approximate and error part using PCA.

A new observation (column) vector in the testing set, $x \in \mathcal{R}^m$ can be projected into the lower-dimensional score space $t_i = x^T P_i$ where P_i is the i_{th} loading vector. The transformed variable t_i is also called the i_{th} principal component of x. To distinguish between the transformed variables and the transformed observation, the transformed variables will be called principal components, and the individual transformed observations will be called scores.

By appropriately determining the number of loading vectors a, to maintain the PCA model, several techniques exist for determining the value of the reduction order a. The method used in this research is the cumulative percent variance (CPV) test described by I. Karimi and K. Salahshoor [157]. This is due to its simplicity and wide applications. This method is described by the equation below.

$$CPV\% = \frac{\sum_{j=1}^{a} \lambda_j}{\sum_{j=1}^{m} \lambda_j} = \frac{\sum_{j=1}^{a} \lambda_j}{trace(\wedge)} \tag{3.8}$$

The value of a is selected to make the CPV larger than a pre-determined threshold. The threshold considered in this research is 95%.

3.3.1.2 Hotelling's T^2 Statistic

To detect faults (Such as flooding in a distillation column), Hotelling's T^2 and the squared prediction error (SPE) monitoring statistics are used to detect flooding as a fault for new measurements. By including in the matrix P the loading vectors associated only with the a largest singular values, the T^2 statistic for the lower dimensional space can be computed [21]

$$T^2 = x^T P \Sigma{-}2_a P^T x \tag{3.9}$$

where Σ_a contains the first a rows and columns of Σ. The T^2 statistic measures the variations in the score space only. The threshold for the T^2 statistic is given as

$$T^2_\alpha = \frac{a(n-1)(n+1)}{n(n-a)} F_{\alpha(\alpha,n-a)} \tag{3.10}$$

where a is the number of principal components, n is the number of measured observations, α is the confidence level, and $F_{\alpha(\alpha,n-a)}$ is the F distribution with the appropriate degrees of freedom and confidence level.

3.3.1.3 Squarred Prediction Error (SPE) or Q Statistic

The portion of the observation space corresponding to the $m - a$ smallest singular values can be monitored more robustly using the SPE or Q statistic. The SPE is a squared 2-norm measuring the deviation of the observations to the lower-dimensional PCA representation. It is computed below

$$\begin{aligned} r &= I - PP^T \\ SPE &= r^T r \end{aligned} \tag{3.11}$$

where r is the residual vector, a projection of the observation x into the residual space. The threshold of the SPE is computed below

$$Q_\alpha = \theta_1 \left[\frac{h_0 C_\alpha \sqrt{2\theta_2}}{\theta_1} + 1 + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2} \right]^{\frac{1}{h_0}} \tag{3.12}$$

where

$$\theta_i = \sum_{j=a+1}^{n} \sigma_j^{2i}, \; h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2} \tag{3.13}$$

C_α is the normal deviate corresponding to the $(1 - \alpha)$ percentile at a given significance level α.

3.3.1.4 PCA For Flooding Detection

PCA can detect flooding in a distillation column, where a PCA model is first constructed using process data obtained under normal operating conditions (non-flooding conditions). Then, the model is used along with one of the detection indices, such as T^2 or Q statistic, to detect flooding for new data samples (Flooding data) if the detection index falls outside the control limits, which are defined by the thresholds associated with these indices. Table 3.1 summarizes the procedures for this detection.

Table 3.1: PCA Steps for Flooding detection

```
1: Given:
   • A training flooding-free data set that represents the normal process operation
   • A testing data set (that contains flooding events)
   • A fixed false alarm probability $\alpha$
2: Data Preprocessing
   • Scale the data to zero mean and unit variance
3: Build PCA model using the training flooding-free data
   • Compute the covariance matrix $S$
   • Calculate the eigenvalues and eigenvectors of $S$
   • Sort the eigenvalues in decreasing order
   • Determine how many principal components to be used and obtain the loading vector $P$
   • Express the data matrix as a sum of approximate and residual
4: Compute the Flooding detection indices
   • Compute the $T^2$ statistic and the threshold $T^2_\alpha$
   • Compute the SPE or $Q$ statistic and the threshold $Q_\alpha$
5: Test the new data
   • Scale the flooding data
   • Express the data matrix as a sum of approximate and residual using the loading vector $P$
   • Compute the $T^2$ statistic and the $Q$ statistic
6: Check for flooding
   • if $T^2 \geq T^2_\alpha$, declare flooding
   • if $Q$ statistic $\geq Q_\alpha$, declare flooding
```

3.3.2 Flooding Detection using Autoencoder

An autoencoder is a self-supervisory neural network used to learn efficient codings of unlabelled data. It learns a representation for a set of input data by training the neural network to ignore insignificant data (i.e., often termed as "noise") [158]. A typical autoencoder comprises an input, output, and several hidden layers. An autoencoder consists of two parts: an encoder and a decoder, generally implemented by neural networks. The encoder and decoder can be viewed as two functions $h = f(x)$ and $\hat{x} = g(h)$, the $f(x)$ maps data point x from data space to feature space, while $g(h)$ produces a reconstruction of data point x by mapping h from feature space to data space [159]. The main goal of the autoencoder is to make the output vector similar to the original space by minimizing the reconstruction error between them. The operations of an autoencoder can be divided into Encoding, Decoding, and Reconstruction Loss as illustrated in Fig.3.3.

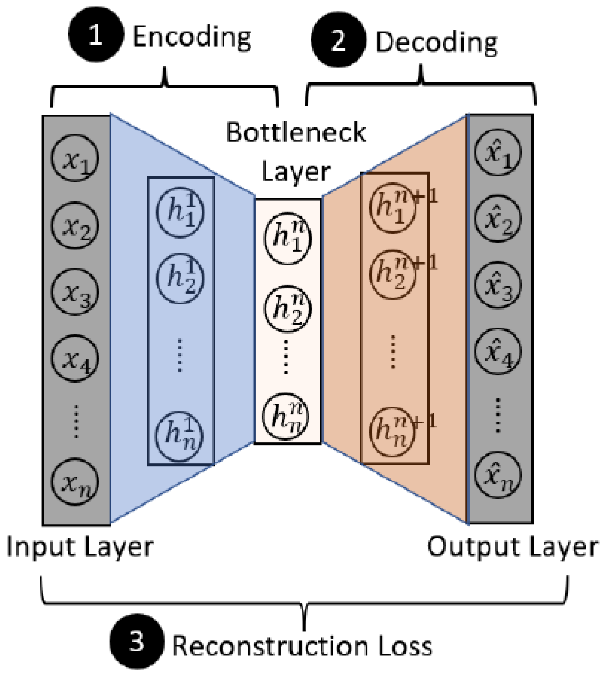


Figure 3.3: A typical illustration of an Autoencoder [158].

3.3.2.1 Encoding

In the encoding operation of an autoencoder, input data x is a m high dimensional vector (i.e., $x \in \mathcal{R}^m$) that is mapped to a low dimensional bottleneck layer representation h after removing any insignificant feature. This bottleneck layer is the feature space. In the encoding operation, the input vectors $x_i \in \mathcal{R}^m$ are compressed into d ($d < m$) numbers of neurons that make up the hidden layer. The activation of the neuron i in the hidden layer of the encoder is given by:

$$h_i = f_{\theta^e}(x) = f(\sum_{j=1}^{n} W_{ij}^{input} x_j + b_i^{input}) \tag{3.14}$$

where x is the input vector, θ^e is the parameters $\{W^{input}, b^{input}\}$, W is the encoder weight matrix with size $m \times d$, b is a bias vector of dimensionality d and f is the activation function. Hence, the input vector is encoded into a lower-dimensional vector.

3.3.2.2 Decoding

In the decoding operation, the bottleneck layer representation of h is used to generate the output $\hat{x}$ that maps back from the feature space into the reconstruction of x, i.e., the resulting hidden representation h_i is then decoded back to the original input space $\mathcal{R}^m$. The mapping function is as follows.

$$\hat{x}_i = g_{\theta^d}(x) = g(\sum_{j=1}^{n} W_{ij}^{hidden} h_j + b_i^{hidden}) \tag{3.15}$$

where W is the decoder weight matrix, b is the bias, g is the activation function of the decoder and $\hat{x}$ represent the reconstructed input sample. The parameter set of the decoder is $\theta^d = \{W^{hidden}, b^{hidden}\}$. it is important to note that W^{hidden} and b_{hidden} of the decoder maybe unrelated to the W^{input} and b_{input} of the encoder.

3.3.2.3 Reconstruction loss

The autoencoder is trained by minimizing a reconstruction loss. This loss is the difference between the output (reconstructed input) and the input, as shown below.

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} \|\hat{x} - x\|_2^2 \tag{3.16}$$

where $\theta = (\theta^e; \theta^d) = (W^{input}, b^{input}; W^{hidden}, b^{hidden})$. The above optimization problem can be solved using a gradient or stochastic gradient descent algorithm.

3.3.2.4 Autoencoder For Flooding detection

A fault or anomaly can be defined as an observation diverging from the majority of the data [158]. Flooding is a type of anomaly that can be detected in a similar way. As the normal data in the test dataset (Flooding dataset) meet the normal profile, which is built in the training phase of the autoencoder, the corresponding reconstruction error is smaller, whereas the anomalous data (flooding data) will have a relatively higher reconstruction error. As a result, by thresholding the reconstruction error, we can easily classify the flooding data:

$$D(x_i) = \begin{cases} \text{normal} & \text{for} \quad \epsilon_i < threshold \\ \text{flooding} & \text{for} \quad \epsilon_i \geq threshold \end{cases} \tag{3.17}$$

where $D(x_i)$ is the decision function to classify sample x_i as flooding or normal, ϵ_i is the reconstruction error of the $i^(th)$ sample. The threshold can be obtained by taking a standard deviation shift in the mean of the reconstruction loss for the normal operating dataset.

Autoencoder can be used to detect flooding in a distillation column by following the steps below.

Step 1: Data Preprocessing: Normalizing the data to prescale all the features into a specific interval is necessary. In this research work, data normalization was done according to the Eqn. 3.18 to convert all the features into the range of $[0, 1]$.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.18}$$

Step 2: Window Data Generation: This step is needed to create batches of time series data from the original dataset using a rolling or sliding window. For a dataset with sample size T, suppose that the number of increments between successive sliding windows is one period, then the entire dataset can be partitioned into $N = T–m + 1$ subsamples. The first rolling window contains observations for period 1 through m, the second rolling window contains observations for the period 2 through $m + 1$, and so on, as shown in Fig.3.4.

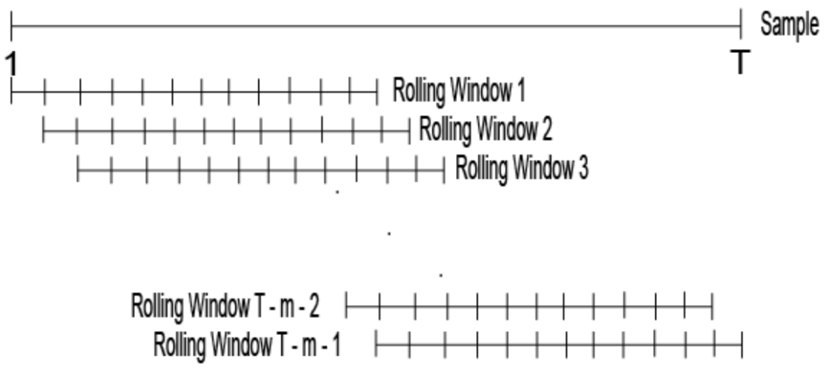


Figure 3.4: Window data generation for training the Autoencoder.

Step 3: Autoencoder Training: In the training phase, the normal operating dataset was used to train the autoencoder by minimizing the reconstruction loss. Therefore, by training the autoencoder, normal profiles of the operating data can be built.

Step 4: Setting threshold: After training the autoencoder, the error was obtained between the output (reconstructed input) and the input. In this work, a standard deviation above the mean of the reconstructed errors was set as the threshold.

Step 5: Flooding detection: In this step, the flooding dataset (testing dataset) was fed into the autoencoder model. The reconstruction error was utilized as the anomalous scores. Therefore, any data whose score is larger than the threshold will be determined as a flooding data sample based on the decision function in Eqn. 3.17.

The framework for flooding detection using autoencoder following the aforementioned steps is shown in Fig. 3.5.

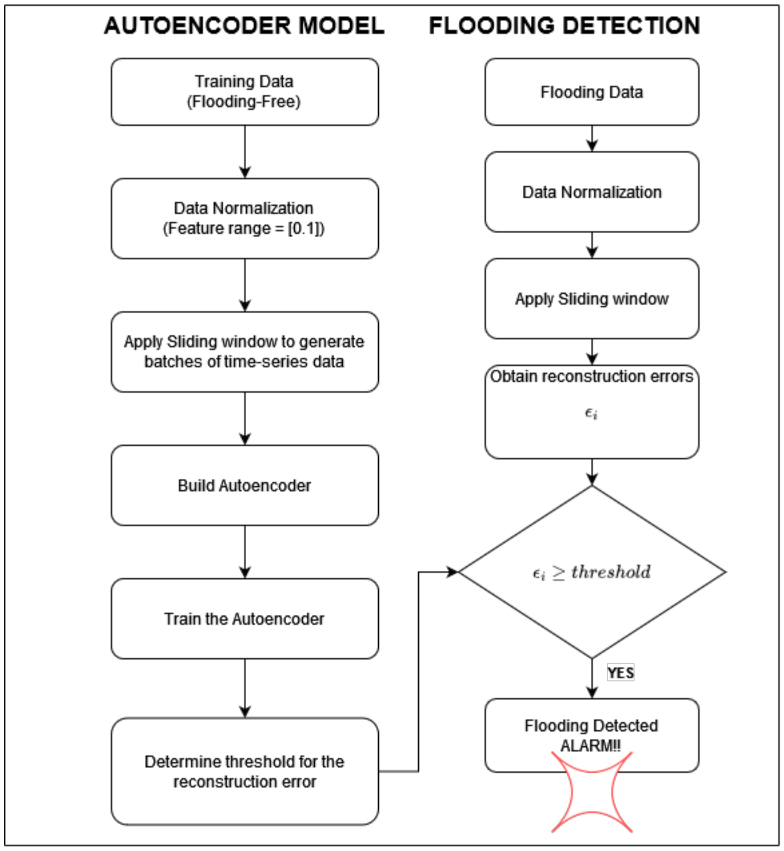

AUTOENCODER MODEL
FLOODING DETECTION
Training Data (Flooding-Free)
Data Normalization (Feature range = [0.1])
Apply Sliding window to generate batches of time-series data
Build Autoencoder
Train the Autoencoder
Determine threshold for the reconstruction error
Flooding Data
Data Normalization
Apply Sliding window
Obtain reconstruction errors ϵ_i
$\epsilon_i \geq threshold$
YES
Flooding Detected ALARM!!

Figure 3.5: A block diagram of Flooding detection using Autoencoder.

3.4 Flooding Detection using Supervised Learning Methods

The differential pressure across the distillation column is one of the important features to monitor for flooding. During flooding, there is a rapid rise in the pressure drop across the column. For early flooding detection in a distillation column, the ML models must accurately forecast the pressure drop for increasing and high-pressure drop cases in the column. This approach is divided into two parts: pressure drop forecasting and pressure drop classification, as indicated by the dotted sections in Fig.3.6. The pressure drop forecasting part deals with generating synthetic flooding data and training supervised learning methods to forecast the pressure drop across the column, while the pressure drop classification part deals with determining the operating state of the column using the historical pressure drop, i.e., grouping the pressure drop into normal operating state or flooding state. The mathematical formulations of TimeGAN and the supervised learning methods considered are already discussed in section 2.7 and section 2.6 of Chapter 2 of this thesis, respectively. The clustering algorithm (k-means) formulation is also discussed in section 2.6. The progression of the approaches used for early flooding detection using pressure drop forecast is summarized in the steps below:

Step 1: Data Preprocessing: To achieve this, the plant data are collected and pre-processed by filling up any missing data and are filtered using exponential weighted moving average (EWMA) according to Eqn. 3.1 to remove noisy signals.

Step 2: Feature Selection: The variables for the pressure drop forecast are chosen using a correlation plot and engineering judgment based on literature and the mode of operation of the distillation column. The correlation coefficient measures the degree of association between two variables. It is a statistical measure of the strength and direction of a relationship between two variables. For two variables X and Y, a sample correlation coefficient between them is given by the equation below.

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y}}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{3.19}$$

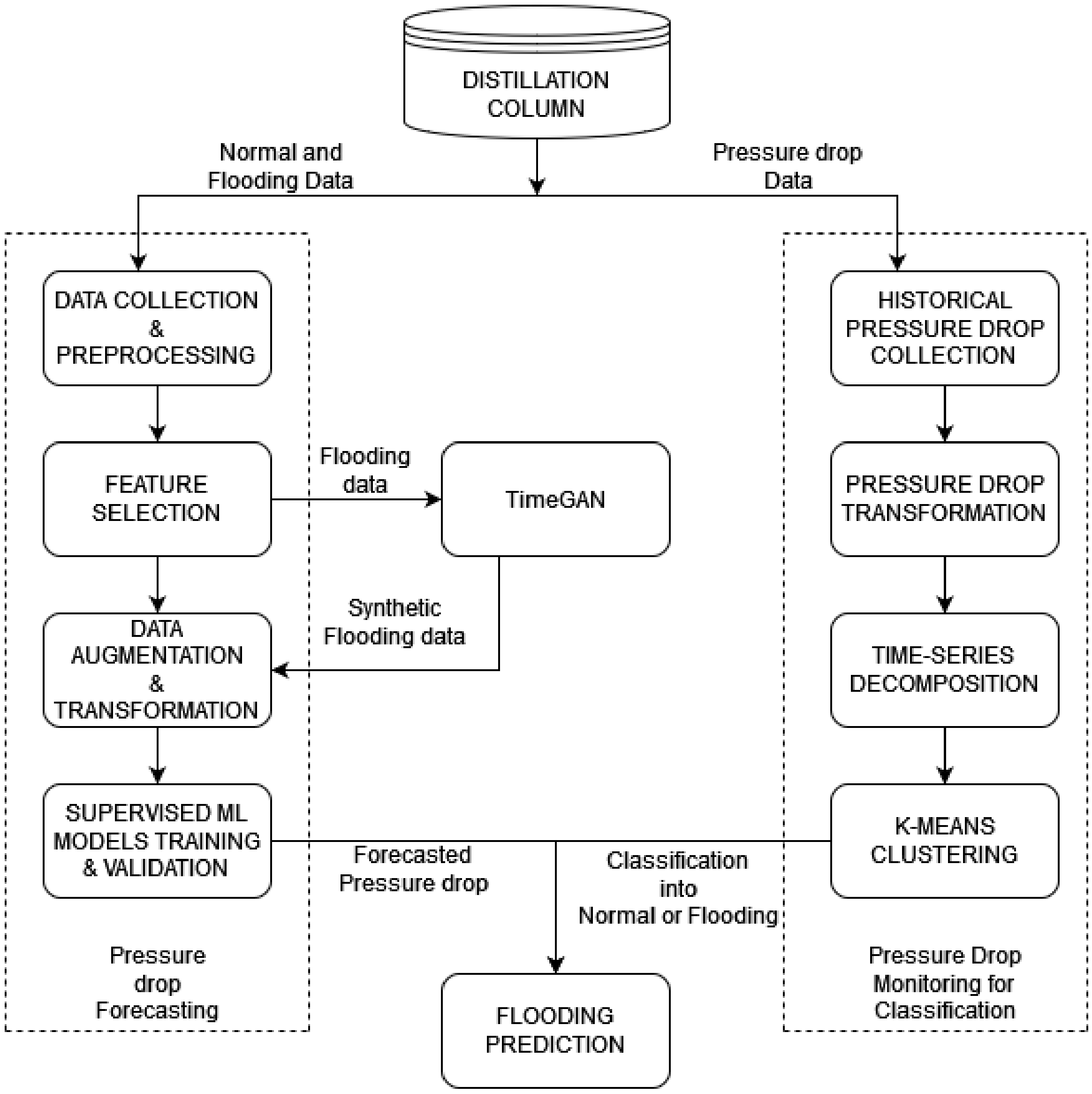

DISTILLATION COLUMN
Normal and Flooding Data
Pressure drop Data
DATA COLLECTION & PREPROCESSING
FEATURE SELECTION
Flooding data
TimeGAN
Synthetic Flooding data
DATA AUGMENTATION & TRANSFORMATION
SUPERVISED ML MODELS TRAINING & VALIDATION
Forecasted Pressure drop
Pressure drop Forecasting
HISTORICAL PRESSURE DROP COLLECTION
PRESSURE DROP TRANSFORMATION
TIME-SERIES DECOMPOSITION
K-MEANS CLUSTERING
Classification into Normal or Flooding
Pressure Drop Monitoring for Classification
FLOODING PREDICTION

Figure 3.6: A block diagram of flooding detection using supervised learning methods to forecast the pressure drop across the trays of a distillation column.

where x_i and y_i are individual data points, $\bar{x}$ and $\bar{y}$ are the means of variables X and Y, r_{xy} is the coefficient of correlation between X and Y. r_{xy} ranges from $+1$ to -1. When r_{xy} is $+1$, it indicates a perfect positive linear relationship between X and Y. When r_{xy} is -1, it indicates a perfect negative linear relationship between X and Y. When r_{xy} is 0, it indicates no linear relationship between X and Y.

Correlation analysis is used to select features with a strong linear relation with pressure drop, and such features are useful for forecasting the pressure drop across the distillation column. The correlation coefficient r_{xy} does not convey all there is to know about the association between two variables, as nonlinear associations can exist that are not revealed by this statistical measure. Hence, engineering judgment based on literature coupled with the mode of operation of the distillation column is used to also select features that have an effect on the pressure drop. This approach is sufficient to select variables that can be used to forecast pressure drop in the distillation column.

Step 3: Synthetic Flooding Data Generation: Training ML models for flooding prediction requires sufficient flooding regime data so that the models can capture the dynamics of the column before flooding. For this case of insufficient flooding regime data, synthetic flooding data are generated using time-series generative adversarial networks (TimeGAN), a framework proven effective in generating virtual time series data samples. The implementation of TimeGAN is discussed in Section 2.7 of this thesis. The TimeGAN is trained to generate more flooding regime data to augment the original dataset.

Step 4: Data Transformation: To preserve the temporal dynamics of the multivariate time series data during training, supervised learning methods can be applied to time series data by transforming the data using sliding window method [160]. This is a way of framing time series forecasting as a supervised learning problem. The sliding window method transforms the data by preserving the past and future measurements for each data point, making the data points independent and identically distributed (iid). For this case, the future pressure drop will be predicted based on the past pressure drop data and other selected parameters according to Eqn. 3.20.

$$(\Delta p_{t+1}, \Delta p_{t+2}, \ldots) = f(\ldots, \Delta p_{t-1}, \Delta X_{i,t-1}, \ldots, \Delta p_t, \Delta X_{i,t}, \ldots) \tag{3.20}$$

A transformation like this requires selecting two parameters: window size and response size. The window size is the time window of the past data, and the response size is the forecast window of the data. The application of this transformation results in many additional columns to the dataset.

Step 5: Training Supervised Learning Models: Different ML methods are then trained to forecast the pressure drop of the column. The performance of these models can be known using error metrics such as mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2) of the model.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y_i}|}{n} \tag{3.21}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y_i})^2}{n}} \tag{3.22}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{3.23}$$

where y_i is the ith observed variable, $\hat{y_i}$ is the ith predicted variable, n is the total number of observed samples, $\bar{y}$ is the mean of the observed variable.

Step 6: Determination of Operating States: Forecasting the pressure drop alone is insufficient for early flooding detection. Hence, pressure drop monitoring uses a clustering algorithm to classify the historical pressure drop into operating states (normal or flooding). This is done by transforming the historical times series pressure drop data using Eqn.3.20. This made each sample of the transformed pressure drop independent and identically distributed because each sample contains past, present, and future pressure drop values. Time series data can be decomposed into four features: trend, noise, seasonality, and level. Due to the absence of noise in the transformed pressure drop data and flooding not occurring in specific intervals (seasonality), only trend and level can be extracted from the transformed pressure drop to ensure good visualization and interpretability of the clusters. The two chosen features are represented by slope (trend: increase or decrease in the data point) and median (level of the data point). The clustering algorithm is then applied to the extracted features (slope and mean) of the transformed pressure drop data for classification into normal or flooding state. This trained clustering algorithm is used on the forecasted pressure drop to classify the pressure drop into a normal or flooding operating state for early warning of possible flooding in the distillation column.

Chapter 4

Results and Discussion

4.1 Case Study and Data Collection

The distillation column studied in this work is an industrial debutanizer that takes the bottoms of de-ethanizer as feed to separate C_3/C_4s from naphtha range materials. Plant data are collected from the sensors of the debutanizer. The data collected are available every minute and contain 46 variables, such as the pressure drop across the column, heat duties, feed flow, feed temperature, etc. For this case study, $525,000$ sample points are collected with less than 1% as flooding regime data (the rest are normal operation data). This leads to a case of data imbalance, with flooding data being the minority.

4.2 Data Preprocessing

The collected data are preprocessed by filling up missing data using forward fill method. Fig.4.1 and Fig.4.2 show the plot of some features collected from the sensors of the distillation column.

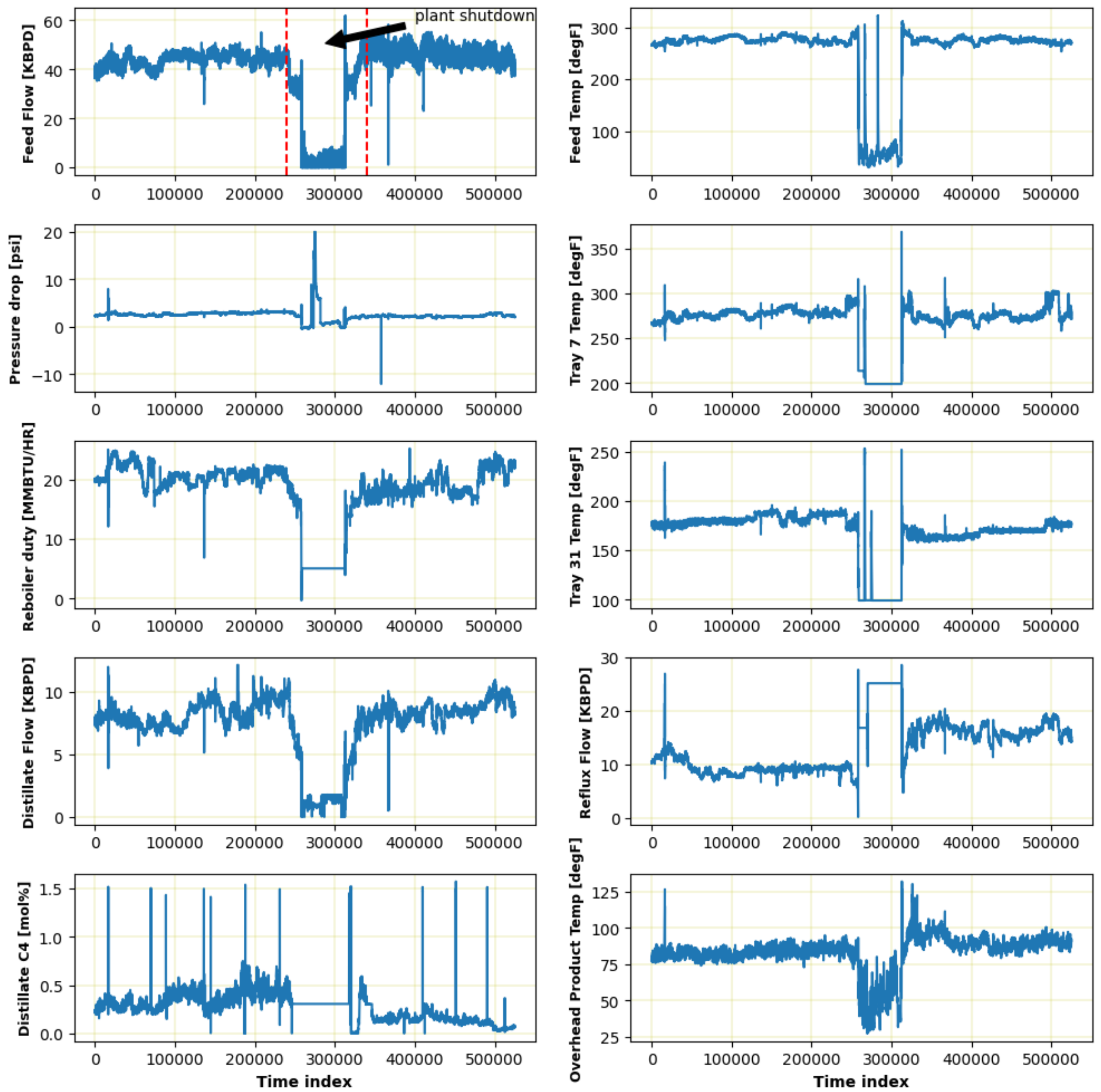


Figure 4.1: Plot of some features collected from the sensors of the industrial distillation column. The horizontal axis corresponds to the index of each time-series sample of the data.

From the observed plots, it can be seen that the feed flow to the distillation column reduced to 0 kbpd at around 240,000 till 340,000 sample points. This reduction in feed flow results from plant shutdown during the affected period. The plant shutdown period data were removed since operational information can not be gotten from such data. Only data samples that describe the operational dynamics of the distillation column are considered.

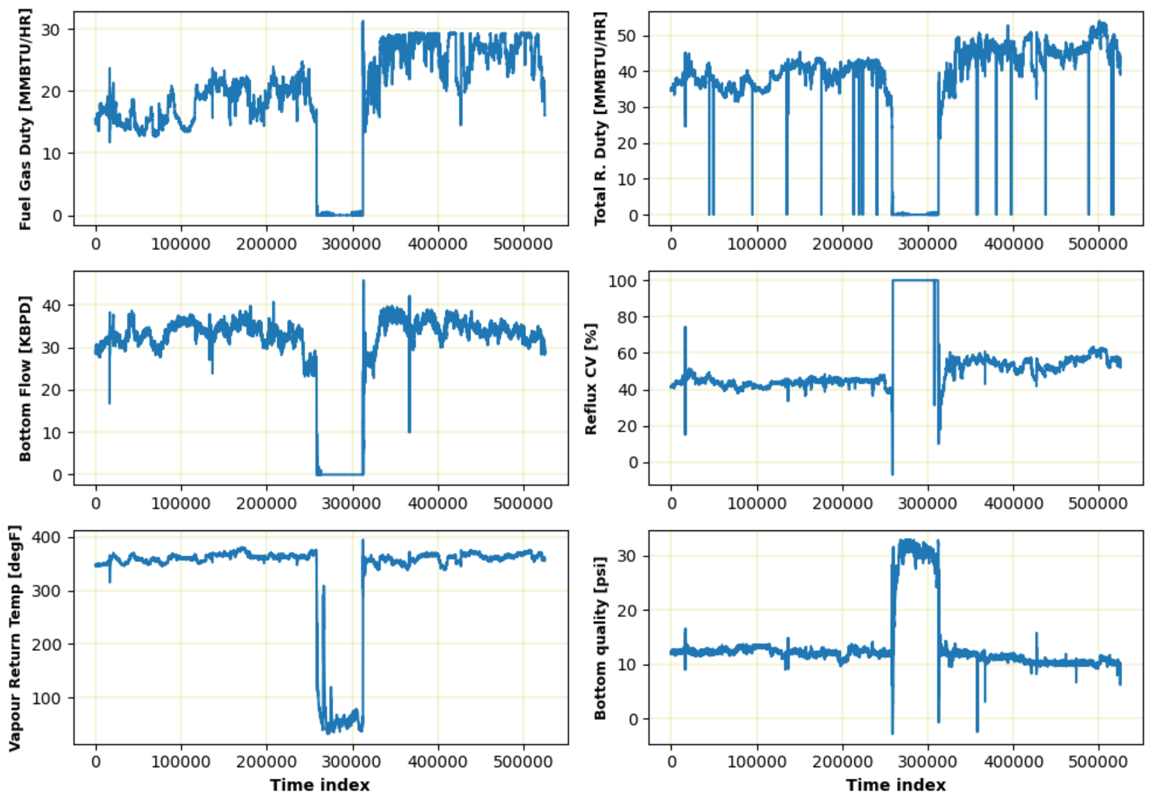


Figure 4.2: Plot of some features collected from the sensors of the industrial distillation column. The horizontal axis corresponds to the index of each time-series sample of the data.

EWMA was applied to the data based on Eqn. 3.1 to remove sensor noise from the collected data. To choose the value of α for the EWMA, it is important to obtain the metrics of different values of α and their effect on the collected data. Signal-to-noise ratio (SNR) and mean squared error (MSE) are used for this evaluation. SNR is the measured strength of the desired signal relative to noise; it is measured in decibels (dB). The smaller the MSE, the greater the SNR and the better the denoising effect [161].

$$SNR = 10 \log \left(\frac{\sum_{n=1}^{N} f(n)^2}{\sum_{n=1}^{N} [f(n) - \hat{f}(n)]^2} \right) \tag{4.1}$$

$$MSE = \frac{\sqrt{\sum_{n=1}^{N}[f(n) - \hat{f}(n)]^2}}{N} \tag{4.2}$$

where $f(n)$ is the signal containing noise, $\hat{f}(n)$ is the denoised signal, N is the length of the signal.

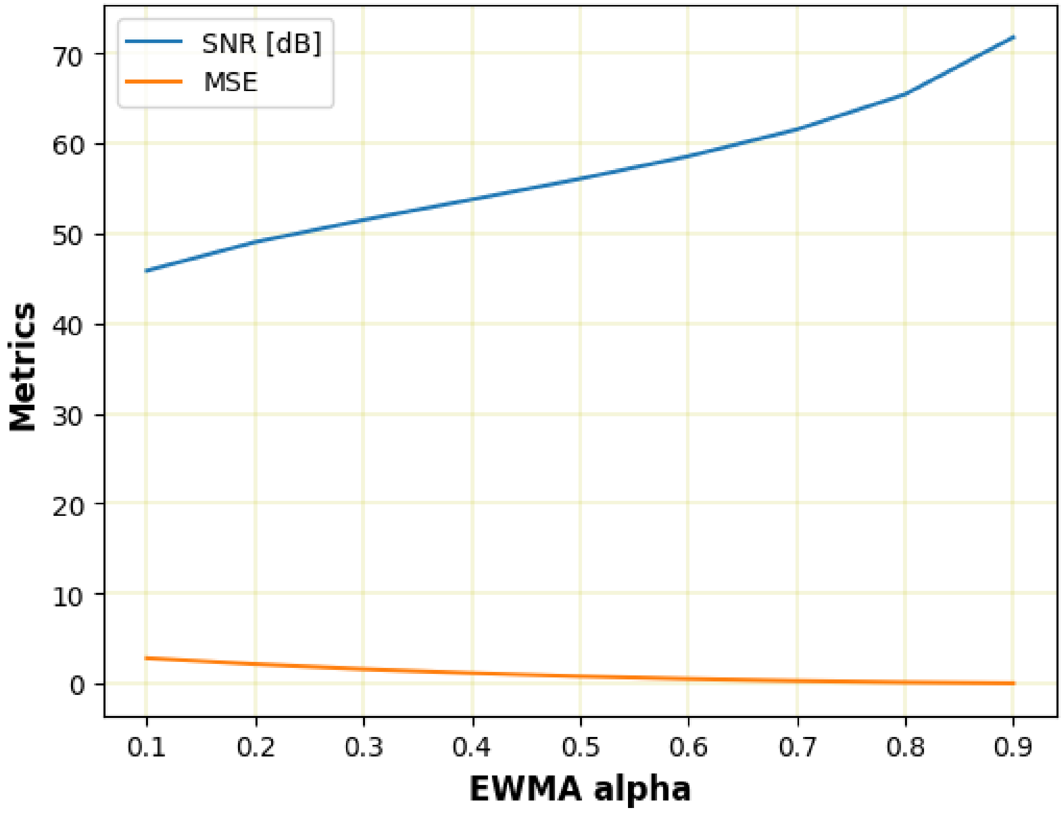


Figure 4.3: Metrics evaluation for denoising the data.

Fig.4.3 shows the plot of SNR and MSE for different values of α for the EWMA. It can be observed from the plot that as the SNR increases for the α, the MSE reduces. Since EWMA can be used to estimate the long-term trend of a time series of data while giving more weight to recent measurements or observations, α of 8 is selected for denoising the data as a trade-off between the noise level in the data and maintaining the dynamics while denoising the data.

Some of the features after denoising are visualized in Fig.4.4 and Fig.4.5. For easy visualization, the effect of the EMWA is shown for the first 500 sample points of some features of the data collected from the distillation column.

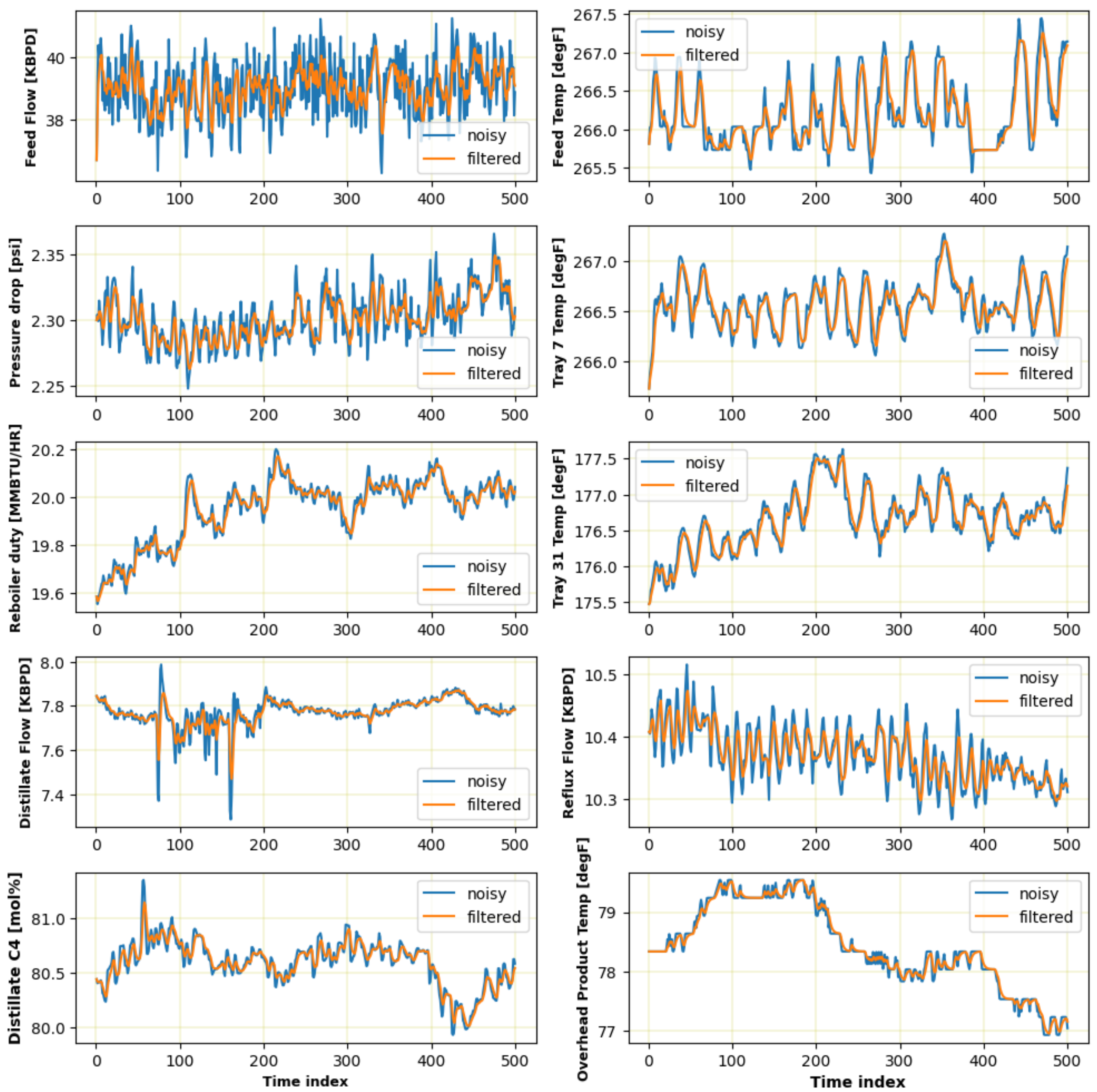


Figure 4.4: Sample plots of some denoised features of the collected data. The horizontal axis corresponds to the index of each time-series sample of the data.

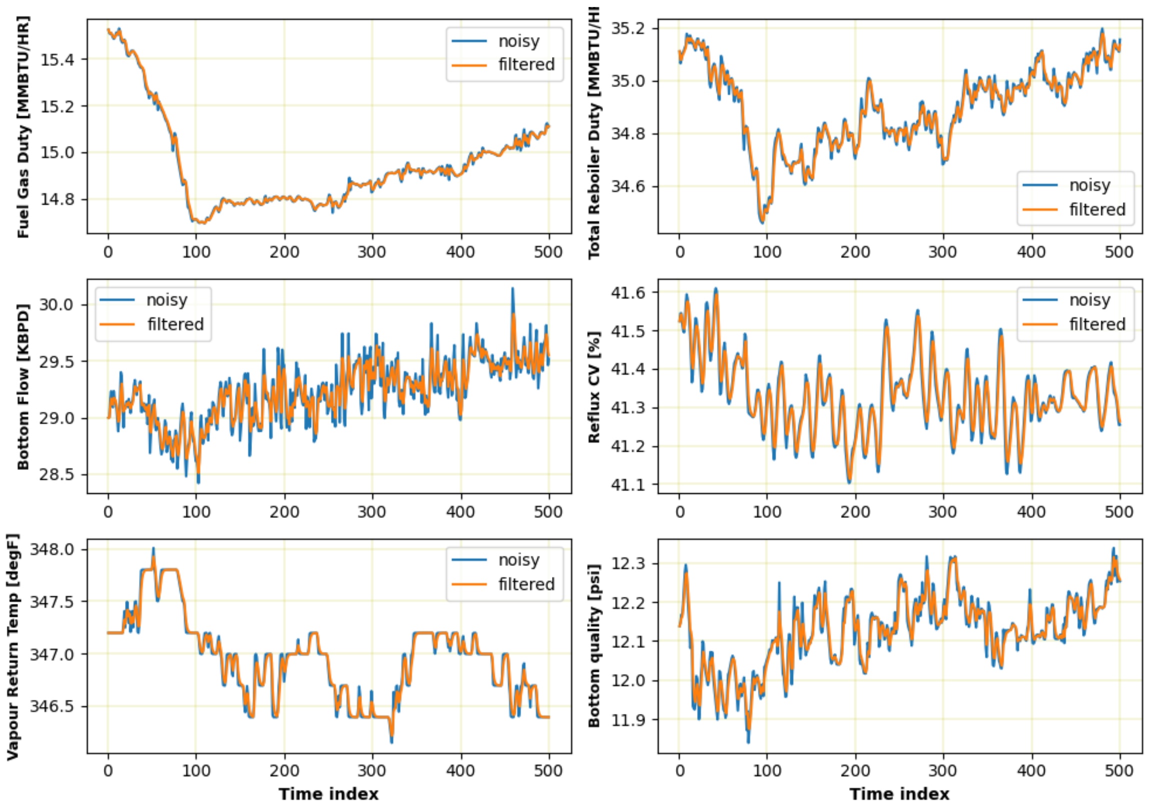


Figure 4.5: Sample plots of some denoised features of the collected data. The horizontal axis corresponds to the index of each time-series sample of the data.

4.3 Flooding in the Distillation Column

The concept of flooding can be visualized from a major flooding event that occurred in the distillation column. Fig.4.6 shows some important features during a major flooding event in the distillation column. The pressure drop across the column is being used to monitor flooding events. Normal operating conditions within the column are for pressure drop below 3 psi ($dp < 3$) while flooding becomes noticeable when the pressure drop is above 3.5 psi ($dp > 3.5$).

From Fig.4.6, the pressure drop deviated from the normal operating range at 9:26 on the day of a major flooding event, and flooding became noticeable at 10:20. The increase in the vaporization rate of the reboiler caused more vapor to be boiled up, this increases the vapor flow upward the column causing the pressure drop to rise as a result of entrainment of liquid in the rectifying section of the distillation column.

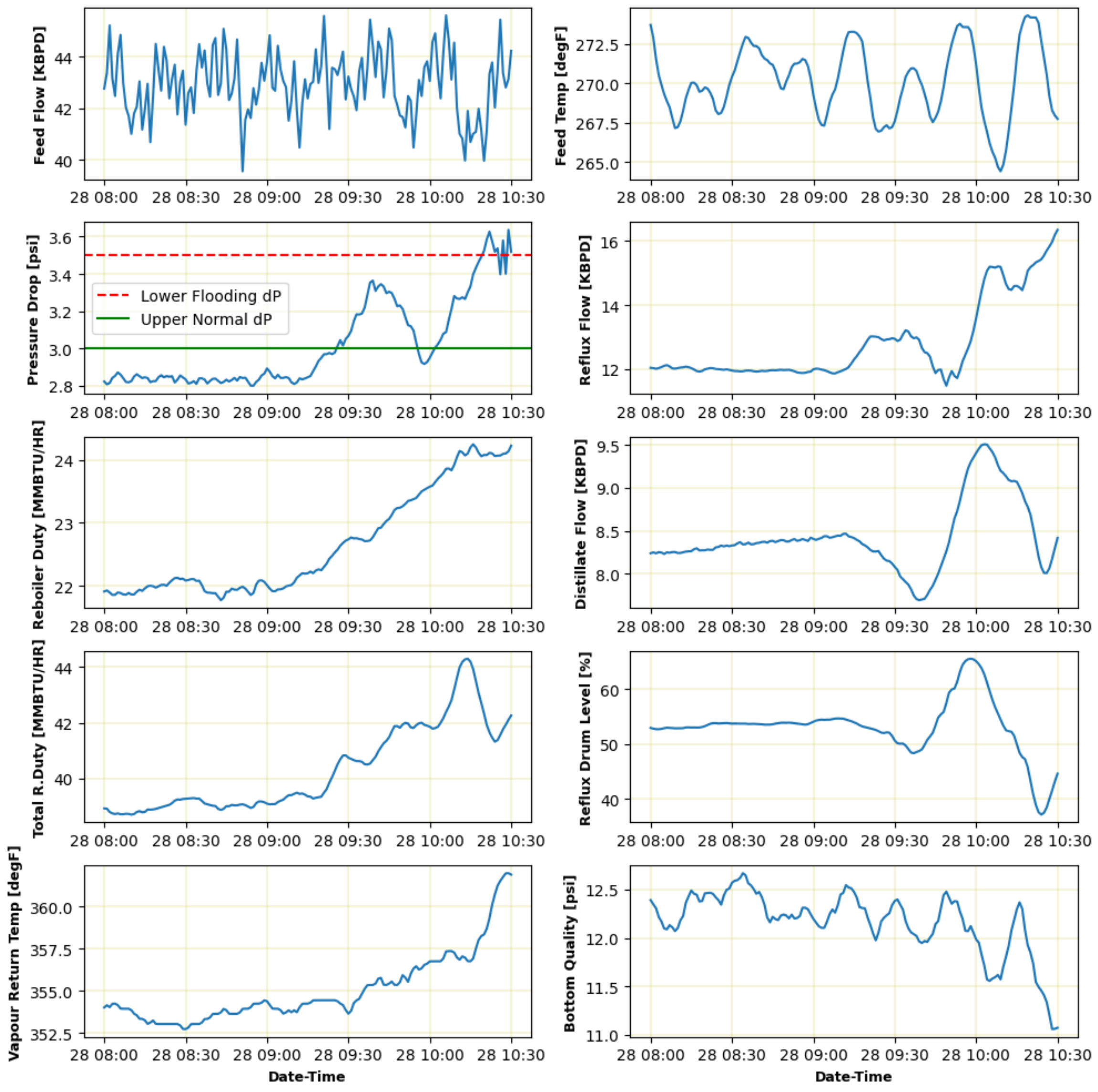


Figure 4.6: Plot of some features in the distillation column during a flooding event. The horizontal axis corresponds to the day and time of the each time-series data sample.

This entrainment caused an irregularity in the distillate flow, which led to an increase in the distillate flow for some minutes before encountering a significant decrease in the flow of distillate, leading to a significant rise in the reflux flow back to the column, causing accumulation of liquid in the column. This effect can be seen in the decline in the quality of the bottom product. The continuous high vaporization rate of the reboiler causes a further spike in the pressure drop, as seen in Fig.4.7. At this point, there is an overflow in the reflux drum level (maintained 100% level) due to the liquid accumulation in the column.

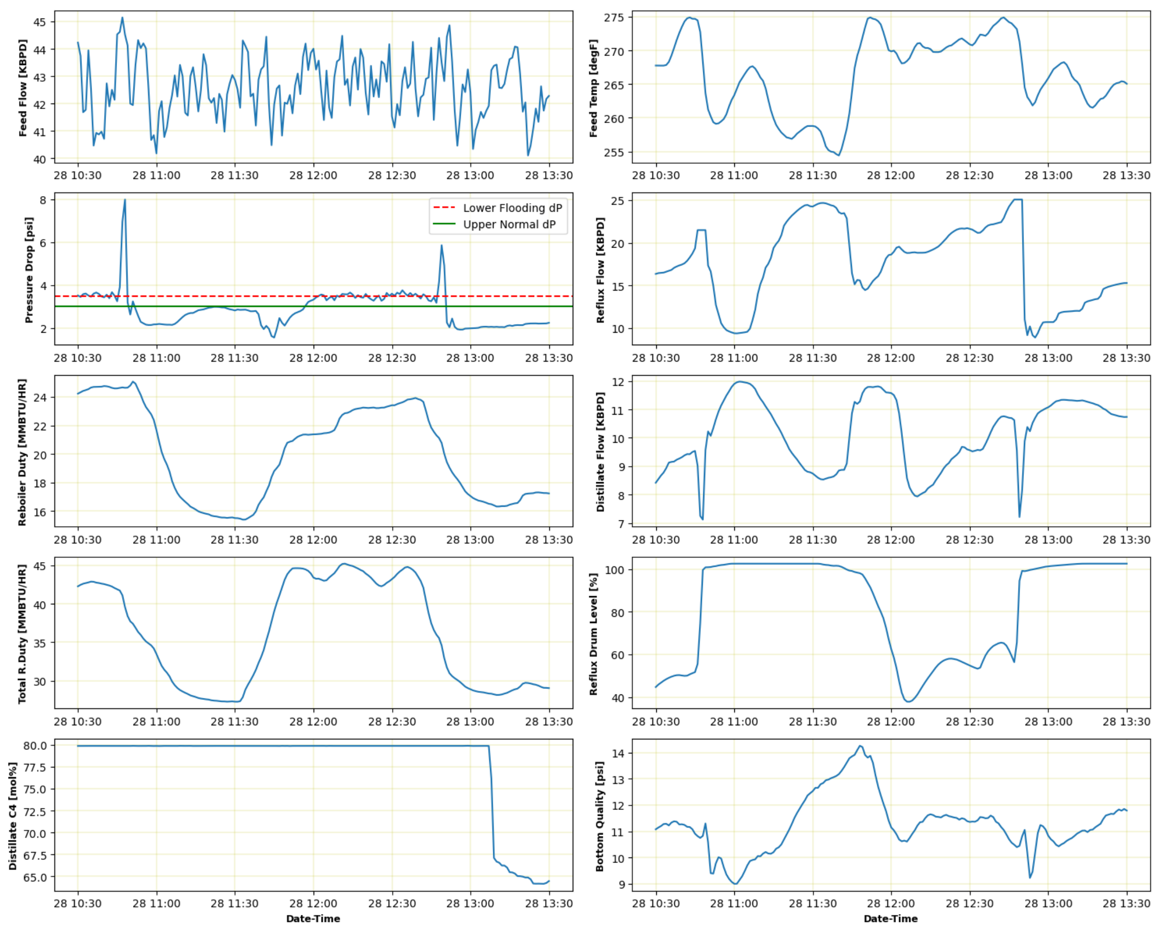


Figure 4.7: Plot of some features in the distillation column during a flooding event. The horizontal axis corresponds to the day and time of the each time-series data sample.

Further decrease is noticeable in the quality of the bottom product until there is a reduction in the heat duty of the reboiler, after which the quality of the bottom product improves. The effect of this flooding event is later seen on the distillate product quality at 13:05 in Fig.4.7 when there is a decrease in the quality of distillate product C_4.

This cause-and-effect relationship can be represented with a simple signed directed graph as shown in Fig.4.8. An increase in the reboiler duty (RD) leads to a high vaporization rate (VFR) in the column, subsequently causing liquid accumulation

(LA). This accumulation of liquid leads to an occurrence of flooding (FO). Also, this caused the differential pressure (PD) across the column to rise. Accumulation of liquid in the column leads to a high level in the reflux drum (RDL), which prompts more liquid to be returned to the column, which is an indication of high reflux flowrate (RF) and, at the same time, a decrease in the distillate flowrate (DFR) causing the decrease in the distillate product quality at the long run. This occurrence leads to a decrease in the bottom product quality (BPQ) in the short run, as noticed in the collected data.

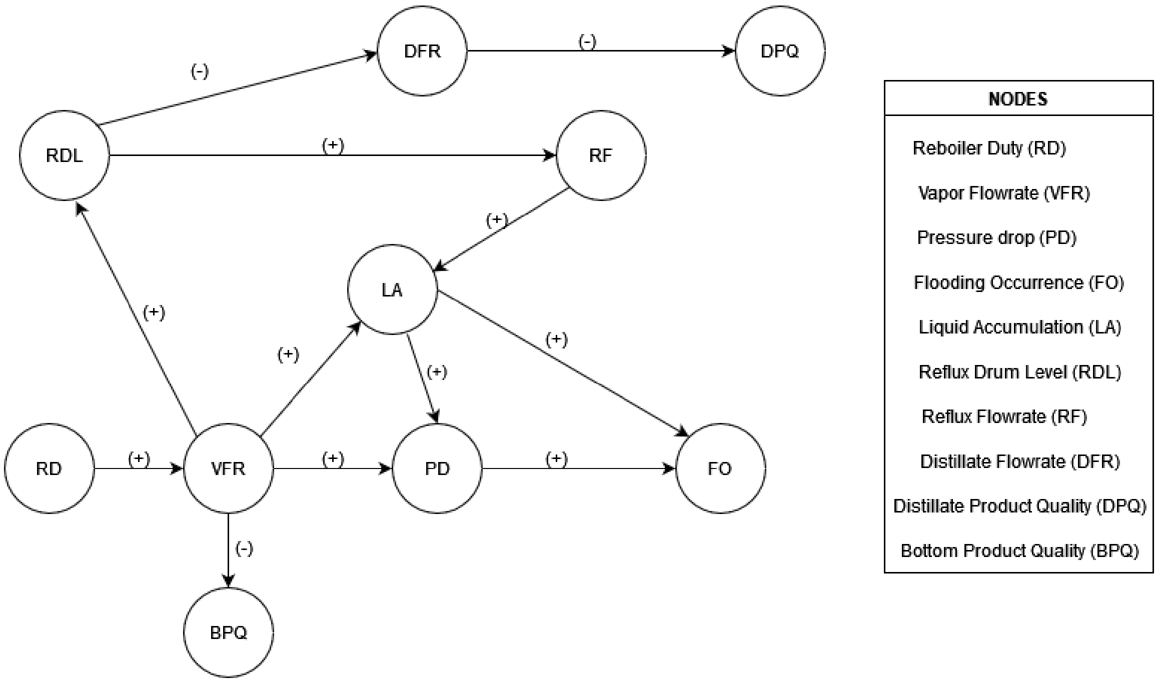


Figure 4.8: A signed directed graph illustrating the cause and effect of flooding occurrence in the distillation column.

4.4 Flooding Detection using PCA

A PCA model was developed for flooding detection in the distillation column using the sample data collected at the nominal operating conditions. All variables collected are used for the PCA model since the results obtained using all variables are better than selecting a few variables for the PCA analysis. Hence, feature selection wasn't done for PCA. The data collected was scaled to zero mean and unit variance. 16 principal components account for 95.36% variation in the original data and are sufficient to build a PCA model, as shown in Fig.4.9. The monitoring charts for the PCA model developed based on the nominal data, the T^2 statistic plot, and the Q statistic or SPE plot obtained for the model at the nominal operating conditions are shown in Fig.4.10.

The control limits were obtained based on 99% confidence interval. It can be seen from Fig.4.9 that the monitoring statistics for the PCA model (T^2 statistic and Q statistic) resulted in some false alarms for the training data. Hence, for flooding detection, flooding is declared after the violation of the control limits for continuous successive sampling time to reduce the occurrence of false alarms.

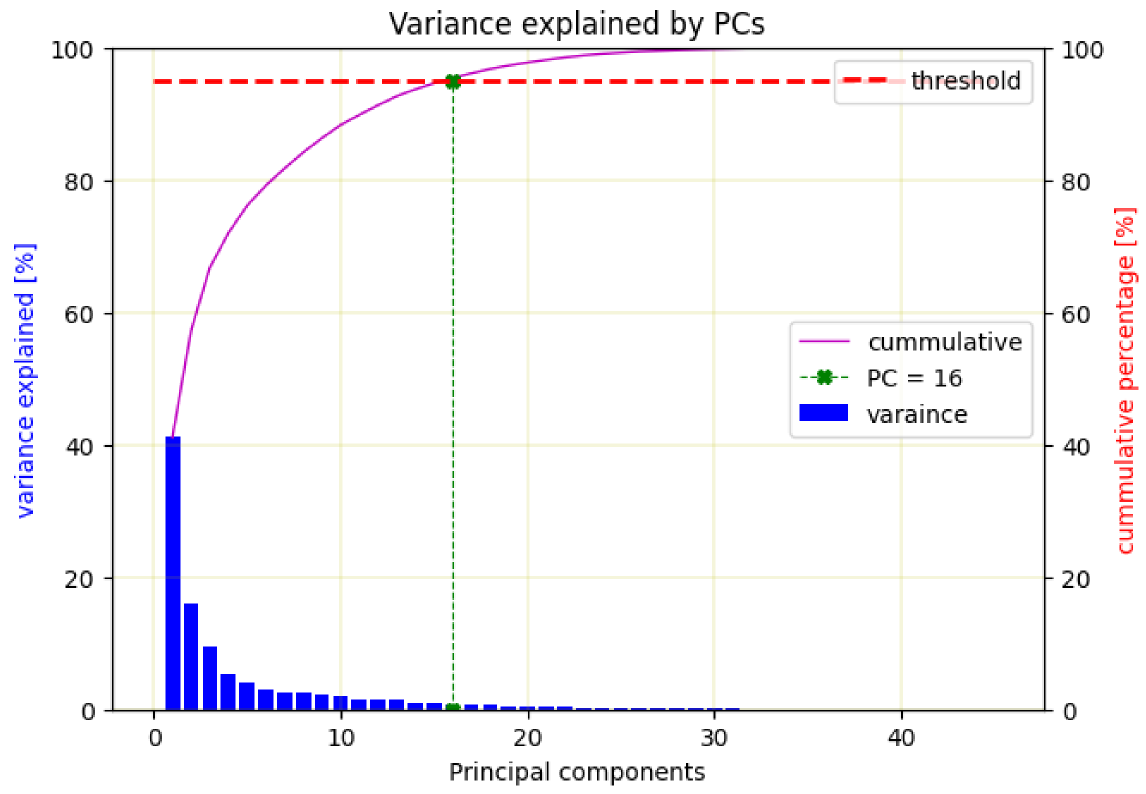


Figure 4.9: Determination of the principal components using cumulative percent-variant test.

The flooding data of a major flooding event in the distillation column is used on the trained PCA model to detect flooding. The flooding data is scaled using the mean and variance of the normal operating data to capture the deviation in the variance using the trained PCA model. The T^2 statistic and the Q statistic of the PCA model for the flooding data are shown in Fig.4.11 and Fig.4.12, respectively.

Flooding was detected using the T^2 statistic at 9:50 on the day of a major flooding event as shown in Fig.4.11 and at 9:20 using the Q statistic as shown in Fig.4.12. The Q statistic detected flooding earlier compared to the T^2 statistic. Based on the flooding data, the buildup of flooding started at 9:40, and flooding became noticeable and pronounced at 10:21 on the day of a major flooding event. Hence, the Q statistic of the PCA model detected flooding 10 minutes earlier before the buildup of flooding and 51 minutes before flooding became pronounced in the distillation column. This is the earliest detection of flooding using PCA.

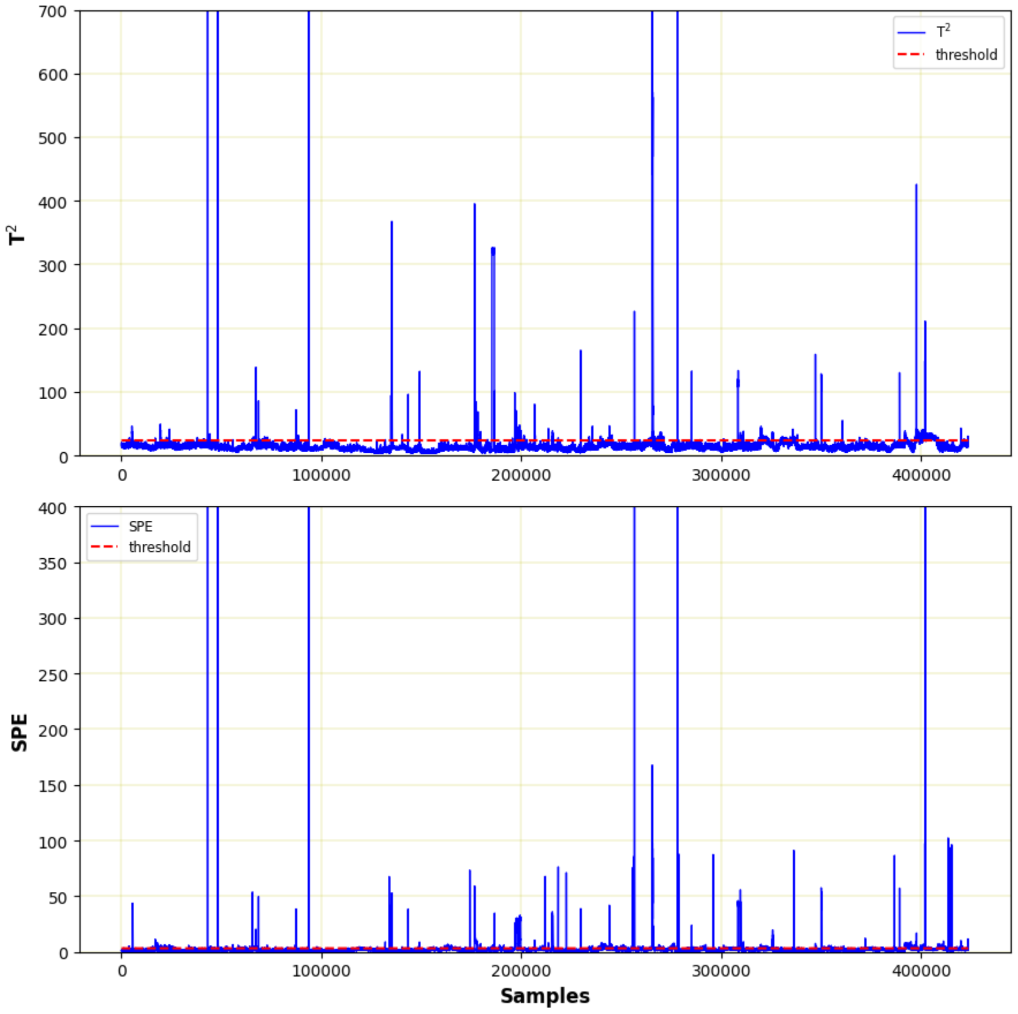


Figure 4.10: PCA monitoring chart for training (nominal operating) data.

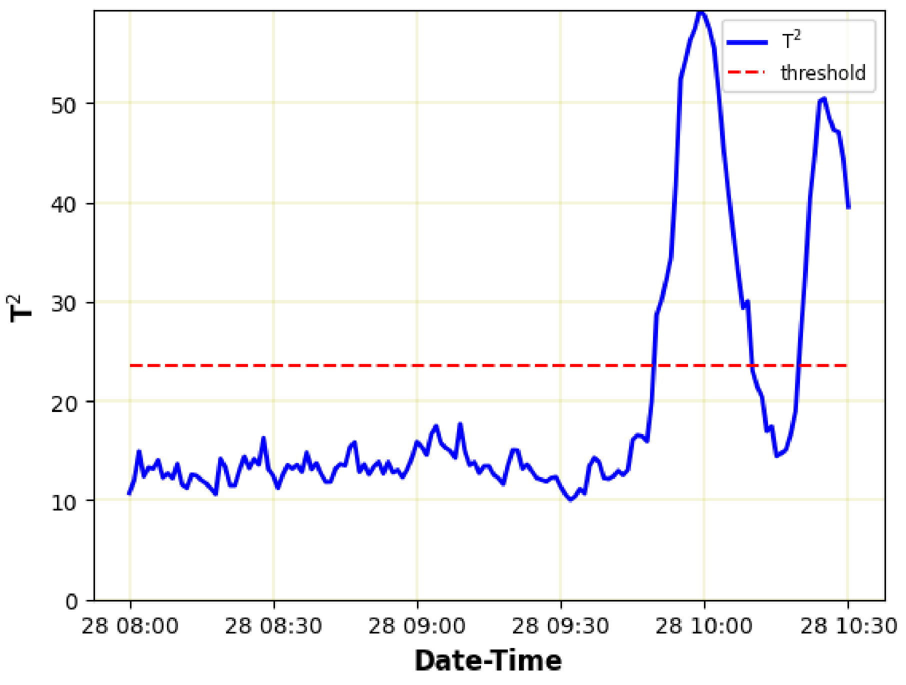


Figure 4.11: Flooding detection using T^2 statistic on the testing (flooding) data. The horizontal axis indicates the day and time of each time-series data sample.

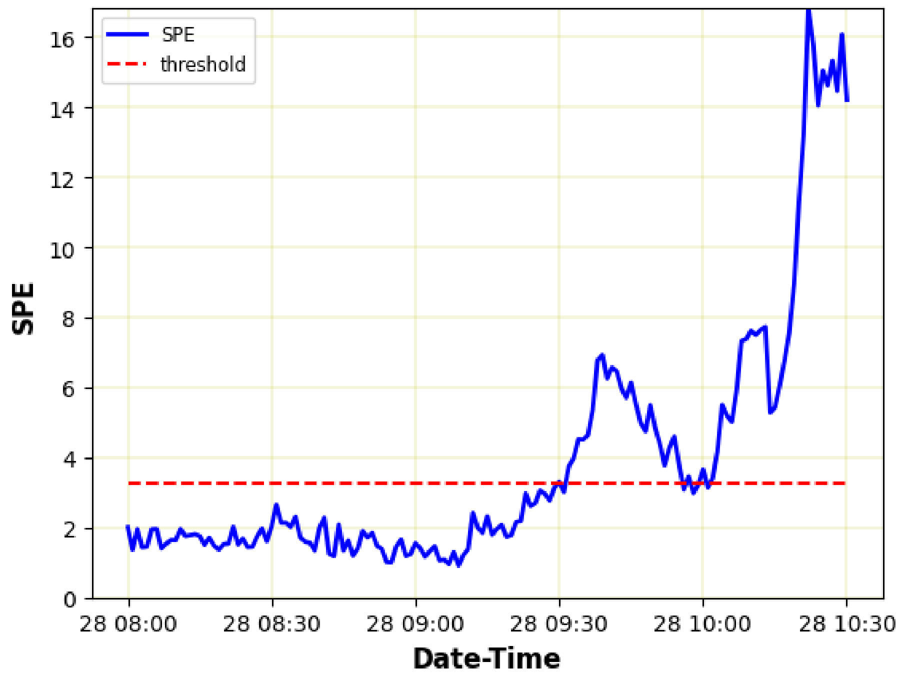


Figure 4.12: Flooding detection using Q statistic on the testing (flooding). The horizontal axis indicates the day and time of each time-series data sample.

The contribution plot of the PCA model based on the time of detection of the Q statistic is shown in Fig.4.13. This shows how each feature contributed to the monitoring statistics at the point of detection. This point of detection is at 9:30 on the day of a major flooding event as detected by the Q statistic.

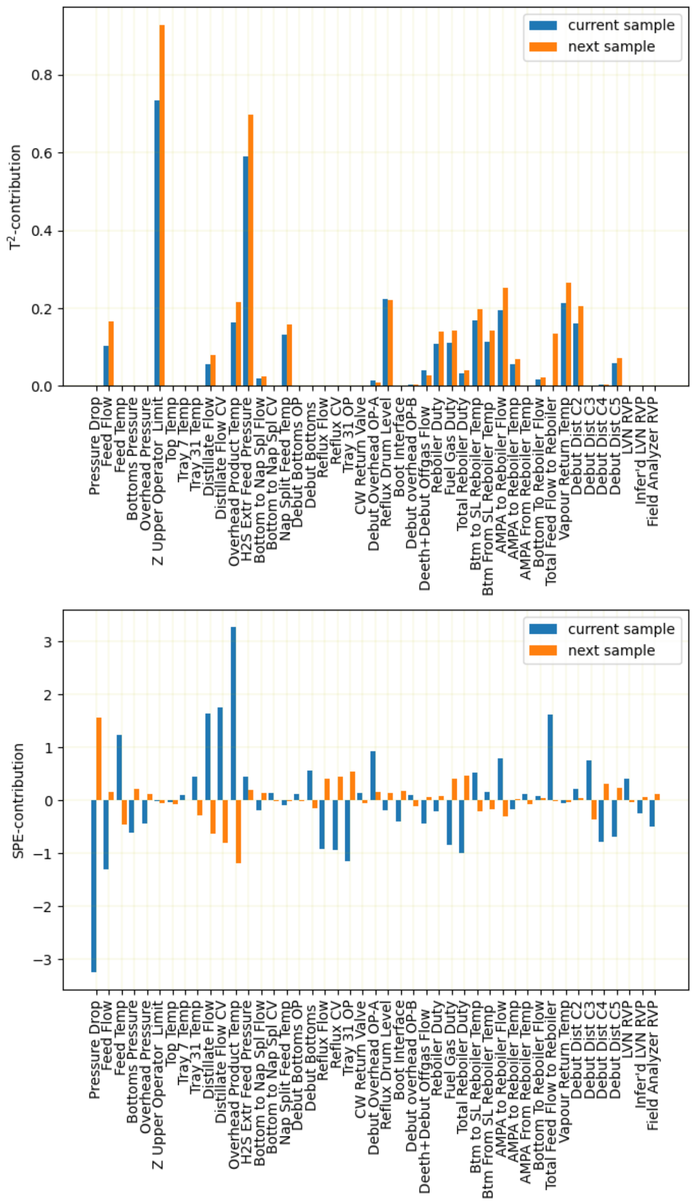


Figure 4.13: Contribution of each feature of the column data to PCA monitoring statistics at the time of detection (9:30) on the day of a major flooding event in the distillation column.

The pressure across the column (‘Pressure Drop’), the temperature of the overhead product (‘Overhead Product Temp’), flow of feed to the reboiler (’Feed Flow To Reboiler’), flow of the distillate (‘Distillate Flow’), Feed Flow to the column (‘Feed Flow’) and temperature of the feed to the column (‘Feed Temp’) contributed significantly to the Q statistic at this point of flooding detection. This also gives an insight into some of the variables that contributed significantly to the flooding occurrence. Some of these variables are in agreement with the cause-and-effect analysis given in section 4.3. These variables are considered during feature selection for supervised learning methods.

4.5 Flooding Detection using Autoencoders

4.5.1 Detection using Conventional Autoencoder

A conventional Autoencoder is built for flooding detection using neural network layers. Due to a lot of hyperparameters in an autoencoder, such as the number of neurons in the layers, activation function, learning rate, batch size, etc. A randomized grid search of these hyperparameters was conducted using k-fold cross-validation with k=3 by utilizing the scikit-learn library. The autoencoder is built with the combination of hyperparameters, which gave the best result. The encoding layers of the autoencoder contain 3 hidden layers of 32, 24, and 20 neurons, respectively, which are used to encode features of the normal operating data from a higher dimension of 45 features to a lower dimension of 20 encoded features. Also, a decoder is built to reconstruct the original inputs (45 features) from the low dimensional encoded features (20 features) using 2 hidden layers of 24 and 32 neurons and then the output layer of 45 neurons for the reconstructed inputs. The rectified linear unit (ReLU) activation function is used for each layer. The nominal operating data of the distillation column is normalized to a feature range of $(0, 1)$. Selecting the appropriate number of neurons for the hidden layers in a neural network is critical, as it directly impacts the model’s learning capacity, accuracy, and generalization ability. While there is no universal rule, several approaches and guidelines are typically followed to ascertain the optimal number of neurons for each hidden layer. Empirical testing and experimentation where various configurations are tested and performance is evaluated on a validation set. By experimenting with different neuron counts and assessing metrics such as mean absolute error and coefficient of determination, the optimal configuration can be identified. This iterative process is often aided by grid search or random search techniques to systematically explore different values and select the best-performing architecture. The autoencoder is trained on 80% of the nominal operating data and validated on the remaining 20% data by minimizing a mean absolute error loss function, which is the difference between the inputs and the reconstructed inputs (outputs) as described in Section 3.3.2. Table 4.1 below summarizes the hyperparameters used for the autoencoder.

Table 4.1: Parameters used for the Autoencoder

Parameters	**Values**
Activation function	ReLU
Epochs	50
learning rate	0.001
Optimizer	Adam
Batch size	160
Loss function	MAE
Neurons at the bottleneck layer	20

The minimization of the loss during training and validation for different numbers of epochs is shown in Fig.4.14. To prevent overfitting, an early stopping was imposed on the autoencoder to stop training if there was no reduction in the loss of the validation data after 5 successive epochs.

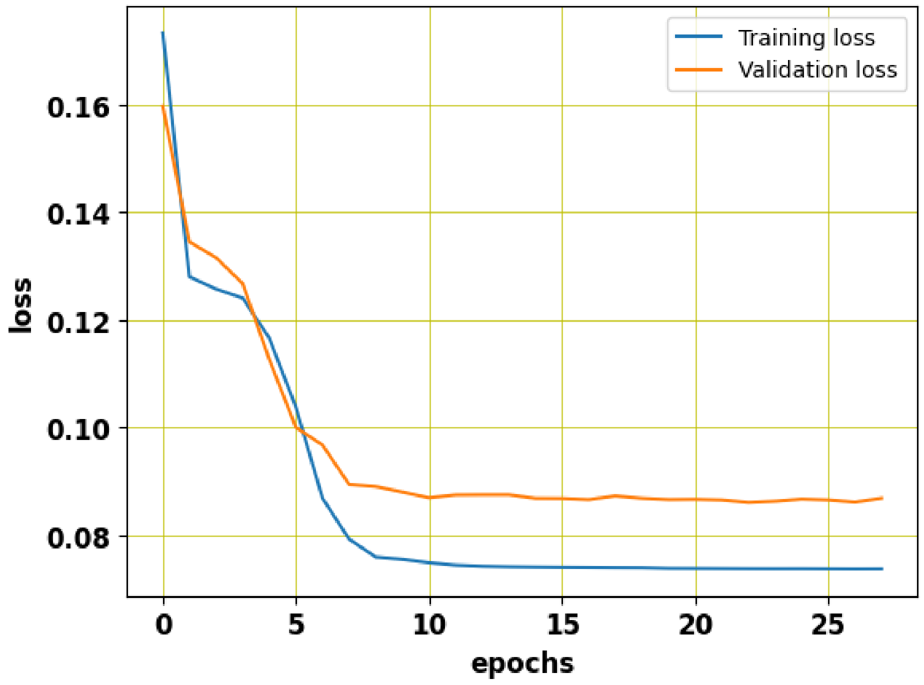


Figure 4.14: Plot of the loss against epochs during training and validation of the autoencoder.

After training, the trained autoecoder is used to reconstruct the entire nominal operating data to obtain a detection threshold. This threshold is taken as a standard deviation shift in the mean of the reconstruction error. Fig.4.15 shows the plot of the reconstruction errors of the nominal operating data. 90% of the reconstructed errors of the nominal operating data fall below the chosen threshold.

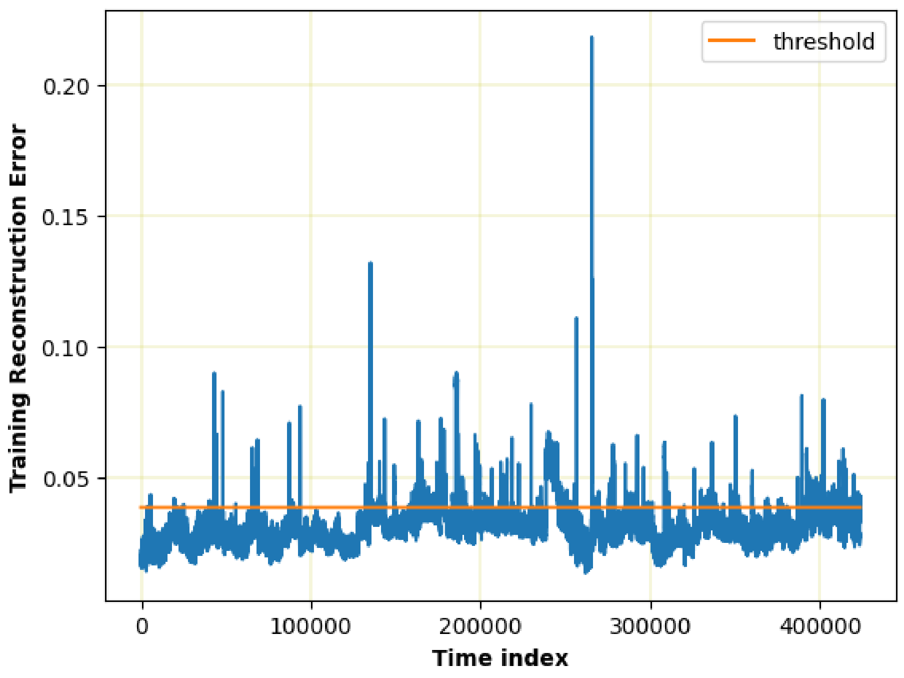


Figure 4.15: Reconstruction errors of the training data using autoencoder. The horizontal axis indicates the index of the time-series data sample.

For flooding detection, the flooding data of a major flooding event is used on the autoencoder; the autoencoder detects flooding based on the principle that for normal operating data, the reconstruction error is smaller, whereas, for anomalous data such as flooding data, the reconstruction error will be higher. Fig.4.16 shows the reconstruction error of the flooding data.

The autoencoder detected flooding at 9:45 using the flooding data of the day of a major flooding event. At this point, there was already a buildup of flooding in the distillation column before flooding became pronounced and noticeable at 10:21. Hence, the conventional autoencoder detected flooding 5 minutes after the buildup of flooding in the distillation column and 36 minutes before flooding became pronounced and noticeable. This late detection is due to the neural network's inability to handle the distillation column's time series data. Hence, there is a need to use neural networks capable of handling time series data such as LSTM.

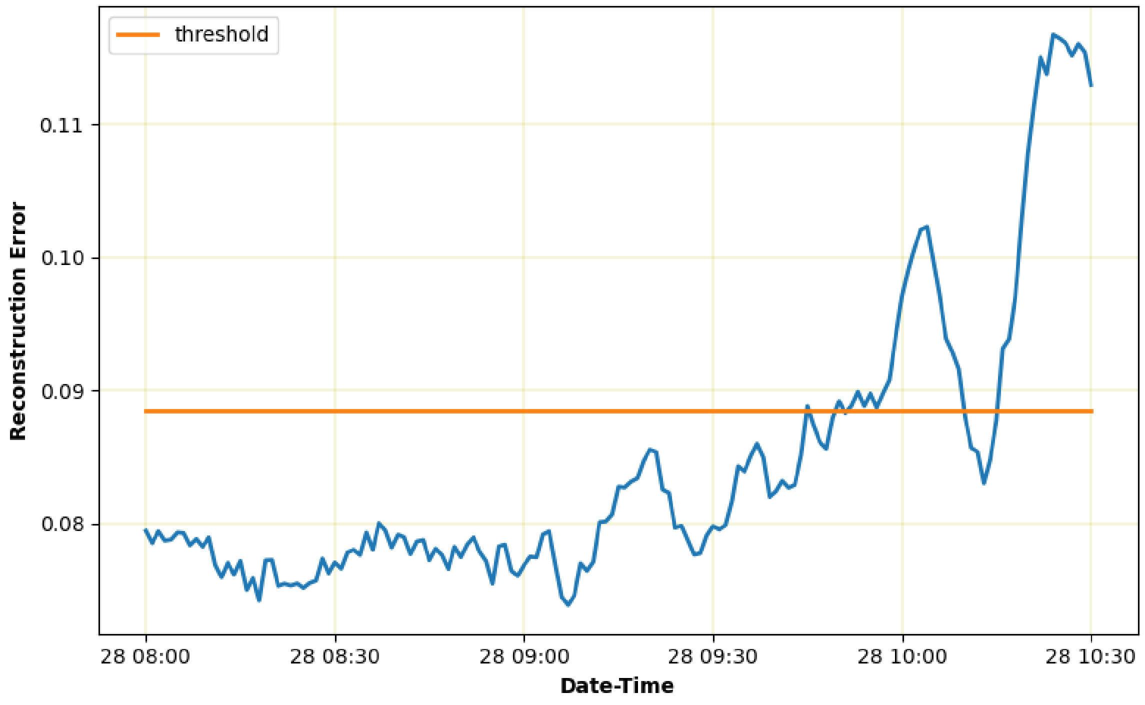


Figure 4.16: Flooding detection using the trained autoencoder on the testing (flooding) data. The horizontal axis represents the day and time of the time-series data sample.

4.5.2 Detection using LSTM Autoencoder

The hidden layers of the encoder and decoder can be replaced with LSTM layers to improve the performance of learning the best parameters to reconstruct the inputs while considering the temporal correlation of the time series data of the distillation column. The hyperparameter tuning of the LSTM autoencoder was done using randomized grid search to get the best possible combination of hyperparameters that will give a better result. 2 layers of LSTM, each of 30 and 20 neurons, respectively, are used as the encoder layer to encode important features of the inputs from high dimension of 45 to a low dimension of 20. The encoded features from the LSTM encoder are passed through a repeat vector to symmetrical 2-layers of LSTM, each of 20 and 30 neurons as the decoder layer. The LSTM-decoder reconstructs the inputs from the encoded features. The rectified linear unit (ReLU) activation function is used for each layer. Table 4.2 gives the parameters of the LSTM autoencoder.

Table 4.2: Parameters used for the LSTM Autoencoder.

Parameters	**Values**
Activation function	ReLU
Epochs	50
learning rate	0.002
Optimizer	Adam
Batch size	128
Loss function	MAE
Sequence length	30

The nominal operating data of the distillation column was normalized to a feature range of $(0, 1)$. The LSTM-Autoencoder was trained on 80% of the nominal operating data and validated on the remaining 20% data by minimizing the loss function, which is the difference between the inputs and the reconstructed inputs (outputs) as described in Section 3.3.2. The minimization of the loss during training and validation for different numbers of epochs is shown in Fig. 4.17. To prevent overfitting, an early stopping was imposed on the LSTM autoencoder to stop training if there was no reduction in the loss of the validation data after 3 successive epochs.

The trained LSTM-Autoecoder was used to reconstruct the entire nominal operating data to obtain the detection threshold. This threshold is taken as a standard deviation shift in the mean of the reconstruction error. Fig. 4.18 shows the plot of the reconstruction errors of the nominal operating data. 95% of the reconstructed errors of the nominal operating data fall below this chosen threshold.

For flooding detection, the flooding data of the major flooding event is used on the LSTM-Autoencoder, and the LSTM-Autoencoder is used to reconstruct the flooding data. The reconstruction errors of the flooding data are shown in Fig.4.19.

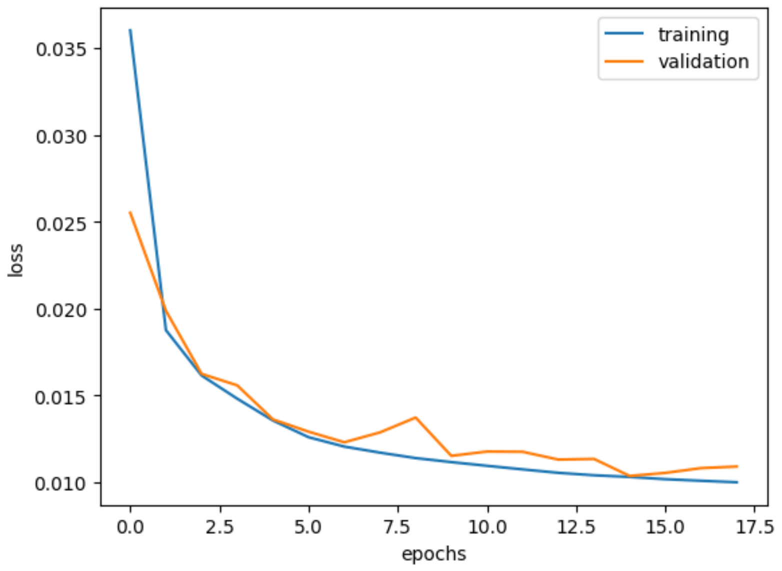


Figure 4.17: Plot of the loss against epochs during training and validation of the LSTM-Autoencoder.

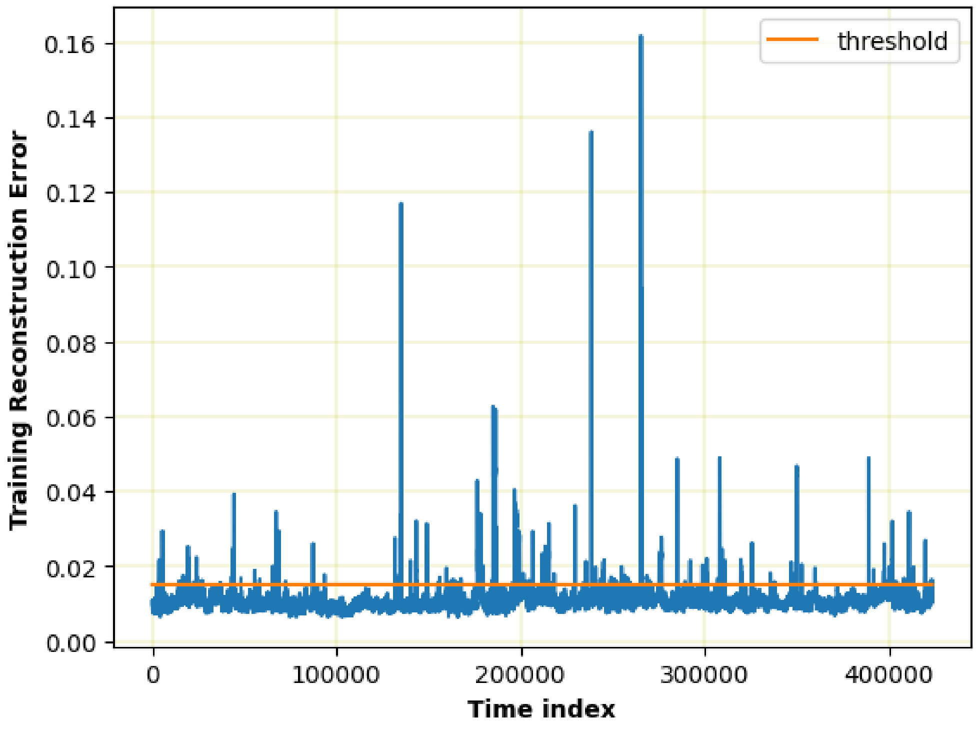


Figure 4.18: Reconstruction errors of the training data using LSTM-Autoencoder. The horizontal axis indicates the index of the each time-series data sample.

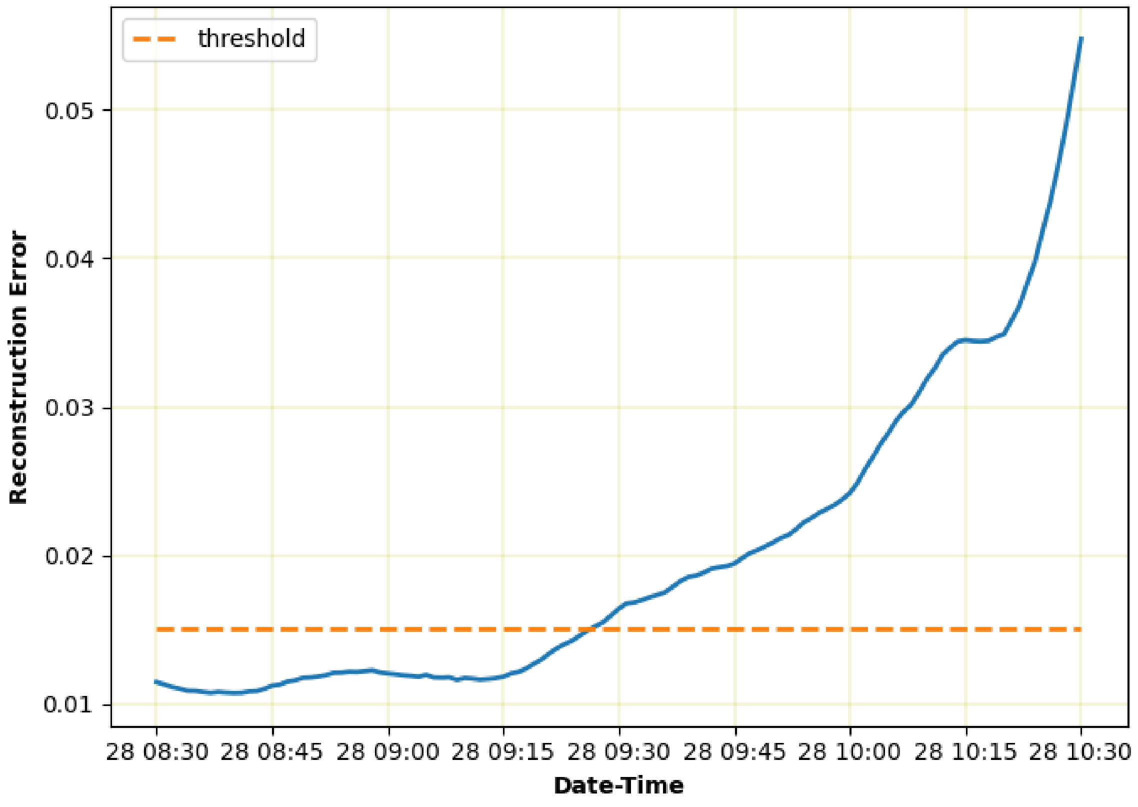


Figure 4.19: Flooding detection using the LSTM-Autoencoder on the testing (flooding) data. The horizontal axis indicates the day and time of each time-series data sample.

The LSTM-Autoencoder detected flooding at 9:27 using the flooding data of the day of a major flooding event. This is an improvement on the detection of the conventional autoencoder. Since flooding buildup in the column started around 9:40 and became pronounced or noticeable at 10:21, the LSTM-Autoencoder detected flooding 13 minutes earlier before the buildup of flooding and 54 minutes before flooding became pronounced in the distillation column. The improvement of flooding detection of the LSTM-Autoencoder over the conventional autoencoder is because LSTM can handle time series data and has the ability to handle complex dependencies within sequential data.

4.6 Flooding Detection through Pressure drop Forecast

4.6.1 Feature Selection

The features used for the pressure drop forecast are selected using engineering knowledge (based on literature and mode of operation of the distillation column) and correlation matrix. From literature and engineering knowledge, reflux ratio is an important feature of a distillation column. It is the ratio between the boil-up rate and the takeoff rate, i.e., the ratio of the reflux rate to the distillate rate. It is not part of the features collected from the distillation column. Hence, it is calculated using Eqn. 4.3 and added to the features.

$$R = \frac{L}{D} \tag{4.3}$$

where R is the reflux ratio, L is the mass or molar flow rate of the liquid reflux returned to the column, and D is the mass or molar flow rate of the distillate stream leaving the distillation column.

The correlation matrix with pressure drop for the features is shown in Fig.4.20. The correlation plot indicates a strong positive correlation (indicated by deep-orange color) between pressure drop and some variables, such as reflux ratio, reboiler duty, temperature measurements, etc. Also, the pressure drop has a strong negative correlation (indicated by the deep blue color) with variables such as bottom product concentration (RVPs), reflux drum level, tray 31 temperature, etc. If two independent variables have a strong correlation, one of such variables must be chosen to prevent the problem of co-linearity [9]. Such cases can be seen from the correction matrix as the reboiler duty has a strong positive correlation with some features such as ‘Nap split temp,’ ‘Reflux Flow,’ ‘Total Reboiler Duty,’ etc. Hence, the reboiler duty is sufficient to represent most of the features for the pressure drop forecast. The reboiler duty (‘Reboiler Duty’) is selected as a feature since it strongly correlates with pressure drop and gives information about the heat duty in the column.

The reflux ratio (‘Reflux Ratio’) is also retained as a feature since it indicates the amount of reflux back to the column compared to the amount collected as the distillate. Feed Flow ('Feed Flow') gives the feed flow entering the column. Hence, it is retained as a feature.

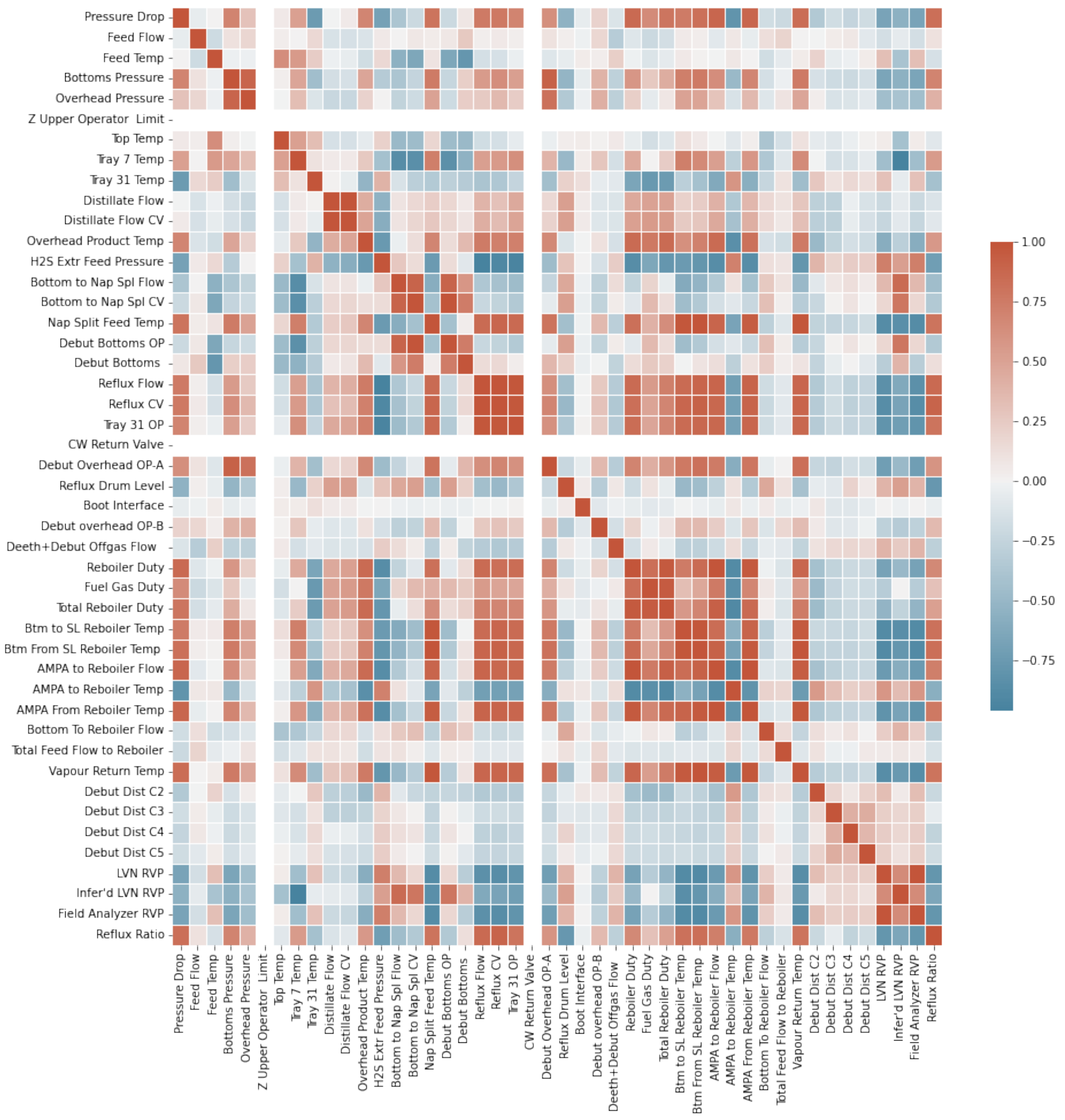


Figure 4.20: Correlation matrix for the column data, the right-hand parameter indicates the correlation coefficient.

A pressure-compensated temperature indicator (TI) is placed on tray-7 to manipulate reboiler duty to maintain a constant bottom quality and on tray-31 to manipulate the reflux rate to maintain a constant distillate quality. Hence, tray-7 and tray-31 temperatures (‘Tray 7 Temp’, ‘Tray 31 Temp’) are selected as a feature for pressure drop forecast, and they also have a strong correlation with pressure drop. The overhead product temperature (‘Overhead Product Temp’) is also retained as a feature since it gives information about the top product in the column and has a strong positive correlation with the pressure drop. The liquid level in the reflux drum (‘Reflux Drum Level’) is also selected to forecast the pressure drop as it gives information about the accumulation of liquid in the reflux drum.

A measure of the distillate C_4 product quality(‘Debut Dist C4’) and bottom product quality (‘LVN RVP’) are also selected for the forecast since they are affected during flooding. Finally, the pressure drop (‘Pressure Drop’) is selected since it describes the pressure drop trend in the distillation column. A total of 10 features are selected to build ML models to forecast the pressure drop for early flooding detection in the distillation column.

4.6.2 Synthetic Flooding Data Generation

After selecting the needed features, synthetic flooding regime data are generated using TimeGAN as described in Section 2.7 to combat the issue of data imbalance. The dynamics of the column when transitioning from normal operating conditions to flooding conditions is essential and needs to be captured by the TimeGAN model. This transition is the pre-flooding condition, where the TimeGAN model can capture the dynamics prior to flooding. Hence, samples up to 60 minutes before flooding are chosen as the pre-flooding samples. The flooding regime data is chosen to contain 40 minutes of data from the normal operating data, the pre-flooding samples, and 35 minutes after the buildup of flooding just before its full development in the column. This flooding regime data is illustrated in Fig.4.21. This will enable the transition from normal operating conditions to the buildup of flooding conditions to be captured effectively in the synthetic flooding data.

The flooding regime data are normalized to a feature range of $(0, 1)$ and are preprocessed in time batches using the sliding window described in Section 3.3.2.4. Each batch of the flooding regime data has 31 sequence lengths of time-series data. The TimeGAN was trained continuously by feeding the real flooding regime data batchwise. 24 batches of the data are used to carry out stochastic gradient descent optimization to update the parameters of the TimeGAN model. A learning rate of

0.0003 is used for this optimization. The dynamics of the flooding regime data will be captured and generated in similar batches to the training data. Hence, each time batch of the synthetic flooding data is independent and identically distributed (i.i.d.). Similarly, each batch of the generated flooding data has 31 timesteps. A sample batch of the real flooding regime data and the generated data from the TimeGAN is shown in Fig.4.22.

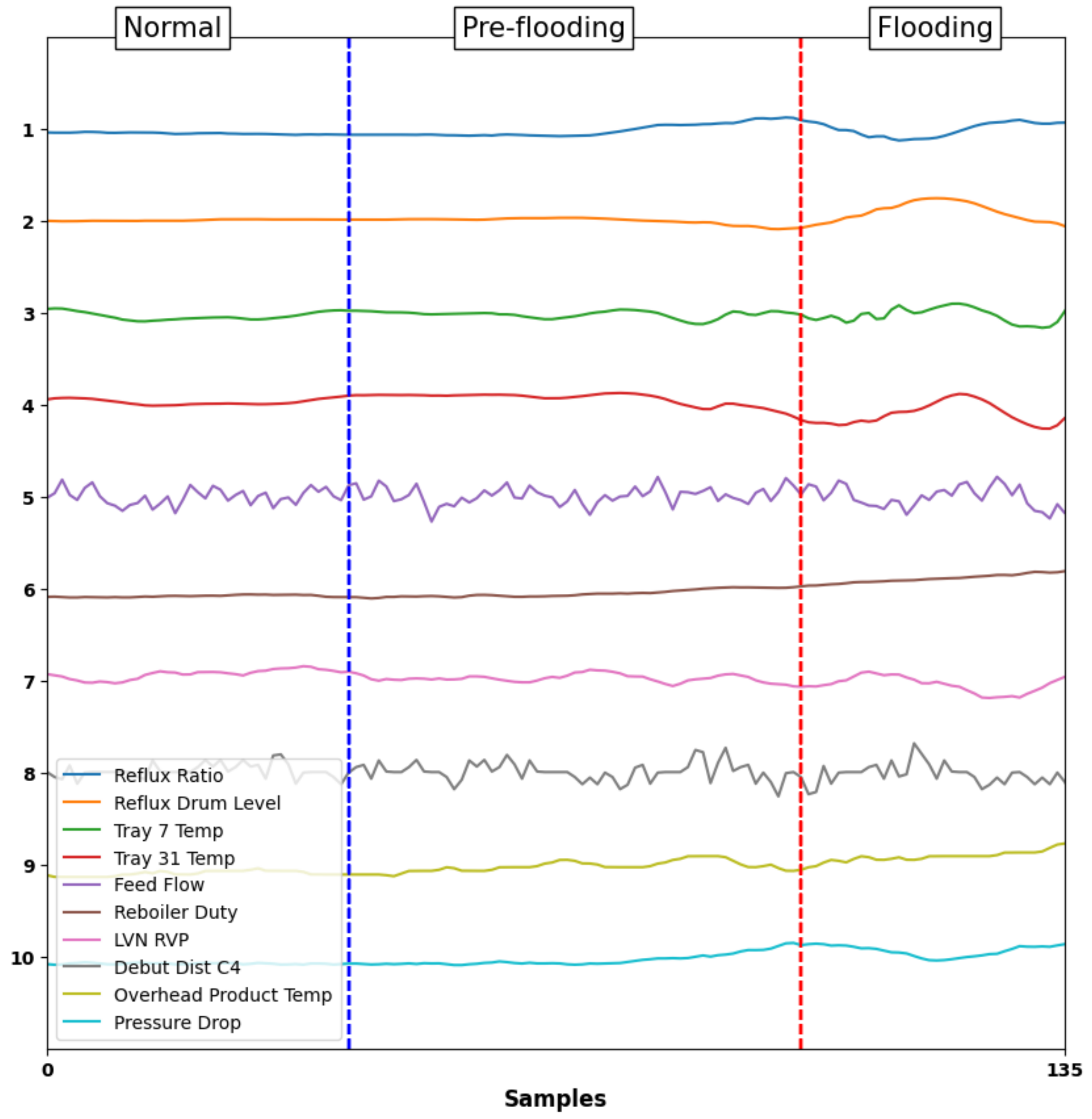


Figure 4.21: Sample plot of normalized original flooding regime data used for training the TimeGAN model to generate synthetic flooding data.

It is important to state that the graphical comparison shown in Fig.4.22 is not a form of evaluation of the generated synthetic flooding data. Generated time-series data are difficult to evaluate graphically, but they can be evaluated statistically and based on their predictive properties. These forms of evaluation are considered in this work to evaluate the synthetic samples that the TimeGAN model generated.

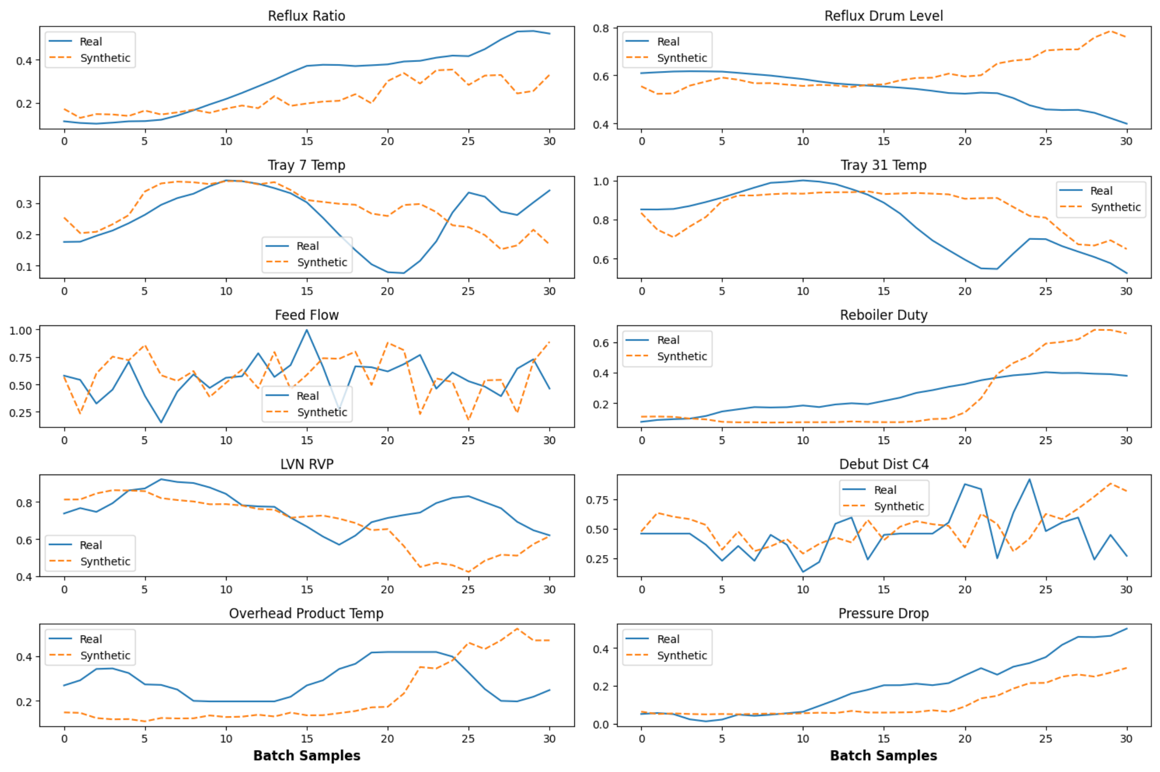


Figure 4.22: Visualization of a sample batch of normalized real and synthetic flooding regime data.

To be sure that the TimeGAN is generating virtual samples that are similar to the real samples, the generated data are evaluated based on their distribution using principal component analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) plots as shown in Fig. 4.23 by flattening the temporal dimension. This visualizes how closely the distribution of generated samples resembles that of the original in 2-dimensional space. PCA is to validate the linear distribution in 2-dimensional space, while t-SNE is for the non-linear distribution in 2-dimensional space. The original data (in blue) and the generated data (in red) align closely and are almost perfectly in sync, as observed in the plots.

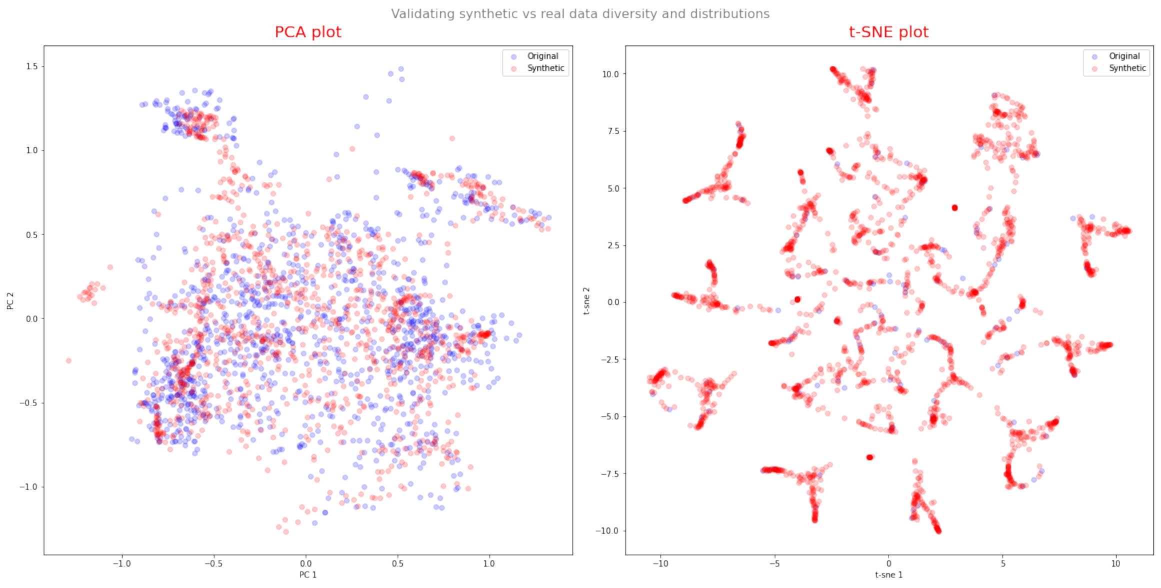


Figure 4.23: Validating the synthetic flooding data diversity and distribution with the real flooding data using PCA and t-SNE plot.

For the synthetic samples to be useful, the sampled data should inherit the predictive characteristics of the original. In particular, we expect TimeGAN to excel in capturing conditional distributions over time. Therefore, using the synthetic dataset, we train a post hoc sequence-prediction model (by optimizing a 2-layer LSTM) to predict next-step temporal vectors over each input sequence. This simply involves using a 2-layer LSTM model to predict the next step in the time-series sequence of data. For example, the sequence length of each batch of the data generated by the TimeGAN is 31. The LSTM sequence prediction layer will be trained on the 30 previous sequence lengths so as to predict the final sequence length in each batch. This way, the synthetic samples can be evaluated based on their predictive characteristics. Then, the trained sequence prediction model is evaluated on the original dataset. This is a case of train-synthetic test-real (TSTR) evaluation. This evaluation is compared with training the LSTM sequence-prediction model on the original flooding regime data and evaluating the model on the same original flooding data: a case of train-real test-real (TRTR) evaluation. The two evaluation methods should have similar results if the synthetic dataset is similar to the original dataset.

Also, to verify the usefulness of the synthetic dataset, the original dataset is augmented with the synthetic dataset. The augmented data is used to train the LSTM sequence-prediction model, and then the trained model is evaluated on the original dataset. The performance of the trained models for each of the evaluations is measured in terms of the mean absolute error (MAE), root mean square error (RMSE), and coefficient of determination (R^2) and are shown in Table 4.3

Table 4.3: Predictive evaluation of the synthetic flooding regime data.

Model	**MAE**	**RMSE**	$\mathbf{R^2}$
TSTR	0.0727	0.0911	0.6071
TRTR	0.067	0.0765	0.7166
Augmented	0.0500	0.0710	0.7500

From the results obtained, the synthetic flooding regime data generated using TimeGAN inherit the predictive characteristics of the original data, and its augmentation with the original flooding data leads to a reduction in error and better training of the sequence-prediction model. Hence, the trained TimeGAN model can be used to generate more synthetic flooding regime samples to augment the column dataset and eliminate the issue of data imbalance. The TimeGAN model is used to generate 51 flooding regime events to augment the original data. The generated synthetic flooding data has 6,168 batches, each batch having 31 sequence lengths of synthetic flooding data. The ratio of flooding data to normal operating data before and after augmentation with synthetic flooding regime data is recorded in Table 4.4.

Table 4.4: Flooding and normal operating data ratio in the dataset before and after augmentation with synthetic flooding regime data.

Augmentation	**Flooding [%]**	**Normal [%]**	**Total Samples**
Before	0.40	99.60	$250,000$
After	43.37	56.63	$441,208$

4.6.3 Model Training for Pressure drop Forecast

To prepare the data for supervised learning. The data has to be transformed based on Eqn. 3.20 given in Section 3.4. The window size and the response size must be chosen for this transformation. The window size is the number of past time steps in the sliding window transformation, while the response size is the number of future time steps in the transformation. The sliding window parameters are chosen using a grid search to investigate the window size between 5 minutes and 20 minutes. In contrast, the response size is investigated between 15 mins and 35 mins using random forest as a reference model for a regression analysis. Since a transformation like this leads to additional columns in the dataset, the aim is to have a small window size and a long response size while maintaining a good prediction. This helps to optimize storage and to reduce computational time during training. The result of this investigation is shown in Table 4.5. From the results obtained, there is a decrease in the performance of the model as the window size increases. This is due to the increase in the number of features used as predictors. Also, the model's performance is reduced for each window size as the response size increases. A window size of 5 minutes might be too small to capture the dynamics of the distillation column; the chosen window size is 10 minutes, and the response size is 20 minutes. This combination of window and response size gives good prediction accuracy while trying to minimize training time and optimize memory usage. Hence, the pressure drop forecasting is done for a 20-minute forecast.

Table 4.5: Investigated window and response size for data transformation using the sliding window method.

RandomForest	window size = 5		window size = 10		window size = 15		window size = 20	
response size	**RMSE**	**R^2**	**RMSE**	**R^2**	**RMSE**	**R^2**	**RMSE**	**R^2**
15	0.0345	0.9826	0.0386	0.9783	0.0391	0.9778	0.0393	0.9772
20	0.0378	0.9789	0.0403	0.9763	0.0379	0.9789	0.0397	0.9768
25	0.0392	0.9774	0.0397	0.9770	0.0421	0.9741	0.0408	0.9755
30	0.0429	0.9728	0.0417	0.9746	0.0421	0.9739	0.0448	0.9707
35	0.0444	0.9711	0.0450	0.9704	0.0451	0.9702	0.0437	0.9721

Different supervised ML algorithms are compared for a 20-minute pressure drop forecast. These algorithms include linear regression, random forest, extra trees, gradient boosting, and LSTM. The training is done using k-fold cross-validation using $k = 3$. The training dataset is split into k disjoint equally sized subsets. The training of the models is conducted on $k - 1$ datasets, while the validation is done on a single subset. This training is repeated k times, always with different validation sets. This gives a more robust approach to knowing the model's performance across the dataset. Hyperparameters of the tree regressors were optimized using a randomized grid cross-validation search with $k = 3$. The LSTM is also optimized by considering the number of neurons in the layer and the training epochs. Table 4.6 shows the mean result of the model's performance on the validation set. From the results obtained, all the supervised learning models have good results on the validation data, each with $R^2 > 0.95$. It takes longer to train the gradient boosting model and random forest model compared to other models.

Table 4.6: Performance of different supervised ML algorithms on the validation data set for pressure drop forecast.

Algorithm	$\mathbf{R^2}$	**RMSE**	**Training time** [s]
Linear Regression	0.9844	0.0322	5.97
Random forest	0.9964	0.0154	9970.69
Extra trees	0.9979	0.0118	1463.47
Gradient Boosting	0.9864	0.03	12028.81
LSTM	0.9886	0.0283	2553.09

4.6.4 Operating States Determination

The time series pressure drop data are classified based on the last 10 minutes measurements and 20 minutes forecast window to determine the operating condition of the column, i.e., flooding or not flooding. The pressure drop data is transformed based on Eqn. 3.20 to have the same form as the data used for training. Each transformed data point is independent and identically distributed with a total window size of 30 minutes. The trend and level of each data point's 30 minutes window are determined from the slope of a linear fit and the median of the 30 minutes window for each data point, respectively. Flooding is associated with an increase in pressure drop. Hence, the data is filtered for positive slopes. These features are used to identify meaningful clusters in the pressure drop data by using k-means algorithm. As the algorithm is a distance-based method, prior data scaling is performed before implementing the algorithm, and the data is transformed back to the original values for visualization.

The elbow method described in [151] as illustrated in section 2.6.6 is used to determine the number of clusters in the data. The k-means algorithm is applied by varying the number of clusters and obtaining the sum of squared distances of samples to their closest cluster, which is the inertia. The number of clusters reached just before additional clusters start leading to a smaller change in inertia (i.e., an elbow where the distortion in inertia goes down slowly) is the appropriate number of clusters. This result is shown in Figure 4.24. The elbow is found at six clusters, after which the change in inertia becomes smaller. The k-means algorithm is implemented on the extracted features of the historical pressure drop for a 6 number of clusters. The visualization is shown in Fig.4.25 by plotting the median against the slope to show the cluster centers. Prior scaling of the extracted features (median and slope) was done before implementing the k-means algorithm. The data was transformed back to the original values for visualization purposes.

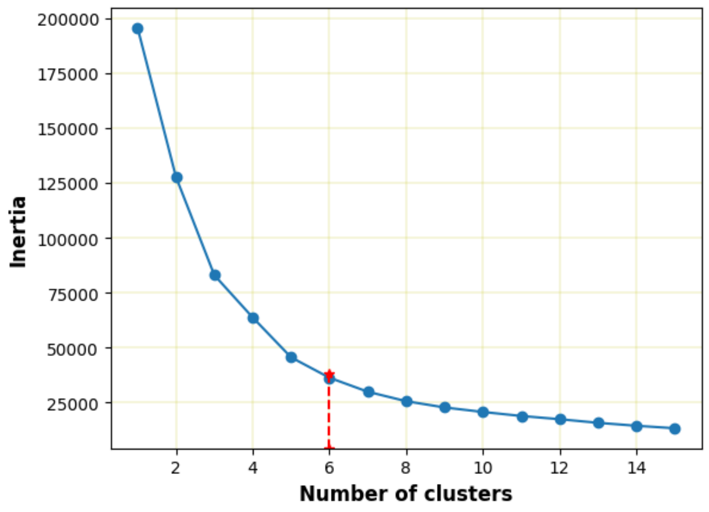


Figure 4.24: Elbow method to determine the number of clusters for k-means clustering algorithm.

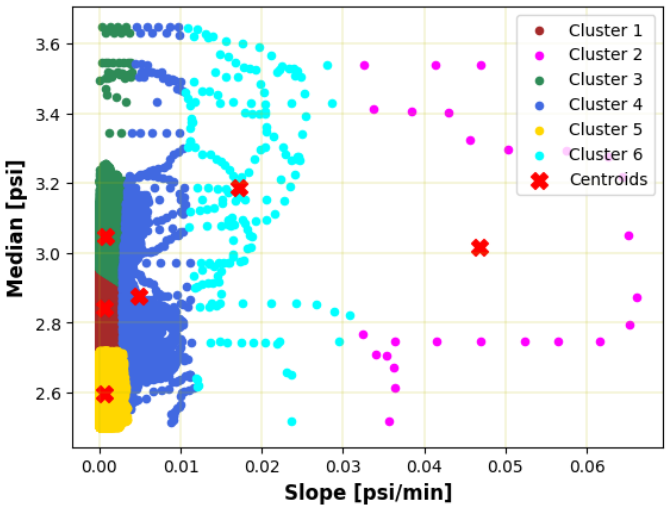


Figure 4.25: Plot of the median against the slope and the resulting decomposed time-series historical pressure drop data clusters.

Low/moderate pressure drops characterize normal operation; hence, clusters 1, 3, 4, and 5 indicate the normal operating state in the pressure drop since most of the data lies around the region of zero or low slope. Clusters 2 and 6 describe the operating states in which the pressure drop is increasing. These clusters represent the possible flooding state of the column since an increase in pressure drop and high-pressure drop characterizes flooding.

4.6.5 Model Testing and Flooding Prediction

We must use all the available models in our arsenal for early flooding detection to know a suitable model for flooding detection. Hence, all the models are evaluated on the holdout testing dataset. This holdout testing data is collected on the day of a major flooding event in the distillation column and has not been exposed to the models during training. The performance of the trained model on the testing data is shown in Table 4.7.

Table 4.7: Performance of the supervised ML algorithms on the holdout-testing data set for pressure drop forecast.

Algorithm	$\mathbf{R^2}$	**RMSE**
Linear Regression	0.9617	0.0285
Random forest	0.9414	0.0352
Extra trees	0.9583	0.0297
Gradient boosting	0.9605	0.0289
LSTM	0.9190	0.0412

The trained-supervised ML models (to forecast pressure drop) in conjunction with the trained clustering algorithm (to classify the forecasted pressure drop) are needed for efficient and accurate flooding prediction in the distillation column. This is achieved by using the supervised ML model to forecast the pressure drop for 20 minutes ahead and then using the clustering algorithm to classify the forecasted pressure drop into a normal operating state or flooding state. The time at which the different supervised ML methods detected flooding is shown in Table 4.8.

Table 4.8: Time of flooding detection of trained-supervised ML models to forecast the pressure drop across the column using the holdout testing data. Time is given before flooding starts building up in the column and before flooding becomes pronounced (full development of flooding in the column).

Algorithm	**Detection Time**	**Time [mins] Before flooding**	**Time [mins] Before pronounced flooding**
LSTM	9:21	19	60
Random forest	9:22	18	59
Extra trees	9:22	18	59
Gradient Boosting	9:32	8	49
Linear Regression	9:37	3	44

From the results given in Table 4.8, LSTM has the earliest detection time by predicting flooding as early as 19 minutes before flooding occurred in the distillation column and 60 minutes before flooding became pronounced (causing an overflow in the reflux drum) in the distillation column after its full development. Random forest and extra trees predicted flooding 18 minutes before flooding in the column and 59 minutes before the flooding became pronounced. This early detection will give the operators ample time to make engineering decisions to prevent the full development of flooding in the column. The linear regression has the least detection time, just 3 minutes before flooding occurs in the distillation column, and this detection time might be too late for operators to make engineering decisions in order to prevent flooding or to avoid its full development.

The trained-supervised ML models having the best results (LSTM, Random forest, Extra trees), i.e., predicting flooding early, are further evaluated based on their performance for early flooding detection as shown in Fig.4.26. Fig.4.26 shows the comparison between the 20 minutes pressure drop forecast given by the trained-supervised ML models at 7:02 on the day of a major flooding event in the column and the actual pressure drop observed for the forecasted 20 minutes. The forecasted pressure drop is decomposed and classified by the trained clustering algorithm into a normal operating or flooding cluster. As shown in Fig.4.26, the forecast given by the supervised learning models almost matched the trend of the actual pressure drop observed in the column for the next 20 minutes. The forecast classification indicates that the column will operate normally for the next 20 minutes as the extracted features from the forecast fall in the normal operating clusters.

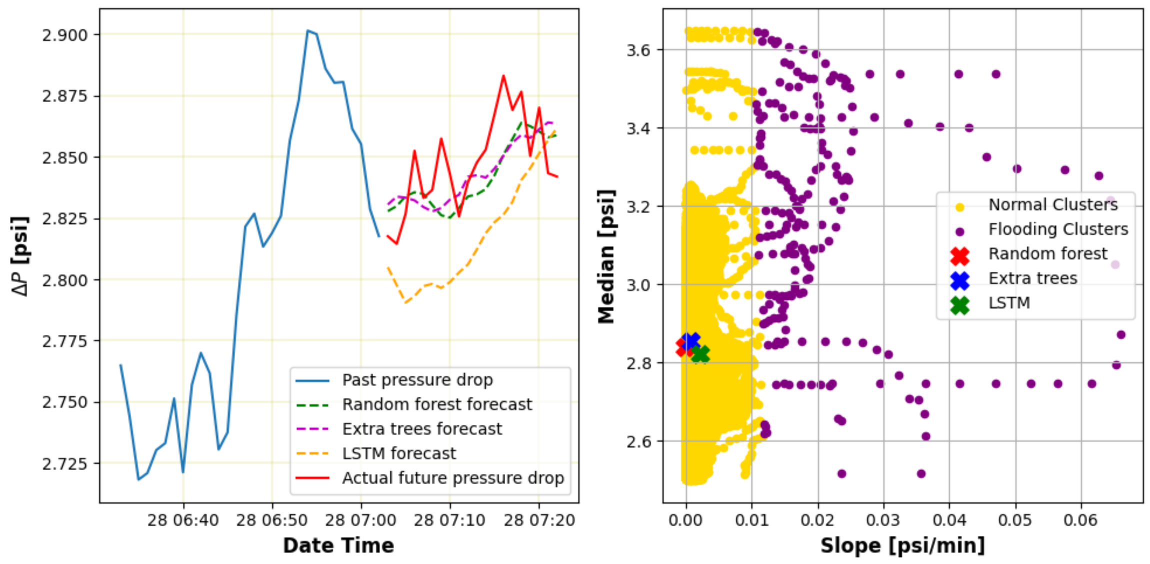


Figure 4.26: Sample illustration of the pressure drop forecast by the supervised ML algorithms for flooding prediction at 7:02 on the day of a major flooding event in the distillation column. The right-hand figure shows the classification of the pressure drop forecast.

To further evaluate the performance of the trained-supervised ML models, a sample forecast at 9:15 is shown in Fig.4.27. The trained-supervised ML models are able to capture the increasing trend in the pressure drop for the next 20 minutes. This can be validated by the noticeable increase in pressure drop observed in the distillation column through the next 20 minutes. This upward trend in pressure drop shows that the trained ML algorithms captured the dynamics of the distillation column. At this time, the models are still forecasting a normal operating state in the column for the next 20 minutes.

At 9:21 on the day of a major flooding event, the LSTM model predicted a possible flooding state. This is 19 minutes prior to the build-up of flooding in the distillation column. Fig.4.28 shows the forecast of the trained-supervised ML models. The models captured the continuous increase in the pressure drop as validated by the actual pressure drop observed in the column for the next 20 minutes. The extracted features from the LSTM fall under the flooding clusters. This is an indication of possible flooding based on the 20-minute forecast. The extracted features from the forecast of the random forest and the extra trees model are closer to the edge of the normal clusters, as shown in Fig.4.28.

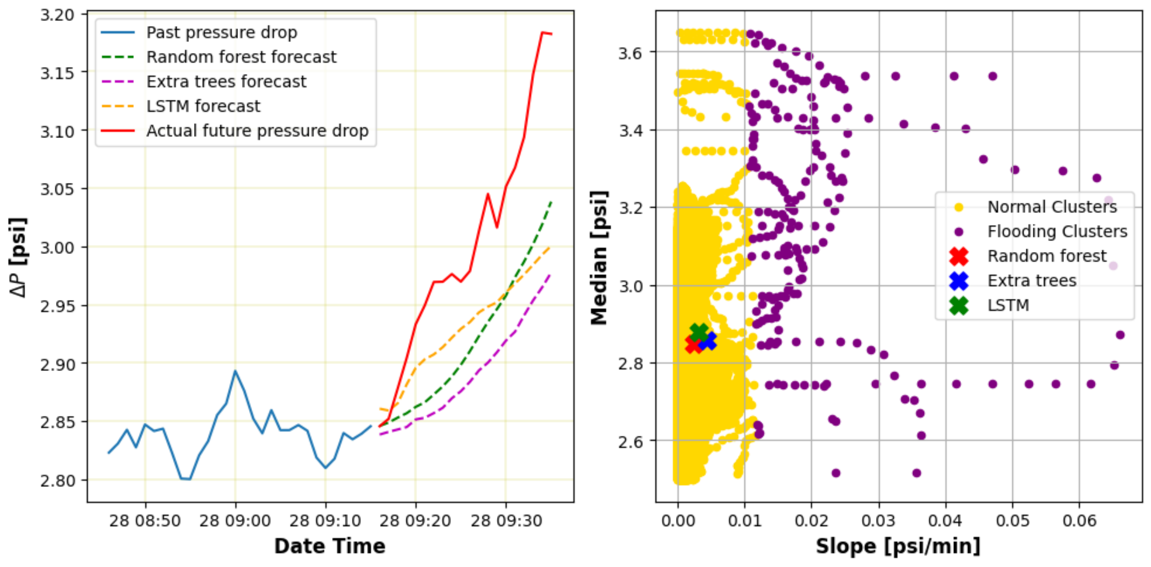


Figure 4.27: Sample illustration of the pressure drop forecast by the supervised ML algorithms for flooding prediction at 9:15 on the day of a major flooding event in the distillation column.

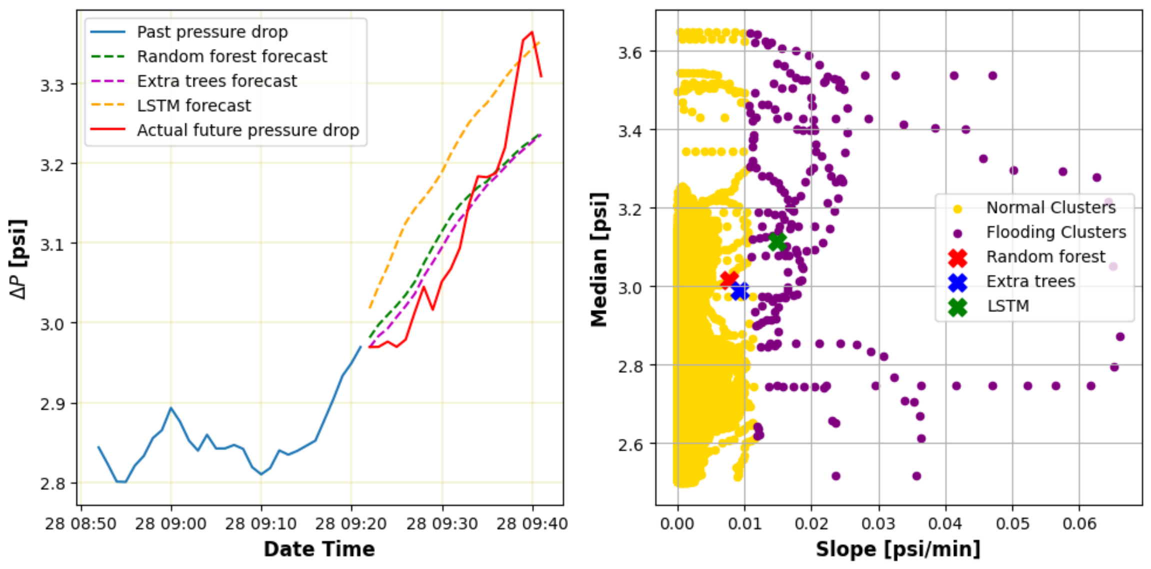


Figure 4.28: Sample illustration of the pressure drop forecast by the supervised ML algorithms for flooding prediction on the day of a major flooding event in the distillation column at 9:21 on the day of a major flooding event (19 minutes before flooding occurs in the column).

At 9:22, which is 18 minutes prior to flooding, the trained-supervised learning models are forecasting a possible flooding event based on the extracted features from their forecast, as shown in Fig.4.29. The models predicted a continuous rise in the pressure drop, as validated by the actual pressure drop observed in the column. At this time of detection, the pressure drop is within the normal operating range ($dp < 3.5$psi). Still, the forecast gives insight into flooding that will happen soon and the pressure drop that will keep increasing. This early prediction by the ML algorithms will give the operators early warning risk of possible flooding within the distillation column and enough time to make engineering decisions to prevent flooding or before it fully develops.

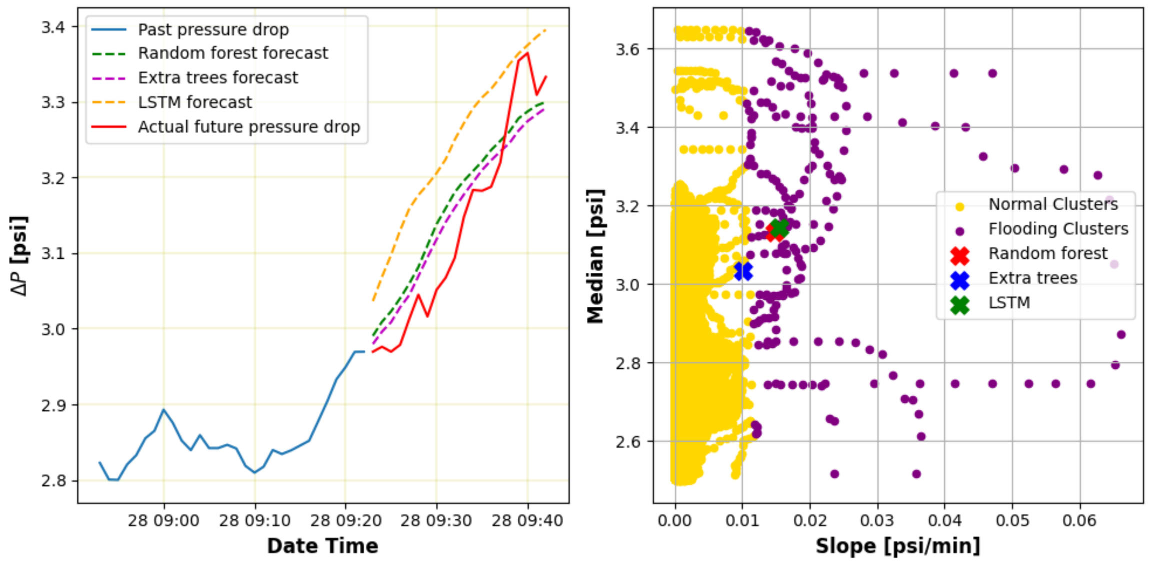


Figure 4.29: Sample illustration of the pressure drop forecast by the supervised ML algorithms for flooding prediction on the day of a major flooding event in the distillation column at 9:22 on the day of a major flooding event (18 minutes before flooding occurs in the column).

The results shown in Fig.4.26, 4.28 and 4.29 illustrate that the Long Short-Term Memory (LSTM) method produces discrepancies between the actual and predicted pressure drops, suggesting some limitations in its accuracy for precise pressure drop prediction. However, these discrepancies are consistent across the model, suggesting a bias rather than random error, which can still provide valuable trends for early flooding detection. The LSTM model's proficiency in capturing flooding events earlier than other models can be attributed to its architecture, which is designed to learn temporal patterns and dependencies within time series data. In the context of pressure drops, this ability allows LSTM to recognize subtle, preceding indicators of flooding that might go unnoticed with conventional methods or models without temporal learning capabilities. Thus, while it may not accurately predict exact pressure drop values, the LSTM captures key signals leading up to flooding events, enabling timely detection of emerging flooding trends.

4.6.6 Additional Variables as Predictors

Additional predictors for pressure drop forecast can be developed based on the operation of the distillation column. These variables are developed to check for an improvement over the previous results obtained for the early detection of flooding in the distillation column through pressure drop forecast. For this case, three approaches are considered.

4.6.6.1 Temperature-Pressure Difference Indicators

Based on the operation of the distillation column, a pressure-compensated temperature indicator is placed on tray-7 to manipulate reboiler duty to maintain a constant bottom quality. Also, on tray-31, to manipulate the reflux rate to maintain a constant distillate quality. A flooding indicator can be developed based on the temperature of these two trays and the pressure drop across the column, as shown below

$$dTdP_i = \frac{T_{tray,i} - T_{ovhd}}{\Delta P}, \quad i = 7, 31 \tag{4.4}$$

where $T_{tray,i}$ is the temperature of i^{th} tray (tray-7, tray-31), T_{ovhd} is the temperature of the overhead product, ΔP is the pressure drop across the column, $dTdP_i$ is the indicator variable for the i^{th} tray (tray-7, tray-31).

Two indicators were developed for tray-7 and tray-31 and can also be used as predictors for pressure drop forecast. Fig.4.30 shows the plot of the indicators and the pressure drop on the day of a major flooding event. It can be seen that the two developed indicators have an inverse relationship with pressure drop. They react sharply in an opposite direction to changes in pressure drop.

These two indicators are added to the 10 selected features and are used as predictors for the pressure drop forecast to detect flooding. This was carried out using the same procedure as before. Table 4.9 gives the performance on the validation set of the ML algorithms trained on these new sets of predictors. The results show that the trained ML algorithms perform well on the validation dataset with a coefficient of determination (R^2) greater than 0.5. Hence, all the trained models are tested on the holdout testing data (These data samples have not been exposed to the models during training), and the results are shown in Table 4.10.

Though the trained ML algorithms have good performance on the validation data set and the holdout-testing data, there is a need to evaluate the models further using the flooding data collected on the day of a major flooding event to determine how early flooding can be predicted in the distillation column when these developed indicators are added to the selected features.

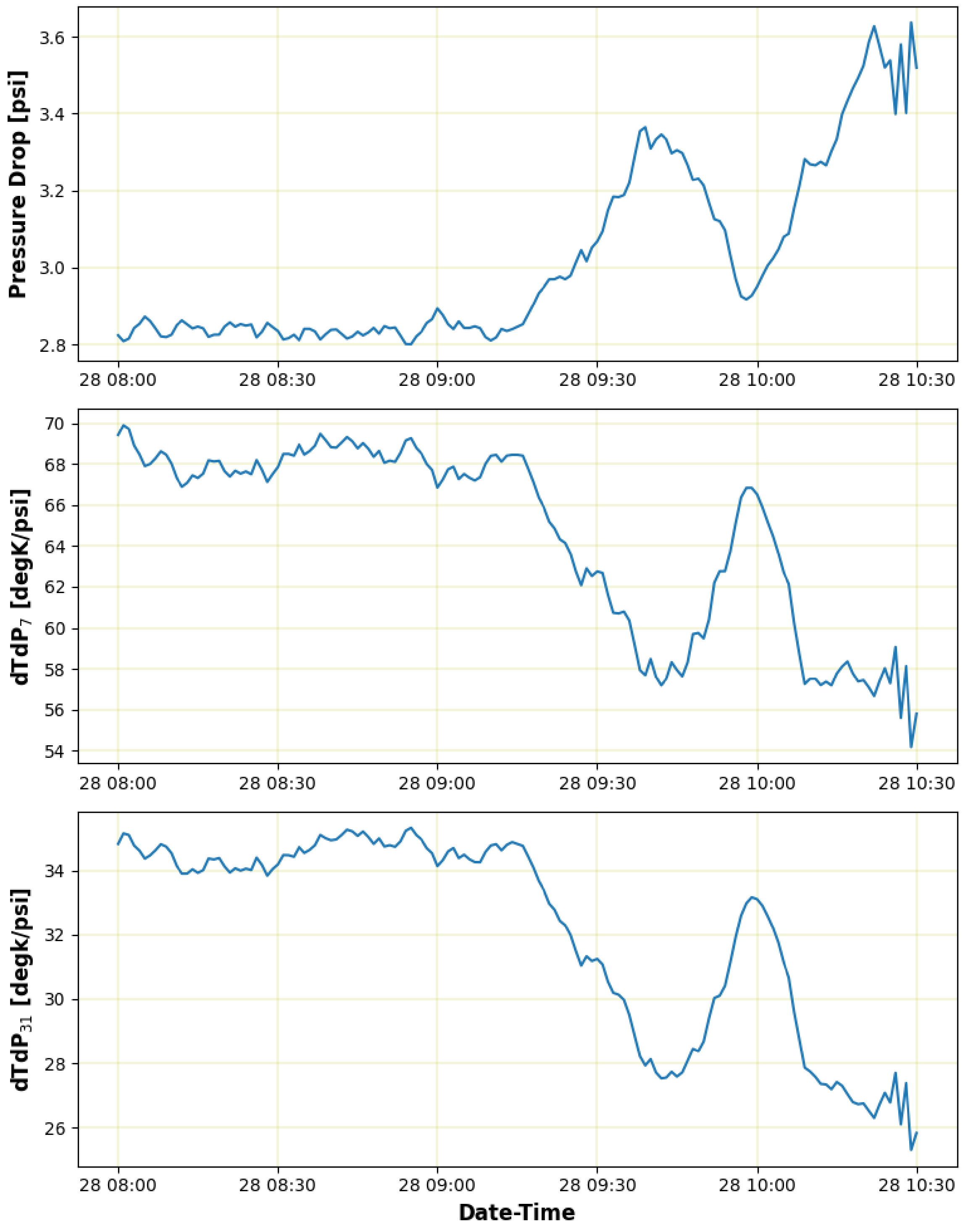


Figure 4.30: Plot of pressure drop and the temperature-pressure indicators during a major flooding event. The horizontal axis indicates the day and time of each time-series data sample.

Table 4.9: Performance of different supervised ML algorithms on the validation data set for pressure drop forecast using the developed indicators as additional variables.

Algorithm	**R^2**	**RMSE**
Linear Regression	0.9850	0.0316
Random forest	0.9886	0.0275
Extra trees	0.9897	0.0262
Gradient boosting	0.9861	0.0304
LSTM	0.9882	0.0283

Table 4.10: Performance of different supervised ML algorithms on the holdout-testing data set for pressure drop forecast using the developed indicators as additional predictors.

Algorithm	**R^2**	**RMSE**
Linear Regression	0.9640	0.0276
Random forest	0.9428	0.0348
Extra trees	0.9599	0.0291
Gradient boosting	0.9601	0.0290
LSTM	0.9144	0.0424

The combination of these trained-supervised learning models and the trained clustering model is used to detect flooding using the flooding data obtained on the day of a major flooding event. The detection time is shown in Table 4.11. There is a slight improvement in the detection of linear regression as it tends to detect flooding earlier with the developed indicator than without the developed indicators. However, there is a significant lateness in the detection of Extra trees. There is no significant improvement in the overall earliest detection time. The LSTM gives the earliest detection time at 9:21, 19 minutes before the occurrence of flooding, and 60 minutes before its full development. This is also the best result obtained without using the developed indicators.

Table 4.11: Time of flooding prediction as given by the trained-supervised ML algorithms with the developed indicators as additional predictors.

Algorithm	**Detection Time**	**Time [mins] Before flooding**	**Time [mins] Before pronounced flooding**
LSTM	9:21	19	60
Random forest	9:22	18	59
Gradient Boosting	9:32	8	49
Extra trees	9:36	4	45
Linear Regression	9:36	4	45

A sample pressure drop forecast at 9:21 is given in Fig. 4.31. This is at 19 minutes before flooding in the distillation column. The trend of the 20-minute pressure drop forecast given by the trained algorithms agrees with the actual pressure drop observed in the column. The clustering model determines the operating state of the distillation column for the next 20 minutes based on the extracted features of the pressure drop forecast given by the models. As seen in Fig.4.31, the forecast given by the LSTM detected flooding at this time since the extracted features fall into the flooding clusters as given by the clustering model.

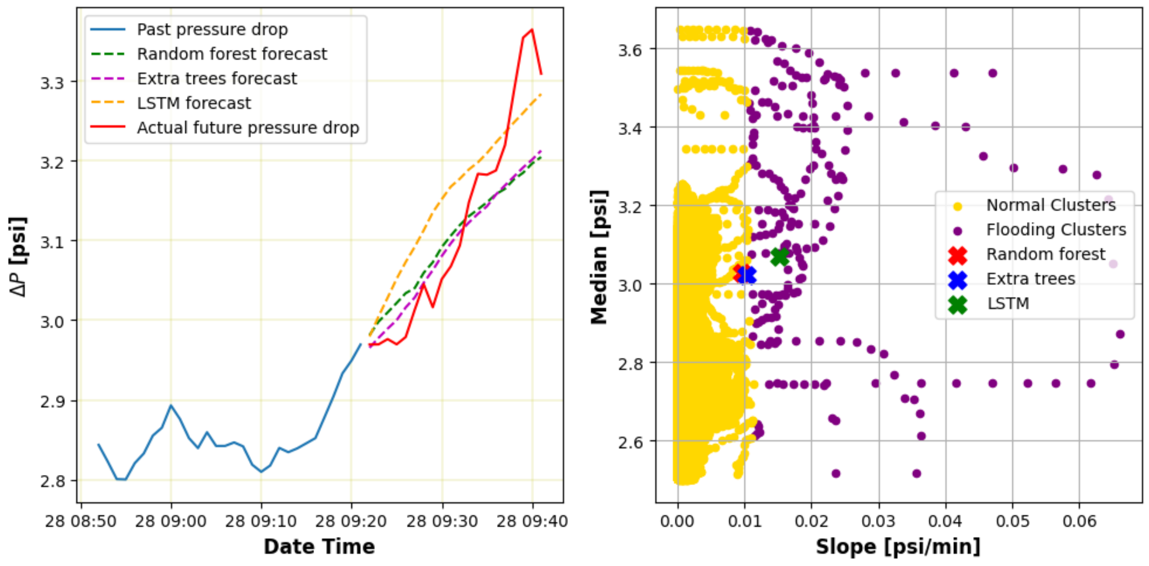


Figure 4.31: Pressure drop forecast at 9:21 (19 minutes before flooding) by the supervised ML algorithms for flooding prediction using the developed indicators as additional predictors.

4.6.6.2 Time-difference Variables (Gradients)

For the pressure drop forecast, 10 features, including the pressure drop, were selected as predictors. These selected features can be converted to gradient variables with respect to time according to Eqn. 4.5.

$$\Delta X_i = \frac{X_{i,t} - X_{i,t-1}}{\Delta t} \tag{4.5}$$

where X_i are the selected features, t is the time step, Δt is the time change, which is taken as 1 minute, and ΔX_i is the gradient variable of the selected variable.

These gradient variables are used as the predictors for the pressure drop forecast. Table 4.12 gives the performance on the validation set of the ML algorithms trained on these gradient variables. The models perform well on the validation dataset. Hence, the models are tested on the holdout testing data (These data samples have not been exposed to the models during training), and the results are shown in Table 4.13.

The combination of these trained-supervised learning models and the trained clustering model is used to detect flooding using the flooding data. This dataset was not exposed to the supervised learning models during training. The detection time is shown in Table 4.14.

Table 4.12: Performance of different supervised ML algorithms on the validation data set for pressure drop forecast using gradient variables.

Algorithm	**R^2**	**RMSE**
Linear Regression	0.9851	0.0309
Random forest	0.9874	0.0285
Extra trees	0.9879	0.0279
Gradient boosting	0.9859	0.0301
LSTM	0.9875	0.0283

Table 4.13: Performance of different supervised ML algorithms on the holdout-testing data set for pressure drop forecast using gradient variables.

Algorithm	$\mathbf{R^2}$	**RMSE**
Linear Regression	0.9691	0.0256
Random forest	0.9674	0.0263
Extra trees	0.9666	0.0266
Gradient boosting	0.9662	0.0267
LSTM	0.9635	0.0283

Table 4.14: Time of flooding prediction as given by the trained-supervised ML algorithms for gradient variables as predictors.

Algorithm	**Detection Time**	**Time [mins] Before flooding**	**Time [mins] Before pronounced flooding**
LSTM	9:31	9	50
Random forest	9:36	4	45
Extra trees	9:37	3	44
Gradient Boosting	10:12	-	9
Linear Regression	10:16	-	5

There is no improvement on the previous results obtained using the selected features as predictors. Flooding buildup started at 9:40 on the day of a major flooding event; the previous results show that LSTM gives the earliest detection time at 9:21, 19 minutes before flooding, and 60 minutes before flooding became pronounced in the distillation column. The best result obtained from using the gradient variables as predictors is the detection at 9:31, which shows no improvement over the previous results.

A sample pressure drop forecast at 9:31 is shown in Fig.4.32. This forecast is given 9 minutes before flooding and 50 minutes before flooding fully develops in the column. The models predicted an increased pressure drop for the next 20 minutes. However, only the forecast given by the LSTM can detect flooding, as given by the clustering algorithm, which determines the operating state of the column based on the forecast given.

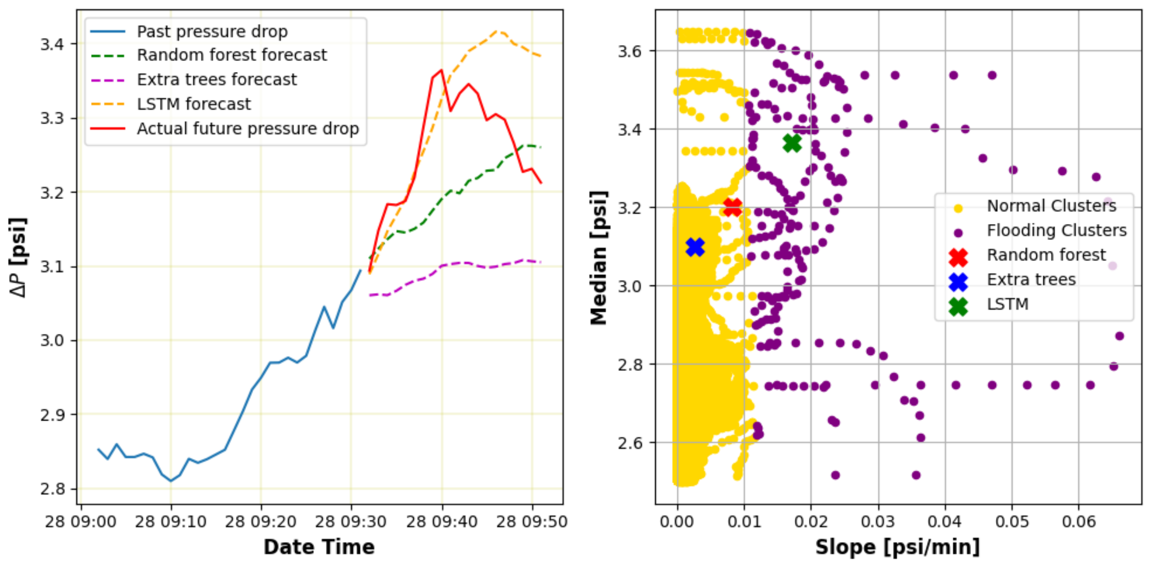


Figure 4.32: Pressure drop forecast at 9:31 (9 minutes before flooding) by the supervised ML algorithms for flooding prediction using the gradient variables as predictors.

4.6.6.3 Material and Energy Balance Indicators

Due to the type of sensor data collected from the distillation column, two variables were developed based on material and energy balance for the distillation column. The variables are based on the errors resulting from the distillation column's total material and energy balance. The total material balance and the energy balance are done based on the available features of the distillation column according to the equations below.

$$M_{bal} = F + (B + D) \tag{4.6}$$

$$E_{bal} = FT_f - (BT_7 + DT_{31}) + \frac{Q_R}{C_{p,s}} \tag{4.7}$$

where F is the feed flow rate, B is the bottom flow rate, D is the distillate flow rate, T_7 is the temperature of tray-7, T_{31} is the temperature of tray-31, Q_R is the heat load on the reboiler, $C_{p,s}$ is the specific heat capacity of naptha used as a scaling value, M_{bal} and E_{bal} are the material balance indicator and the energy balance indicator respectively.

These two indicators are added to the 10 selected features and are used for the pressure drop forecast to detect flooding in the distillation column. The ML algorithms are trained on this new set of predictors, and the performance of the ML algorithms on the validation data set is shown in Table 4.15. The trained ML models have a good performance on the validation data set with a coefficient of determination (R^2) greater than 0.95.

Table 4.15: Performance of different supervised ML algorithms on the validation data set for pressure drop forecast using the material and energy balance indicators as additional predictors.

Algorithm	**R^2**	**RMSE**
Linear Regression	0.9805	0.0357
Random forest	0.9845	0.0318
Extra trees	0.9856	0.0307
Gradient boosting	0.9823	0.0340
LSTM	0.9851	0.0316

The trained ML models are then tested on the holdout-testing data (This testing data set wasn't exposed to the models during training). The performance of the models on the holdout-testing data set is shown in Table 4.16.

Table 4.16: Performance of different supervised ML algorithms on the validation data set for pressure drop forecast using the material and energy balance indicators as additional predictors.

Algorithm	**R^2**	**RMSE**
Linear Regression	0.9600	0.0291
Random forest	0.9425	0.0349
Extra trees	0.9586	0.0296
Gradient boosting	0.9578	0.0299
LSTM	0.9207	0.0412

In conjunction with the trained clustering model, the trained-supervised learning models are used to detect flooding using the flooding data. This dataset was not exposed to the supervised learning models during training. The detection time is shown in Table 4.17. The results show no improvement in the early flooding prediction time of the trained-supervised ML algorithms using the additional material and energy balance indicators. The LSTM gives the earliest detection time at 9:31, which is 9 minutes before flooding and 50 minutes before flooding fully develops in the distillation column on the day of a major flooding event. This is 10 minutes late compared to the normal case of not using the material and energy balance indicators.

Table 4.17: Time of flooding prediction as given by the trained-supervised ML algorithms using the material and energy balance indicators as additional predictors.

Algorithm	**Detection Time**	**Time [mins] Before flooding**	**Time [mins] Before pronounced flooding**
LSTM	9:29	11	52
Random forest	9:52	-	29
Gradient Boosting	10:20	-	1
Linear Regression	10:20	-	1
Extra trees	10:22	-	-

A sample pressure drop forecast at 9:29 is shown in Fig. 4.33. This forecast is given 11 minutes before flooding and 52 minutes before flooding fully develops in the column. The models predicted an increased pressure drop for the next 20 minutes. However, only the forecast given by the LSTM can detect flooding, as given by the clustering algorithm, which determines the operating state of the column based on the forecast given.

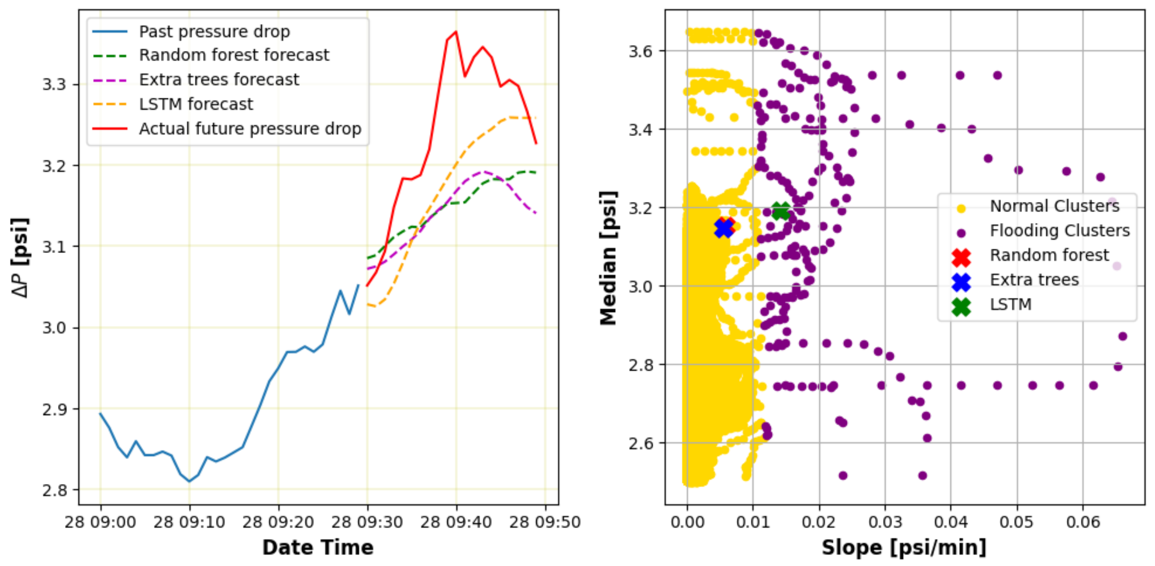


Figure 4.33: Pressure drop forecast at 9:29 (11 minutes before flooding) by the supervised ML algorithms for flooding prediction using the material and energy balance indicators as additional predictors

4.7 Performance Comparison of the ML models

The detection time of the different methods used for flooding detection is given in Table 4.18 using the flooding data collected on the day of the major flooding event in the distillation column as a case study. The conventional autoencoder has the lowest performance in the early detection of flooding in the distillation column. Forecasting the pressure drop using the LSTM model has the best performance since it gives the earliest flooding prediction time. This will create ample time to notify the operators of early flooding risk so that preventive measures can be taken to prevent the full development of the flooding event. Generally, the supervised ML methods, despite being trained with synthetic augmented data, perform better than the unsupervised ML methods.

Table 4.18: Time of flooding detection by different ML methods. The pressure drop forecast is for supervised ML methods.

Method	**Detection time**	**Before Flooding [min]**	**Before Pronounced Flooding [min]**
Pressure Drop Forecast (LSTM)	9:21	19	60
Pressure Drop Forecast (Random Forest)	9:22	18	59
Pressure Drop Forecast (Extra Trees)	9:22	18	59
LSTM Autoencoder	9:27	13	54
PCA-based	9:30	7	50
Conventional Autoencoder	9:45	-	36

Chapter 5

Conclusion & Future work

5.1 Conclusion

One of the most important unit operations in chemical and process industries associated with energy consumption is the distillation column. Poor performance of the distillation column can be attributed to faults such as flooding. Flooding is an abnormal situation that causes liquid build-up in the column, leading to products going off specifications. If left unchecked, the entire production process of the chemical plant can be disrupted.

In this thesis, different ways of early flooding detection in a distillation column for a case of insufficient flooding regime data using ML methods were studied. Early detection of flooding is important so that process operators can take corrective measures to avoid its full development. The method involves the use of unsupervised learning methods such as PCA and Autoencoders to detect flooding and also supervised learning methods to forecast the pressure drop of the distillation column for flooding detection. Using supervised ML algorithm requires sufficient and effective normal operation and flooding data. However, the plant data used in this thesis has limited flooding data, leading to a case of data imbalance with flooding data being the minority. To address this problem, we presented a way of balancing the ratio of normal operation data to flooding data by utilizing time-series generative adversarial networks, known as TimeGAN, to generate synthetic flooding data. The synthetic flooding regime data generated by the TimeGAN are evaluated to confirm that they capture the statistical and predictive properties of the original time-series flooding regime data. From the results obtained, it can be concluded that TimeGAN is effective in generating virtual time-series data that can be used for time-series data scarcity cases.

The results show that early flooding detection in a distillation column is better achieved using supervised ML methods through pressure drop forecasting than unsupervised ML methods. For the supervised ML methods, the trained LSTM model for pressure drop forecast can detect flooding 19 minutes in advance and 60 minutes before flooding fully develops in the distillation column, providing early warning risk of possible flooding events and ample time for the process operators to take corrective measures to prevent the full development of flooding in the distillation column. However, for unsupervised learning, the LSTM Autoencoder can detect flooding 13 minutes in advance.

In summary, it can be concluded for this case study that

- time-series generative adversarial networks (TimeGAN) is an efficient tool to generate synthetic flooding data to address the issue of flooding data scarcity or data imbalance.
- supervised learning methods performed better than unsupervised learning methods in the early detection of distillation column flooding.
- flooding can be detected 19 minutes in advance, based on the result of using LSTM to forecast the pressure drop across the column.

5.2 Future work

The following recommendations are proposed for better prediction of flooding by applying machine learning algorithms to a distillation column where there is insufficient flooding regime data.

- Other types of Machine learning, such as semi-supervised learning methods, can be tested for an improvement over the results of the supervised learning methods for pressure drop forecast.

- Flooding indicators based on empirical correlations can be developed as additional predictors for the supervised learning models to forecast the pressure drop across the column.

Bibliography

[1] N. P. Cheremisinoff, *Handbook of chemical processing equipment*. Elsevier, 2000.

[2] D. S. Rosemount, "Distillation column flooding diagnostics with intelligent differential pressure transmitter", in *Distillation*, 2009. [Online]. Available: `https://api.semanticscholar.org/CorpusID:31565247`.

[3] S. A. Taqvi, L. D. Tufa, H. Zabiri, A. S. Maulud, and F. Uddin, "Multiple fault diagnosis in distillation column using multikernel support vector machine", *Industrial & Engineering Chemistry Research*, vol. 57, no. 43, pp. 14 689–14 706, 2018.

[4] S. A. Taqvi, L. D. Tufa, H. Zabiri, A. S. Maulud, and F. Uddin, "Fault detection in distillation column using narx neural network", *Neural Computing and Applications*, vol. 32, pp. 3503–3519, 2020.

[5] A. A. Ujile and J. T. Iminabo, "Evaluating sieve tray flooding in a distillation column using kister and haas; and fair's correlations", *Chem. Process Eng. Res. 2225-0913*, vol. 25, pp. 16–23, 2014.

[6] Y. Liu, Y. Liang, Z. Gao, and Y. Yao, "Online flooding supervision in packed towers: An integrated data-driven statistical monitoring method", *Chemical Engineering & Technology*, vol. 41, no. 3, pp. 436–446, 2018.

[7] S. L. Brunton, B. R. Noack, and P. Koumoutsakos, "Machine learning for fluid mechanics", *Annual review of fluid mechanics*, vol. 52, pp. 477–508, 2020.

[8] N. O'Mahony, T. Murphy, K. Panduru, D. Riordan, and J. Walsh, "Machine learning algorithms for process analytical technology", in *2016 World Congress on Industrial Control Systems Security (WCICSS)*, IEEE, 2016, pp. 1–7.

[9] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven soft sensors in the process industry", *Computers & chemical engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[10] Y. Liu, C. Li, and Z. Gao, “A novel unified correlation model using ensemble support vector regression for prediction of flooding velocity in randomly packed towers”, *Journal of Industrial and Engineering Chemistry*, vol. 20, no. 3, pp. 1109–1118, 2014.

[11] J. Brockkötter, M. Cielanga, B. Weber, and A. Jupke, “Prediction and characterization of flooding in pulsed sieve plate extraction columns using data-driven models”, *Industrial & Engineering Chemistry Research*, vol. 59, no. 44, pp. 19 726–19 735, 2020.

[12] T. Butters, S. Guttel, J. Shapiro, and T. Sharpe, “Automatic real-time fault detection for industrial assets using metasensors”, 2015.

[13] W. Sun, A. R. Paiva, P. Xu, A. Sundaram, and R. D. Braatz, “Fault detection and identification using bayesian recurrent neural networks”, *Computers & Chemical Engineering*, vol. 141, p. 106 991, 2020.

[14] W. Li, S. Gu, X. Zhang, and T. Chen, “Transfer learning for process fault diagnosis: Knowledge transfer from simulation to physical processes”, *Computers & Chemical Engineering*, vol. 139, p. 106 904, 2020.

[15] G. Van Rossum and F. L. Drake, *Introduction to python 3: python documentation manual part 1*. CreateSpace, 2009.

[16] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, *et al.*, “Array programming with numpy”, *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[17] W. McKinney *et al.*, “Data structures for statistical computing in python”, in *Proceedings of the 9th Python in Science Conference*, Austin, TX, vol. 445, 2010, pp. 51–56.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, “Scikit-learn: Machine learning in python”, *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[19] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, “Scipy 1.0: Fundamental algorithms for scientific computing in python”, *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[20] J. D. Hunter, “Matplotlib: A 2d graphics environment”, *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: `10.1109/MCSE.2007.55`.

[21] L. H. Chiang, E. L. Russell, and R. D. Braatz, *Fault detection and diagnosis in industrial systems*. Springer Science & Business Media, 2000.

[22] V. Venkatasubramanian, “Process fault detection and diagnosis: Past, present and future”, *IFAC Proceedings Volumes*, vol. 34, no. 27, pp. 1–13, 2001.

[23] S. Bailey, “From desktop to plant floor, a crt is the control operators window on the process”, *Control Engineering*, vol. 31, no. 6, pp. 86–90, 1984.

[24] S. Jung, J. Woo, and C. Kang, “Analysis of severe industrial accidents caused by hazardous chemicals in south korea from january 2008 to june 2018”, *Safety science*, vol. 124, p. 104 580, 2020.

[25] R. Isermann and P. Balle, “Trends in the application of model-based fault detection and diagnosis of technical processes”, *Control engineering practice*, vol. 5, no. 5, pp. 709–719, 1997.

[26] M. Madakyaru, F. Harrou, and Y. Sun, “Improved data-based fault detection strategy and application to distillation columns”, *Process Safety and Environmental Protection*, vol. 107, pp. 22–34, 2017.

[27] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, “A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies”, *Computers & chemical engineering*, vol. 27, no. 3, pp. 313–326, 2003.

[28] S. Simani and S. Farsoni, *Fault Diagnosis and Sustainable Control of Wind Turbines: Robust data-driven and model-based strategies*. Butterworth-Heinemann, 2018.

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

[30] R. Isermann, “Model-based fault-detection and diagnosis–status and applications”, *Annual Reviews in control*, vol. 29, no. 1, pp. 71–85, 2005.

[31] J. Chen and R. J. Patton, *Robust model-based fault diagnosis for dynamic systems*. Springer Science & Business Media, 2012, vol. 3.

[32] D. T. Dalle Molle and D. M. Himmelblau, “Fault detection in a single-stage evaporator via parameter estimation using the kalman filter”, *Industrial & engineering chemistry research*, vol. 26, no. 12, pp. 2482–2489, 1987.

[33] R. Isermann, “Process fault detection based on modeling and estimation methods—a survey”, *automatica*, vol. 20, no. 4, pp. 387–404, 1984.

[34] J. V. Beck and K. J. Arnold, *Parameter estimation in engineering and science.* James Beck, 1977.

[35] R. Isermann, *Fault diagnosis in dynamic systems: Theory and applications, chapter 7: Process fault diagnosis based on dynamic models and parameter estimation methods*, 1989.

[36] F. L. Lewis, L. Xie, and D. Popa, *Optimal and robust estimation: with an introduction to stochastic control theory.* CRC press, 2017.

[37] S. X. Ding, *Model-based fault diagnosis techniques: design schemes, algorithms, and tools.* Springer Science & Business Media, 2008.

[38] J. Gertler, “Analytical redundancy methods in fault detection and isolation-survey and synthesis”, *IFAC Proceedings Volumes*, vol. 24, no. 6, pp. 9–21, 1991.

[39] A. S. Willsky, “A survey of design methods for failure detection in dynamic systems”, *Automatica*, vol. 12, no. 6, pp. 601–611, 1976.

[40] E. Chow and A. Willsky, “Analytical redundancy and the design of robust failure detection systems”, *IEEE Transactions on automatic control*, vol. 29, no. 7, pp. 603–614, 1984.

[41] S. X. Ding, *Data-driven design of fault diagnosis and fault-tolerant control systems.* Springer, 2014.

[42] W. A. Shewhart, “Economic quality control of manufactured product 1”, *Bell System Technical Journal*, vol. 9, no. 2, pp. 364–389, 1930.

[43] P. Fasolo and D. E. Seborg, “An sqc approach to monitoring and fault detection in hvac control systems”, in *Proceedings of 1994 American Control Conference-ACC’94*, IEEE, vol. 3, 1994, pp. 3055–3059.

[44] D. C. Montgomery, *Introduction to statistical quality control.* John wiley & sons, 2019.

[45] S. W. Choi, C. Lee, J.-M. Lee, J. H. Park, and I.-B. Lee, “Fault detection and identification of nonlinear processes based on kernel pca”, *Chemometrics and intelligent laboratory systems*, vol. 75, no. 1, pp. 55–67, 2005.

[46] Q. Jiang, X. Yan, and W. Zhao, “Fault detection and diagnosis in chemical processes using sensitive principal component analysis”, *Industrial & Engineering Chemistry Research*, vol. 52, no. 4, pp. 1635–1644, 2013.

[47] R. Muradore and P. Fiorini, “A pls-based statistical approach for fault detection and isolation of robotic manipulators”, *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3167–3175, 2011.

[48] T. Komulainen, M. Sourander, and S.-L. Jämsä-Jounela, “An online application of dynamic pls to a dearomatization process”, *Computers & Chemical Engineering*, vol. 28, no. 12, pp. 2611–2619, 2004.

[49] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers, “Fisher discriminant analysis with kernels”, in *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, Ieee, 1999, pp. 41–48.

[50] J.-M. Lee, S. J. Qin, and I.-B. Lee, “Fault detection and diagnosis based on modified independent component analysis”, *AIChE journal*, vol. 52, no. 10, pp. 3501–3514, 2006.

[51] C. F. Alcala and S. J. Qin, “Reconstruction-based contribution for process monitoring”, *Automatica*, vol. 45, no. 7, pp. 1593–1600, 2009.

[52] D.-S. CHEN, M.-W. LEE, and J. LIU, “Isolating multiple sensor faults based on self-contribution plots with adaptive monitoring”, *China Steel Tech. Rep*, vol. 1, no. 24, pp. 64–73, 2011.

[53] C. Fan, M. Chen, X. Wang, J. Wang, and B. Huang, “A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data”, *Frontiers in energy research*, vol. 9, p. 652 801, 2021.

[54] W. Li, H. Li, S. Gu, and T. Chen, “Process fault diagnosis with model-and knowledge-based approaches: Advances and opportunities”, *Control Engineering Practice*, vol. 105, p. 104 637, 2020.

[55] B. Kuipers, “Qualitative simulation”, *Artificial intelligence*, vol. 29, no. 3, pp. 289–338, 1986.

[56] Q. Shen and R. Leitch, “Fuzzy qualitative simulation”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 4, pp. 1038–1061, 1993.

[57] S.-H. Liao, “Expert system methodologies and applications—a decade review from 1995 to 2004”, *Expert systems with applications*, vol. 28, no. 1, pp. 93–103, 2005.

[58] Y. Qian, X. Li, Y. Jiang, and Y. Wen, “An expert system for real-time fault diagnosis of complex chemical processes”, *Expert Systems with Applications*, vol. 24, no. 4, pp. 425–432, 2003.

[59] D. Chester, D. Lamb, and P. Dhurjati, “Rule-based computer alarm analysis in chemical process plants”, in *Proceedings of the Seventh Annual Conference on Computer Technology*, IEEE Computer Society. Washington DC, 1984, pp. 22–29.

[60] N. Addanki and R. Sethuraman, “Online diagnosis of water chemistry in thermal power plant”, *IFAC Proceedings Volumes*, vol. 25, no. 4, pp. 299–302, 1992.

[61] J. Fussell, “Fault tree analysis-state of the art”, *IEEE Transactions on Reliability*, vol. 23, no. 1, pp. 51–53, 1974.

[62] S. A. Lapp and G. J. Powers, “Computer-aided synthesis of fault-trees”, *IEEE Transactions on Reliability*, vol. 26, no. 1, pp. 2–13, 1977.

[63] A. Rauzy, “Mathematical foundations of minimal cutsets”, *IEEE Transactions on Reliability*, vol. 50, no. 4, pp. 389–396, 2001.

[64] N. H. Ulerich and G. J. Powers, “On-line hazard aversion and fault diagnosis in chemical processes: The digraph+ fault-tree method”, *IEEE Transactions on Reliability*, vol. 37, no. 2, pp. 171–177, 1988.

[65] J. Shiozaki, H. Matsuyama, E. O’shima, and M. Iri, “An improved algorithm for diagnosis of system failures in the chemical process”, *Computers & Chemical Engineering*, vol. 9, no. 3, pp. 285–293, 1985.

[66] T. Umeda, T. Kuriyama, E. O’shima, and H. Matsuyama, “A graphical approach to cause and effect analysis of chemical processing systems”, *Chemical Engineering Science*, vol. 35, no. 12, pp. 2379–2388, 1980.

[67] F. Yang, D. Xiao, and S. L. Shah, "Qualitative fault detection and hazard analysis based on signed directed graphs for large-scale complex systems", *Fault detection*, pp. 15–50, 2010.

[68] F. Qi, B. Huang, and E. C. Tamayo, "A bayesian approach for control loop diagnosis with missing data", *AIChE journal*, vol. 56, no. 1, pp. 179–195, 2010.

[69] C. Rojas-Guzman and M. A. Kramer, "Comparison of belief networks and rule-based expert systems for fault diagnosis of chemical processes", *Engineering Applications of Artificial Intelligence*, vol. 6, no. 3, pp. 191–202, 1993.

[70] P. Weber, D. Theilliol, C. Aubrun, and A. Evsukoff, "Increasing effectiveness of model-based fault diagnosis: A dynamic bayesian network design for decision making", *IFAC Proceedings Volumes*, vol. 39, no. 13, pp. 90–95, 2006.

[71] Z. Zhang and F. Dong, "Fault detection and diagnosis for missing data systems with a three time-slice dynamic bayesian network approach", *Chemometrics and Intelligent Laboratory Systems*, vol. 138, pp. 30–40, 2014.

[72] M. T. Amin, F. Khan, and S. Imtiaz, "Fault detection and pathway analysis using a dynamic bayesian network", *Chemical Engineering Science*, vol. 195, pp. 777–790, 2019.

[73] A. Gorak and H. Schoenmakers, *Distillation: Operation and applications.* Academic Press, 2014.

[74] H. Kister and D. Gill, "Flooding and pressure-drop in structured packings", *CHEMICAL ENGINEER-LONDON*, no. 525, S7–S9, 1992.

[75] J. H. Perry, *Chemical engineers' handbook*, 1950.

[76] H. Kister, *Distillation operation*, 1990.

[77] F. Silvey and G. Keller, "Testing on a commercial scale", *Chem. Eng. Prog*, vol. 62, no. 1, pp. 68–74, 1966.

[78] T. J. Cai, "Column performance testing procedures", in *Distillation*, Elsevier, 2014, pp. 103–154.

[79] N. Lieberman, *Troubleshooting natural gas processing: Wellhead to transmission*, 1987.

[80] H. Z. Kister, "Common techniques for distillation troubleshooting", in *Distillation*, Elsevier, 2014, pp. 37–101.

[81] H. Kister, K. Larson, J. Burke, R. Callejas, and F. Dunbar, "Troubleshooting a water quench tower", in *Proceedings of the 7th Ethylene Producers Conference," Houston, TX*, 1995.

[82] H. Z. Kister and M. Olsson, "An investigation of premature flooding in a distillation column", 2019.

[83] E. Hansuld, L. Briens, and C. Briens, "Acoustic detection of flooding in absorption columns and trickle beds", *Chemical Engineering and Processing: Process Intensification*, vol. 47, no. 5, pp. 871–878, 2008.

[84] S. Parthasarathy, H. Gowan, and P. Indhar, "Prediction of flooding in an absorption column using neural networks", in *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328)*, IEEE, vol. 2, 1999, pp. 1056–1061.

[85] J. Oeing, L. M. Neuendorf, L. Bittorf, W. Krieger, and N. Kockmann, "Flooding prevention in distillation and extraction columns with aid of machine learning approaches", *Chemie Ingenieur Technik*, vol. 93, no. 12, pp. 1917–1929, 2021.

[86] L. M. Ochoa-Estopier, S. Gourvénec, R. Cahors, N. Behara, and J.-B. Scellier, "Prediction of flooding in distillation columns using machine learning", *Digital Chemical Engineering*, vol. 7, p. 100 098, 2023.

[87] R. F. Strigle, "Packed tower design and applications: Random and structured packings", *(No Title)*, 1994.

[88] G. Celata, M. Cumo, G. Farello, and T. Setaro, "Hysteresis effect in flooding", *International journal of multiphase flow*, vol. 17, no. 2, pp. 283–289, 1991.

[89] M. Shoukri, A. Abdul-Razzak, and C. Yan, "Hysteresis effects in countercurrent gas-liquid flow limitations in a vertical tube", *The Canadian Journal of Chemical Engineering*, vol. 72, no. 4, pp. 576–581, 1994.

[90] Z. F. Elsharkawy and M. E. Hammad, "Efficient fault detection and diagnosis of distillation column using gamma scanning", *Journal of Radioanalytical and Nuclear Chemistry*, vol. 316, pp. 741–752, 2018.

[91] J. S. Charlton, *Radioisotope techniques for problem-solving in industrial process plants*. Springer Science & Business Media, 2012.

[92] O. Zahran, H. Kasban, and F. Abd El-Samie, "Utilization of gamma rays for troubleshooting in distillation columns of petrochemical industry", *J. Electron. Electr. Eng*, vol. 2, no. 3, pp. 279–289, 2010.

[93] J. Charlton and M. Polarski, "Radioisotope techniques solve cpi problems", 1983.

[94] W. Severance, "Advances in radiation scanning of distillation columns", 1981.

[95] K. Laraki, R. Alami, R. El Morsli, A. Bensitel, L. El Badri, and E.-M. Hamzaoui, "Diagnosis of distillation column problems using new generation gamma-ray scanning gauge.", *International Arab Journal of Information Technology (IAJIT)*, vol. 5, no. 1, 2008.

[96] A. Jaafar, "Gamma ray scanning for troubleshooting, optimization and predictive maintenance of distillation columns", *Hydrocarbon Asia*, pp. 62–65, 2005.

[97] Z. X. Zhang, G. Y. Wang, Z. H. Yang, X. Yu, H. H. Wang, B. J. Gao, H. T. Zheng, S. L. Zhang, and C. L. Li, "Acoustic signal-based method for recognizing fluid flow states in distillation columns", *Industrial & Engineering Chemistry Research*, vol. 61, no. 48, pp. 17 582–17 592, 2022.

[98] J. W. Boyd and J. Varley, "The uses of passive measurement of acoustic emissions from chemical engineering processes", *Chemical Engineering Science*, vol. 56, no. 5, pp. 1749–1767, 2001.

[99] G.-Y. Wang, Z.-H. Yang, Y. Zhang, H.-H. Wang, Z.-X. Zhang, and B.-J. Gao, "A preliminary fault detection methodology for abnormal distillation column operations using acoustic signals", *Applied Sciences*, vol. 12, no. 24, p. 12 657, 2022.

[100] R. K. Pihlaja and J. P. Miller, *Detection of distillation column flooding*, US Patent 8,216,429, Jul. 2012.

[101] B.-h. Li, B.-c. Hou, W.-t. Yu, X.-b. Lu, and C.-w. Yang, "Applications of artificial intelligence in intelligent manufacturing: A review", *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 1, pp. 86–96, 2017.

[102] A. V. Joshi, *Machine learning and artificial intelligence*. Springer, 2020.

[103] G. James, D. Witten, T. Hastie, R. Tibshirani, *et al.*, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[104] X. Su, X. Yan, and C.-L. Tsai, "Linear regression", *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012.

[105] C. Cortes and V. Vapnik, "Support-vector networks", *Machine learning*, vol. 20, pp. 273–297, 1995.

[106] T. K. Ho, "Random decision forests", in *Proceedings of 3rd international conference on document analysis and recognition*, IEEE, vol. 1, 1995, pp. 278–282.

[107] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost", *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[108] L. Mason, J. Baxter, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent", *Advances in neural information processing systems*, vol. 12, 1999.

[109] S. Ray, "A quick review of machine learning algorithms", in *2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon)*, IEEE, 2019, pp. 35–39.

[110] W. Van Der Aalst and W. van der Aalst, *Data science in action.* Springer, 2016.

[111] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik, "Boosting and other machine learning algorithms", in *Machine Learning Proceedings 1994*, Elsevier, 1994, pp. 53–61.

[112] J. Schmidhuber, "Deep learning in neural networks: An overview", *Neural networks*, vol. 61, pp. 85–117, 2015.

[113] Y. Bengio *et al.*, "Learning deep architectures for ai", *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[114] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects", *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[115] G. W. Milligan and M. C. Cooper, "Methodology review: Clustering methods", *Applied psychological measurement*, vol. 11, no. 4, pp. 329–354, 1987.

[116] T. S. Madhulatha, "An overview on clustering methods", *arXiv preprint arXiv:1205.1117*, 2012.

[117] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning", *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[118] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[119] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, 3. MIT press Cambridge, MA, 2006, vol. 2.

[120] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures", *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[121] Y. Zhuo and Z. Ge, "Auxiliary information-guided industrial data augmentation for any-shot fault learning and diagnosis", *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7535–7545, 2021.

[122] S. Li, J. Luo, and Y. Hu, "Semi-supervised process fault classification based on convolutional ladder network with local and global feature fusion", *Computers & Chemical Engineering*, vol. 140, p. 106 843, 2020.

[123] Z. Wu, W. Lin, and Y. Ji, "An integrated ensemble learning model for imbalanced fault diagnostics and prognostics", *IEEE Access*, vol. 6, pp. 8394–8402, 2018.

[124] P. Peng, W. Zhang, Y. Zhang, Y. Xu, H. Wang, and H. Zhang, "Cost sensitive active learning using bidirectional gated recurrent neural networks for imbalanced fault diagnosis", *Neurocomputing*, vol. 407, pp. 232–245, 2020.

[125] J. Wei, H. Huang, L. Yao, Y. Hu, Q. Fan, and D. Huang, "New imbalanced fault diagnosis framework based on cluster-mwmote and mfo-optimized ls-svm using limited and complex bearing data", *Engineering applications of artificial intelligence*, vol. 96, p. 103 966, 2020.

[126] L. Torgo, P. Branco, R. P. Ribeiro, and B. Pfahringer, "Resampling strategies for regression", *Expert Systems*, vol. 32, no. 3, pp. 465–476, 2015.

[127] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks", *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[128] X. Jiang and Z. Ge, “Data augmentation classifier for imbalanced fault classification”, *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1206–1217, 2020.

[129] Y. Zhuo and Z. Ge, “Gaussian discriminative analysis aided gan for imbalanced big data augmentation and fault classification”, *Journal of Process Control*, vol. 92, pp. 271–287, 2020.

[130] N. Moniz, P. Branco, and L. Torgo, “Resampling strategies for imbalanced time series”, in *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2016, pp. 282–291.

[131] J. Yoon, D. Jarrett, and M. Van der Schaar, “Time-series generative adversarial networks”, *Advances in neural information processing systems*, vol. 32, 2019.

[132] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to linear regression analysis.* John Wiley & Sons, 2021.

[133] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, “Random forests and decision trees”, *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012.

[134] L. Breiman, “Random forests”, *Machine learning*, vol. 45, pp. 5–32, 2001.

[135] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees”, *Machine learning*, vol. 63, pp. 3–42, 2006.

[136] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, “A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2011.

[137] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.* “ O’Reilly Media, Inc.”, 2022.

[138] U. Saeed, S. U. Jan, Y.-D. Lee, and I. Koo, “Fault diagnosis based on extremely randomized trees in wireless sensor networks”, *Reliability engineering & system safety*, vol. 205, p. 107 284, 2021.

[139] A. Natekin and A. Knoll, “Gradient boosting machines, a tutorial”, *Frontiers in neurorobotics*, vol. 7, p. 21, 2013.

[140] J. H. Friedman, “Greedy function approximation: A gradient boosting machine”, *Annals of statistics*, pp. 1189–1232, 2001.

[141] K. Greff, R. K. Srivastava, J. Koutnık, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey”, *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[142] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, *Gradient flow in recurrent nets: The difficulty of learning long-term dependencies*, 2001.

[143] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A novel connectionist system for unconstrained handwriting recognition”, *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2008.

[144] H. Sak, A. W. Senior, and F. Beaufays, “Long short-term memory recurrent neural network architectures for large scale acoustic modeling”, 2014.

[145] E. Marchi, G. Ferroni, F. Eyben, L. Gabrielli, S. Squartini, and B. Schuller, “Multi-resolution linear prediction based features for audio onset detection with bidirectional lstm neural networks”, in *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2014, pp. 2164–2168.

[146] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[147] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with lstm”, *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[148] A. Gomez, “Backpropogating an lstm: A numerical example”, *Aidan Gomez blog at Medium*, 2016.

[149] V. Estivill-Castro, “Why so many clustering algorithms: A position paper”, *ACM SIGKDD explorations newsletter*, vol. 4, no. 1, pp. 65–75, 2002.

[150] A. K. Jain and R. C. Dubes, *Algorithms for clustering data.* Prentice-Hall, Inc., 1988.

[151] P. Makwana, T. Kodinariya, and P. Makwana, “Review on determining of cluster in k-means clustering review on determining number of cluster in k-means clustering”, *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, no. 6, pp. 90–95, 2013.

[152] A. Ng, “Clustering with the k-means algorithm”, *Machine Learning*, pp. 1–2, 2012.

[153] Y. Zhang, Z. Zhou, J. Liu, and J. Yuan, "Data augmentation for improving heating load prediction of heating substation based on timegan", *Energy*, vol. 260, p. 124 919, 2022.

[154] J. M. Lucas and M. S. Saccucci, "Exponentially weighted moving average control schemes: Properties and enhancements", *Technometrics*, vol. 32, no. 1, pp. 1–12, 1990.

[155] T. Villegas, M. J. Fuente, and M. Rodrıguez, "Principal component analysis for fault detection and diagnosis. experience with a pilot plant", *Advances in Computational Intelligence, Man-Machine Systems and Cybernetics*, pp. 147–152, 2010.

[156] J. E. Jackson and G. S. Mudholkar, "Control procedures for residuals associated with principal component analysis", *Technometrics*, vol. 21, no. 3, pp. 341–349, 1979.

[157] I. Karimi and K. Salahshoor, "A new fault detection and diagnosis approach for a distillation column based on a combined pca and anfis scheme", in *2012 24th Chinese Control and Decision Conference (CCDC)*, IEEE, 2012, pp. 3408–3413.

[158] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder-based anomaly detection for indoor air quality time-series data", *IEEE Sensors Journal*, vol. 23, no. 4, pp. 3787–3800, 2023.

[159] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants", in *2018 IEEE international conference on systems, man, and cybernetics (SMC)*, IEEE, 2018, pp. 415–419.

[160] L. Bittorf, N. Böttger, D. Neumann, A. Winter, and N. Kockmann, "Characterization of an automated spinning-band column as a module for laboratory distillation", *Chemical Engineering & Technology*, vol. 44, no. 9, pp. 1660–1667, 2021.

[161] Q. Ai, Q. Liu, W. Meng, and S. Q. Xie, *Advanced rehabilitative technology: neural interfaces and devices*. Academic Press, 2018.