UNIVERSITY OF CAPE TOWN

HYDRAULIC DATA PREPROCESSING FOR ANOMALY BASED INTRUSION DETECTION ON SCADA LEVEL OF WATER TREATMENT SYSTEMS

Student:
Ignitious Vukosi Mboweni
MBWIGN002

Supervisor:
Dr Daniel Ramotsoela
Co-supervisor:
Prof Adnan Abu-Mahfouz

Submitted to the Department of Electrical Engineering at the University of Cape Town in full fulfilment of the academic requirements for the degree

Master of Science in Engineering specialising in Electrical Engineering

December 11, 2023

Declaration

I, Ignitious Vukosi Mboweni, for this dissertation titled, “Hydraulic Data Preprocessing for Anomaly Based Intrusion Detection on SCADA Level of Water Treatment Systems”, declare that:

I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor. Where I have referred to work done by other people, they are acknowledged by citation.

Signed by candidate

_____

Ignitious Vukosi Mboweni

December 11, 2023

UNIVERSITY OF CAPE TOWN

Abstract

Faculty of Engineering and the Built Environment

Department of Electrical Engineering

Master of Science in Engineering specialising in Electrical Engineering

Hydraulic Data Preprocessing for Behavioural Intrusion Detection on SCADA Level of Water Treatment Systems

by Ignitious Vukosi Mboweni

The confidentiality, integrity and availability of critical infrastructure is crucial for any economy to operate efficiently. Critical water systems infrastructure is a target of many attackers who aim to penetrate the system for malicious reasons. The use of cyber-physical systems (CPSs) in Water Treatment Systems (WTSs) unveils many vulnerabilities that attackers can use. Although preventative security mechanisms are put into place they too can be defeated, and in this case, a second layer of security is essential. Intrusion detection mechanisms are important reactive security mechanisms to limit the damage done by a successful attack in the system.

The ability to uncover data patterns and gather knowledge from data is a significant benefit of machine learning (ML), however factors such as noise, missing values, excessive features, and inconsistent and redundant data negatively affects the performance of the model, hence a need for data preprocessing which makes it possible to achieve speed and accuracy on a ML process by unveiling veracity in the data ergo making it valuable. Although many ML techniques for intrusion detection have been studied, comprehensive data preprocessing is scarcely documented. This begets a need for an adoptable data preprocessing workflow specifically for critical water systems infrastructure sensor and actuator data that researchers who intend on working on advancing cyber security in CPSs can utilise.

The work provided in this dissertation explores data preprocessing techniques on secure water treatment (SWaT) testbed data and provides ideal critical water systems infrastructure specific data preprocessing techniques for a resultant informative dataset to yield high results when applied on machine learning (ML) classification models. The SWaT dataset was chosen as it was designed for cyber security research with a WTS use case. The techniques in this study

can be applied to a similar kind of dataset collected from a similar environment and not limited to water treatment.

Experiments were set up to evaluate the effect of preprocessing measures and the results showed good improvement on the model's performance which is a good indication of the impact that the data preprocessing has. The best performance was achieved when the preprocessed dataset was randomly split into training and testing, yielding a significant improvement in accuracy, F1 score and time to detection for both algorithms used in the study, namely Fine Tree and Boosted Trees Ensemble.

Acknowledgements

I give all the glory to the almighty God for the gift of life and leading me through my journey in life which is the reason I was able to complete and submit this dissertation.

I would like to further express my heartfelt gratitude to the following parties:

- The almighty God, for giving me strength and guidance to complete this research.
- Dr Daniel Ramotsoela for his guidance and support.
- My family and friends for their patience and support.
- My colleagues who made this journey very bearable.
- The Telkom Centre of Excellence at the University of Cape Town (UCT) for their resources and support.
- The iTrust centre of research in cyber security and Singapore University of Technology and Design for allowing access to their SWaT dataset.

Contents

Contents 7

List of Figures 10

List of Tables 12

List of Abbreviations 13

Chapter 1. Introduction 15

1.1 Background 15

1.2 Research Questions 16

1.3 Research Objectives 17

1.4 Published Work 17

1.5 Dissertation Structure 18

Chapter 2. Literature review 20

2.1 Cyber-Physical Systems Security 20

2.2 Machine learning in IDSs 23

2.2.1 Supervised vs unsupervised learning 23

2.2.2 Density-based algorithms 23

2.2.3 Parametric vs non-parametric algorithms 24

2.2.4 Classification algorithms 24

2.2.5 Performance evaluation 26

2.2.6 Misclassification cost 28

2.2.7 Validation schemes 29

2.2.8 Multistage classification 29

2.2.9 Ensemble learners 31

2.3 Data preprocessing 33

2.3.1 Missing data imputations 33

2.3.2 Outlier filtering 34

2.3.3 Noise reduction 35
2.3.4 Feature selection 36
2.3.5 Feature extraction 37
2.4 Comparison and discussion 41
2.5 Conclusion 44
Chapter 3. Data 45
Chapter 4. Methodology 50
4.1 Modelling approach 50
4.1.1 Problem Exploration 51
4.1.2 Problem formulation 52
4.1.3 Baseline models 52
4.1.4 Data Preprocessing 55
Chapter 5. Application 59
5.1 Computing resources and parallel computing 59
5.2 Modelling approach 60
5.2.1 Problem Exploration 60
5.2.2 Formulating the problem 70
5.2.3 Baseline Models 70
5.2.4 Data Preprocessing 71
5.3 Summary 96
Chapter 6. Experiments and results 97
6.1 Baseline models 97
6.2 Experiment 1: Models trained on new dataset 97
6.3 Experiment 2: Feature scaling 98
6.4 Experiment 3: PCA applied 99
6.5 Experiment 4: Randomly partitioned data 99
Chapter 7. Discussion 101

7.1 Experiment 1: Models trained on new dataset 101

7.2 Experiment 2: Feature scaling 101

7.3 Experiment 3: PCA applied 102

7.4 Experiment 4: Random partitioned data 102

7.5 What can be learnt from the results? 102

7.6 Limitations and recommendations 102

Chapter 8. Conclusion 104

References 105

Appendix A: Results 115

Appendix B: Dataset reduction by correlation analysis 120

List of Figures

Figure 1-1. Illustration of connectedness between machine learning and cyber-security in critical water systems infrastructure. 16
Figure 2-1. Left: A two-dimensional partitioned feature space. Right: Decision tree corresponding to the partitioning of the feature space. 25
Figure 2-2. Confusion Matrix 26
Figure 3-1.Picture of SWaT testbed [70] 45
Figure 3-2.SWaT's six stage processes [71] 47
Figure 3-3. SWaT's HMI/SCADA interface [71] 48
Figure 3-4. Attacks in SWaT dataset 49
Figure 4-1. ML modelling workflow 50
Figure 4-2. Attack observed on LIT301 sensor data 51
Figure 4-3. Modelling approach for training and evaluating ML models 53
Figure 4-4. Classification Tree TTD 54
Figure 4-5. Data processing steps 55
Figure 4-6. Representation of rational transfer function using its direct-form II transposed implementation [81]. 56
Figure 5-1. CPU partition 60
Figure 5-2. Attacks on level transmitter 61
Figure 5-3. Left: Attacks on motorised valve MV10. Right: Attacks on level transmitter LIT101 61
Figure 5-4. Left: Attacks on pump P101. Right: Attacks on level transmitter LIT301 62
Figure 5-5. Left: Attacks on dechlorinator actuator UV401. Middle: Attacks on RO pH analyser sensor AIT501. Right: Attacks on pump P501 63
Figure 5-6. Test data attacks 71
Figure 5-7. LIT101 sensor data process for extracting slopes. 72
Figure 5-8. Plots of LIT101 raw and filtered data 73
Figure 5-9. LIT101 local minima of high prominence 73
Figure 5-10. Plot of FIT101 75
Figure 5-11. FIT101 zoomed in to show sinusoidal structure 75
Figure 5-12. FIT101 signal after moving mean smoothing. 76
Figure 5-13. Power spectrum computed from a window on FIT101 data. 76
Figure 5-14. Plot of AIT201 77

Figure 5-15. Plot showing AIT201 raw and filtered data. 78
Figure 5-16. Plot of AIT202 78
Figure 5-17.Plot showing AIT202 raw and smoothed and detrended data. 80
Figure 5-18. Left: Plot of raw AIT202 data. Right: Plot of smoothed and detrended AIT202 Data. 80
Figure 5-19. Plot of processed AIT201 data showing local minimum and maximum extrema. 81
Figure 5-20. Plot of AIT203. 81
Figure 5-21. Plot showing AIT203 and extracted features. 82
Figure 5-22. Plot of FIT201 83
Figure 5-23. Plot of processed FIT201 data showing local maximum extrema. 83
Figure 5-24. Plot of FIT301 84
Figure 5-25. Plot showing FIT301 and extracted features. 85
Figure 5-26. Plot of LIT301. 85
Figure 5-27. Plot showing LIT301 and extracted features. 86
Figure 5-28. Plot of LIT301 87
Figure 5-29. AIT402 MATLAB processing code 87
Figure 5-30. Plots of raw and smoothed AIT402 data. 88
Figure 5-31. Plots of raw and smoothed FIT401 data. 88
Figure 5-32. Plot of LIT401 89
Figure 5-33. Plot showing LIT401 and extracted features. 90
Figure 5-34. Plots of raw and smoothed AIT401 data. 90
Figure 5-35. Plots of raw and smoothed AIT502 data. 91
Figure 5-36. Plots of raw and smoothed AIT503 data. 92
Figure 5-37. Plots of raw and smoothed AIT504 data. 92
Figure 5-38. Full plots of raw and smoothed PIT502 data. 93
Figure 5-39.Close-up plots of raw and smoothed PIT502 data. 93
Figure 5-40.Close-up plot of FIT601 data. 94
Figure 5-41. Workflow for treating process data from industrial control systems. 96
Figure 6-1. Experiment 1 approach 98
Figure 6-2. Experiment 2 approach 98
Figure 6-3. Experiment 3 approach 99
Figure 6-4. Experiment 4 approach 100

List of Tables

Table 2-1. Summary of vulnerabilities. C: Cyber, CP: Cyber-Physical, P: Physical, I: Isolation Assumption, Con: Connectivity, O, Openness, H Heterogeneity, and S: Many Stakeholders 21
Table 2-2. Communication schemes in IDSs, their drawbacks, and advantages 22
Table 2-3.Methods used by the top 4 BATADAL competition teams and their results on 7 attack 31
Table 2-4. Classification results for IRIS dataset obtained by Ashouri et al. [65]. 40
Table 2-5. Comparison of reviewed methods 44
Table 3-1.Description of sensors and actuators used in SWaT [8] 45
Table 5-1.Table showing pump pairs (main and backup) and resultant variable. 63
Table 5-2. Correlation coefficients among variables in stage P1. 64
Table 5-3. Correlation coefficients among variables in stage P2. 65
Table 5-4. Correlation coefficients among variables in stage P3. 66
Table 5-5. Correlation coefficients among variables in stage P4. 67
Table 5-6. Correlation coefficients among variables in stage P5 (table too wide and is split in 2). 68
Table 5-7. Dataset variables at this stage. 69
Table 5-8.Model parameters: 71
Table 5-9. Dataset variables at this stage. 95
Table 6-1. Baseline results 97
Table 6-2. Experiment 1 results 98
Table 6-3. Experiment 2 results 99
Table 6-4. Experiment 3 results 99
Table 6-5. Experiment 4 results 100

List of Abbreviations

AD	Anomaly Detection
ANN	Artificial Neural Network
BATADAL	BATtle of the Attack Detection Algorithms
CI	Critical Infrastructure
CPS	Cyber-Physical Systems
DB	Density-Based
DMA	District Metered Areas
FDI	Fault Detection and Isolation
ICS	Industrial Control Systems
IDS	Intrusion Detection System
k-NN	k-nearest neighbours
LDA	Linear Discriminant Analysis
LOF	Local Outlier Factor
LR	Logistic Regression
LUS	Local Unimodal Sampling
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSI	Multispectral Imaging
MSMP	Multi Stage Multi Point
MSSP	Multi Stage Single Point
ORP	Oxidation Reduction Potential
PCA	Principal Component Analysis
PLC	Programmable Logic Controllers
PSO	Particle Swarm Optimization
QDA	Quadratic Discriminant Analysis
RMSE	Root Mean Square Error
RNN	Recurrent Neural Networks
RO	Reverse Osmosis

SCADA	Supervisory Control and Data Acquisition
SOD	Subspace Outlier Degree
SSMP	Single Stage Multi Point
SSSP	Stage Single Point
SVM	Support Vector Machine
SWaT	Secure Water Treatment
TNR	True Negative Rate
TPR	True Positive Rate
TTD	Time to detection
UF	Ultrafiltration
WDS	Water Distribution Systems
WTS	Water Treatment Systems
WMA	Weighted Majority Algorithm

Chapter 1. Introduction

The data preprocessing step is essential in the ML) modelling process as it enhances the computational efficiency and model accuracy while preserving the discriminatory information in the dataset. The sensor and actuator data obtained from industrial control systems (ICS) such as supervisory control and data acquisition (SCADA) is intricate, inconsistent, and non-linear, requiring the application of sophisticated preprocessing techniques. This study proposes the utilization of ML for detecting attacks in a Water Treatment System, with a particular emphasis on the processing of hydraulic component data, such as sensors that measure pressure, flow rate, temperature and position, and actuators such as pumps, valves, and cylinders.

1.1 Background

WTSs are a critical element of critical infrastructure (CI), playing a vital role in promoting public health, well-being, productivity, and functionality by providing access to safe drinking water. These systems are essential in enabling communities to lead healthy, productive, and functional lives and thus require appropriate protection. With the advancement of technology, modern WTSs have become more flexible and adaptable, utilizing cyber-physical systems (CPSs) that allow for the integration of physical component monitoring and control with computing and communication [1], [2].The integration of smart water networks supports the development of future smart cities. However, the use of CPS communication schemes also introduces vulnerabilities that can be exploited by attackers to gain access to the SCADA system for malicious purposes.

Traditionally, security was built on the premise of separating systems from the rest of the world and performing monitoring and control locally [3]. However, the use of off-the-shelf components in SCADA systems, instead of proprietary software and hardware, has created new vulnerabilities to threats and attacks. This is because legacy control systems and their communication methods are inherently insecure and prone to attacks [3], [4].

Attacks in CPSs take the form of intrusions, which are deliberate efforts to compromise the system's integrity, confidentiality, or availability [5]. While preventive security measures are in place, they can be circumvented, thereby requiring the implementation of reactive mechanisms to support the recovery phase. This phase is vital in addressing the consequences of a successful attack and ensuring the system's return to a secure state [6]. Anomaly detection (AD) is a popular behavioural intrusion detection technique that classifies system behaviour as

normal or anomalous by analysing data. This technique allows for the detection of both known and unknown attacks by creating a profile of normal behaviour for the system and flagging deviations as anomalous and potentially malicious. Figure 1-1 provides a visual illustration of how ML fits together with CPSs and critical water systems infrastructure.

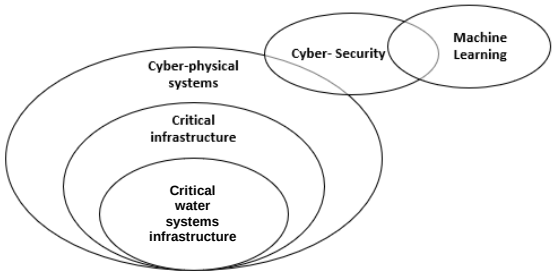

Figure 1-1. Illustration of connectedness between machine learning and cyber-security in critical water systems infrastructure.

1.2 Research Questions

Studies on intrusion detection have been abundant in the literature, yet only a fraction of them have been conducted on critical water systems infrastructure. Furthermore, research specifically focused on data preprocessing in this area is limited. This is evidenced by the BATtle of the Attack Detection Algorithms (BATADAL), which showcases state-of-the-art algorithms for cyber-physical attack detection. The participants in the event employed multi-stage detection techniques and dimensionality reduction on sensor data, aimed at detecting both global and local anomalies [7]. However, the event's documentation was deficient in covering the data preprocessing aspect of machine learning modeling, as the focus was solely on the classification techniques.

Secure water treatment (SWaT) is a water treatment testbed for cybersecurity research, it consists of a scaled down six-stage water treatment process that is almost indistinguishable to a real-world treatment plant [8]. The SWaT dataset was collected from 7 days of normal operation and 4 days of operation with attack scenarios. Given that sensor data in its nature is noisy, complex, and non-linear, the proposed work is to formulate ideal preprocessing techniques for sensor and actuator data in critical water systems infrastructure. The proposed work will formulate ideal preprocessing techniques for hydraulic components data, which is data from sensors and actuators which measure and control elements in the water treatment process.

The execution of this study answers the following questions:

1. What baseline model can be chosen for comparison and to build upon?
2. Data understanding
    a. What features will be good predictor variables?
    b. Can visual trends and patterns be identified in the data?
3. Preprocessing techniques understanding
    a. What preprocessing techniques can be applied to this dataset?
    b. Do the techniques improve the ML model performance?
4. Dimensionality reduction
    a. Does a dimensionality reduction technique work well for this dataset?
    b. Can the benefit of dimensionality reduction be quantified in detection time and accuracy?

1.3 Research Objectives

This study brings attention to a crucial part of ML modelling that is often not discussed enough in literature. To address the above research questions, the following approach is followed:

- Conduct an extensive literature study that covers cyber security in CPSs and machine learning modelling approaches with a stealthy focus on data preprocessing.
- Visualise the data by means of plots to uncover patterns in the data.
- Conduct correlation analysis to determine the relationship between the different variables.
- Visualise and study the variables thereafter using knowledge from literature, intuition, and discretion to apply feature extraction.
- Set up different experiments to investigate the effects of feature extraction, feature scaling, dimensionality reduction, and data splitting methods.

1.4 Published Work

1. Conference paper

I. V. Mboweni, D. T. Ramotsoela, and A. M. Abu-Mahfouz, “A machine learning approach to intrusion detection in water distribution systems – A review,” 2021, doi: 0.1109/IECON48115.2021.9589237.

2. Journal Article

I. V Mboweni, D. T. Ramotsoela, and A. M. Abu-mahfouz, “Hydraulic Data Preprocessing for Machine Learning Based Intrusion Detection in Cyber-Physical Systems,” MDPI, Math. Spec. issue "Application Data Anal. to Netw. Secur. ", pp. 1–20, 2023.

1.5 Dissertation Structure

This dissertation is structured to address the research questions outlined in above in an orderly manner and establishes a continuous flow when introducing and addressing key concepts. The structure of this dissertation is as follows:

Chapter 1: Introduction

This chapter introduces the work carried out in this study. It gives the background to the study, the research objective that need to be achieved in the study, the research questions that need to be answered though the technical work carried out in the study and dissertation structure.

Chapter 2: Literature review

This chapter embodies the evaluation of academic literature. It starts by giving context on current security measures used to protect cyber-physical systems. Thereafter machine learning strategies in intrusion detection systems are reviewed with a great focus on the data preprocessing stage of machine learning modelling.

Chapter 3: Data

The SWaT dataset used in the study is firstly discussed in this chapter. Firstly, a background is provided, followed by the data collection process and the physical setup, and finally the attack scenarios in the dataset.

Chapter 4: Methodology

The methodology used in the study is explained in this chapter. The aim is to provide a general workflow that can be applied to other intrusion detection systems for critical water systems infrastructure with different datasets. The methodology looks at the modelling approach which entails formulating the problem, setting up baseline models as a benchmark, preprocessing the data, splitting it, and training the models on it. Once the models have been trained, they are then evaluated on selected criteria which include the accuracy, misclassification cost, training time and time to detection of an attack.

Chapter 5: Application

The data is processed following a workflow and each feature is studied and processed uniquely. The workflow includes steps that address missing data imputation, outlier filtering, noise reduction, feature extraction, feature scaling and dimensionality reduction.

Chapter 6: Experiments and results

Five experiments were conducted to determine the impact of the data processing strategies applied in chapter 5 on the ML algorithms learning ability based on the evaluation criteria used.

Chapter 7: Discussion and conclusion

This is the final chapter where the results obtained in chapter 6 are discussed, and the study is concluded by summarising the objectives of the study and how they were achieved.

Chapter 2. Literature review

This literature review discusses various topics to give a comprehensive construction of the study. Existing research was studied and relevant work to this study was attained. This section first introduces security measures implemented in CPSs and where the vulnerabilities lie. Machine learning in intrusion detection systems (IDSs) is then introduced, this section covers ML concepts, the ML modelling process and outlines how other researchers have used ML techniques in intrusion detection. The ML methods discussed are compared and discussed in the end.

2.1 Cyber-Physical Systems Security

Cybersecurity is intended to secure networks, devices, programs, and data from potential threats. A threat refers to unauthorized access to a system that takes advantage of its vulnerabilities with the aim of compromising any of the three fundamental security requirements.

Types of threats:

1. Denial of service
2. Unauthorised access/integrity breach

According to the literature presented in [3], the growth of cyber-physical systems has given rise to new security challenges. One of the difficulties in ensuring the security of CPSs is the heterogeneity of the constituent components. Understanding the vulnerabilities, threats, and attacks is crucial in the development of effective defense mechanisms.

Every threat possesses these five attributes:

- Source – The initiator of the attack
- Target – CPS component or function
- Motive – Criminal, spying, terroristic, etc.
- Attack vector – Interception, interruption, modification, or fabrication
- Consequence – Unpleasant result of the attack. Compromised integrity, confidentiality, or availability

For a system to be secure, it satisfies the three security requirements: confidentiality, integrity and availability [6].

- Confidentiality – Protecting its contents so that unauthorised access to them is not possible.
- Integrity – The contents are not changed/ altered by an unauthorised user.
- Availability – Authorised users always have access to the contents whenever requested.

The requirements that an IDS should satisfy when designed are given by [9]. These requirements are:

- Not add to the system's weaknesses,
- Require minimal system resources and have no negative effects on overall system performance,
- Be consistently available and remain transparent to the system and the users,
- Use standards to be cooperative and open,
- In the detecting phase, be dependable and reduce false positives and negatives.

A vulnerability is a weak point in a system that an attacker can exploit to gain unauthorized access. In ICSs such as SCADA, these vulnerabilities are documented and listed, as demonstrated in Table 2-1 of [3]. It is important to note that the identification and mitigation of vulnerabilities is a continuous process that requires ongoing assessment and updates to ensure the security of the system.

Table 2-1. Summary of vulnerabilities.
C: Cyber, CP: Cyber-Physical, P: Physical, I: Isolation Assumption, Con: Connectivity, O, Openness, H Heterogeneity, and S: Many Stakeholders

CPS	Vulnerability	Type	Cause
ICS	Open communication protocols	C	I, O
	Wired communications	C	I, H, S
	Wireless communications	C	I, Con
	Web-based attacks	C	Con, H
	Insecure protocols	CP	I, Con
	Interconnected & exposed field devices	CP	I, Con
	Insecure secondary access points	CP	I, Con
	Insecure OS & RTOS	CP	I, Con, H
	Software	CP	Con, H
	Equipment’s physical sabotage	P	I

The communication aspect of cyber-physical systems is crucial for the monitoring and control of SCADA systems. As highlighted in Table 2-1, communication schemes have inherent

vulnerabilities. Table 2-2 provides a comparison of the advantages and disadvantages of popular communication schemes. This information is useful in selecting a suitable communication scheme that balances the requirements of the SCADA system with the available resources and constraints.

Security tools can be categorized into proactive and reactive mechanisms, as outlined in [10]. Proactive security mechanisms, as listed in the table, aim to reduce the likelihood of a successful attack. Reactive mechanisms, such as intrusion detection schemes, play a key role in addressing the consequences of an attack and ensuring system recovery, as complete prevention of all attacks is not always feasible [7]. Detection and response, as part of security engineering, is a crucial aspect of ensuring system security [11].

Table 2-2. Communication schemes in IDSs, their drawbacks, and advantages

Communication	Drawbacks	Benefits
Cyber		
TCP/IP	• Open standards protocols – vulnerable protocol [SP]	Encryption authentication measures
ICCP	• Open standards protocols – vulnerable protocol • The security is quite basic, lacking any encryption or authentication measures. [12]	Easily adoptable standard across multiple control centre energy management systems including SCADA
RPC	Usually requires the TCP/IP protocol which suffers its own security drawbacks	Data Encryption Standard (DES) encryption
Cellular (GSM)	• Vulnerability to the man in the middle attack	Multiple security measures put in place
Microwave	• Vulnerable to tapping	Useful wireless communication method
Satellite	• Uses open telecom network security protocols	• logical access at base stations • encryption authentication measures
Cyber physical		
Modbus	• Lacks encryption [13] • Lacks integrity checks [13] No authentication measures are implemented [14] [1]	It is a streamlined protocol that ensures fast data transmission
DNP3	• Lacks encryption • No authentication measures are implemented	CRC integrity check

2.2 Machine learning in IDSs

Zimek and Filzmoser [15] say that outlier detection has been of importance in data mining and in the past methods were based on statistical reasoning which eventually lost their aptness in determining outliers. Today many methods exist to detect outliers in datasets. Popular ML anomaly detection techniques are density based (DB), parametric and classification methods. In this section the difference between supervised and unsupervised learning is explained and then anomaly detection techniques are introduced.

2.2.1 Supervised vs unsupervised learning

a. Supervised learning

A method used to train an algorithm based on labelled data. The algorithm learns to predict outcomes or classify data by trying to fit itself to the data, this is done by adjusting weights. Supervised learning can be divided into two categories, namely classifications and regression which have various algorithms discussed later in the thesis. Understanding the problem and dataset at hand makes for good decision making when choosing the type of algorithm to apply.

b. Unsupervised learning

In ML unlike supervised learning, the data used here is unlabelled. It is when the model is allowed to work on its own without any prior knowledge input for the purpose of learning and uncovering unknown patterns in the data. Unsupervised learning models are used for clustering, association, and dimensionality reduction of which dimensionality reduction and will be discussed in the thesis as a valuable part of dimensionality reduction.

2.2.2 Density-based algorithms

Density based algorithms such as local outlier factor (LOF) and k-Nearest Neighbours (k-NN) are algorithms that rely on the density/distance of the individual points relative to each other.The DB algorithms, such as LOF and k-NN, operate by assessing the relative proximity of individual data points to each other based on density. In their research, Knorr et al. [16] investigated the utilization of density-based nearest neighbour outlier detection techniques, and devised several methods for identifying outliers in large multidimensional datasets. Their study encompassed three real-world applications: analysing NHL player statistics, a video surveillance system, and workers' compensation claims. Another example of a density-based technique is the Local Outlier Factor, introduced by Breunig et al. [17], which measures the degree of isolation of an object from its neighbours.

2.2.3 Parametric vs non-parametric algorithms

a. Parametric

Parametric methods use known distributions (e.g., Poisson or Gaussian) and fit the data to the model by learning its parameters. Examples include linear regression, Naïve Bayes, and simple neural networks. These methods are a useful tool for modelling complex systems and making predictions based on the data. However, they are limited by their reliance on the underlying assumptions about the distribution of the data, and their performance may suffer if the assumptions are not accurate [15].

b. Non-parametric

In contrast to parametric methods, non-parametric methods do not rely on a predetermined distribution of the data. Instead, the parameters are not fixed and a hypothesis test is necessary for the dataset. Although not always entirely free of parameters, non-parametric methods may still require the user to specify certain parameters. A notable example of a non-parametric algorithm is the k-NN method, which was previously mentioned as a density-based algorithm.

2.2.4 Classification algorithms

Classification, a form of supervised learning, is a method of grouping data into different classes based on pattern understanding. The training datasets are predetermined/labelled, and the classification algorithms predict the probability that new data fall into one of the predetermined categories. Classification methods are discussed in this section and their applications in studied literature, and they form a baseline of the direction of this research.

a. Decision Trees

Decision trees are popular machine learning techniques for classification, it is an intuitive method used to represent decisions and decision making by hierarchically partitioning the input space to a point where the class label's subspace is reached. Each individual node of a decision tree represents a splitting decision and each leaf node is linked to a class label prediction [18]. A decision tree works by traversing the data from the root to the leaf of the tree based on splitting rules. This algorithm can be used with both numerical and categorical attributes which take away one step in the preprocessing phase.

Figure 2-1 below shows a simple decision tree applied on a feature space using recursive binary spitting on to yield a two-dimensional partitioned space [19]. The feature space consisting of x_1 and x_2 is split into five distinct and non-overlapping regions (*R1, R2, …, R5*) and a decision

tree shown on the right is the result. New predictions can be associated to the regions following the tree's logic.

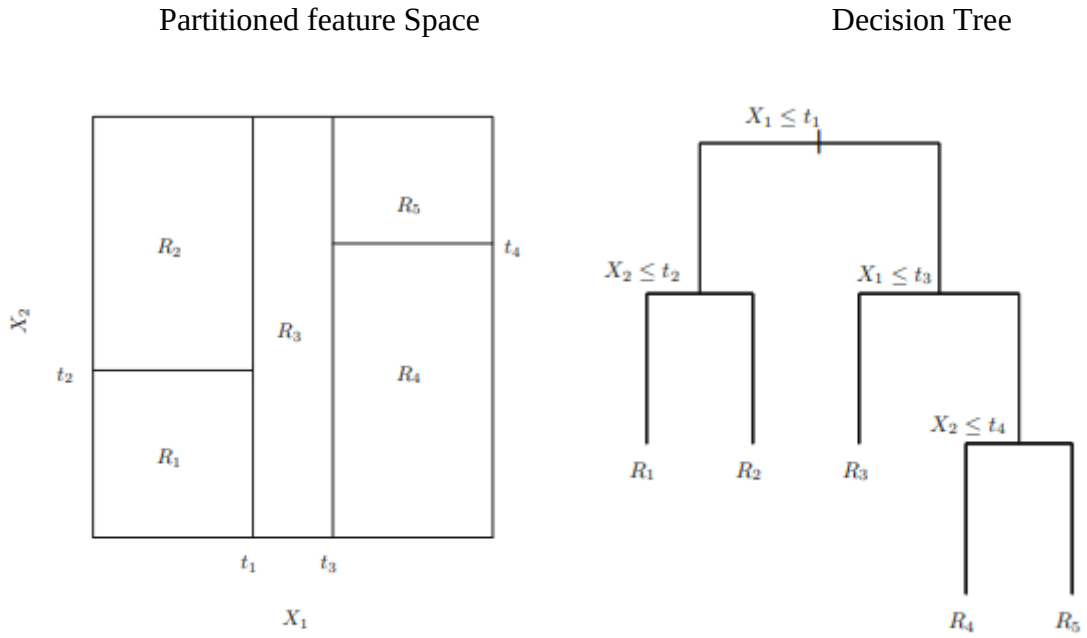

Figure 2-1. Left: A two-dimensional partitioned feature space. Right: Decision tree corresponding to the partitioning of the feature space.

In [20], a mixture of bagging and boosting technique for a decision tree was applied to a KDD99 dataset which is a popular dataset for IDS studies and compared to results of a Naïve Bayes algorithm. The results were comparable however the decision tree yielded better classification results. However, computationally the construction of a Naïve Bayes is faster than the decision tree.

b. Discriminant Analysis

Another popular classification technique is discriminant analysis, developed by Ronald Fisher in 1936 as a statistical approach to classify or discriminate observations into groups that do not overlap based on ratings awarded to one or more measurable variables used for the prediction. There are two main versions of this technique, namely Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) [21]. The two versions are closely related since they both use Gaussian assumptions in their classification. A notable difference being that given predictor classes, LDA presupposes that their covariance matrices are the same while QDA permits covariance matrices that are different. These two methods respectively result in linear and quadratic decision boundaries. LDA was applied in intrusion detection by Subba et. al [22] and their result showed that it is comparable in performance to logistic regression (LR),

support vector machine (SVM) and C4.5 based models but outperformed Naïve Bayes method based model.

2.2.5 Performance evaluation

The decision-making process in IDSs involves the evaluation of its performance using a confusion matrix shown in Figure 2-2. This matrix is a visual representation that succinctly summarizes the performance of a classification algorithm. It provides a comprehensive evaluation by breaking down the results into several categories, such as true positive, false positive, true negative, and false negative. The use of confusion matrices in the evaluation of IDS performance enables a thorough assessment of the algorithm's accuracy and helps to identify areas for improvement. In essence, the confusion matrix plays a crucial role in the development and refinement of IDS algorithms, as it provides valuable insights into the strengths and weaknesses of the system.

- False-negative (FN): The system has an intrusion, however the IDS fails to detect it and in turn labels the event as non-anomalous.
- False-positive (FP): The IDS wrongly identifies a normal event as anomalous.
- True-negative (TN): The IDS correctly identifies a normal event as non-anomalous.
- True-positive (TP): The IDS correctly identifies an anomalous event as such.

		Predicated Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Figure 2-2. Confusion Matrix

The evaluation of the performance of intrusion detection in SCADA systems can be measured through two crucial criteria, time-to-detection (TTD) and classification accuracy. TTD refers to the interval between the moment an attack is detected t_d and when it began t_0. It quantifies the speed at which the algorithm can identify a threat, as defined in the equation below.

$$\boldsymbol{TTD = t_d - t_0} \qquad \textit{2-1}$$

The metric S_{TTD} is a useful evaluation criterion for comparing the TTD performance of different algorithms. This metric is expressed as the ratio between the total duration of all detected attacks and the total duration of all attacks, as shown in Equation 2-2. Its value ranges from 0 to 1, with a score of 1 indicating that all attacks were detected and a score of 0 indicating that none of the attacks were detected. The TTD of the i^{th} attack is represented by TTD_i, the

duration of the i^{th} attack is represented by Δt_i, and n_a represents the total number of attacks in the dataset.

$$S_{TTD} = 1 - \frac{1}{n_a}\sum_{i}^{n_a}\frac{TTD_i}{\Delta t_i} \quad 2\text{-}2$$

In binary classification, the performance of the intrusion detection system is evaluated using metrics such as True Positive Rate (TPR), also known as Recall, and True Negative Rate (TNR), these metrices are shown in Equations 2-3 and 2-4 below. The TPR measures the system's ability to correctly identify a threat, while TNR measures its ability to avoid false alarms. The average of TPR and TNR, known as the S_{CLF} score shown in Equation 2-5, can be used to compare the performance of different algorithms and ranges from 0 (worst) to 1 (perfect classification). When dealing with imbalanced datasets, an alternative metric called the F1 score in Equation 2-7 can be used, which takes into account both Recall and Precision. Precision in Equation 2-6 is the proportion of correctly predicted positive observations to all expected positive observations. The F1 score also ranges from 0 to 1 and provides a more comprehensive evaluation of the system's performance [23].

$$TPR = Recall = \frac{TP}{TP+FN} \quad 2\text{-}3$$

$$TNR = \frac{TN}{FP+TN} \quad 2\text{-}4$$

$$S_{CLF} = \frac{TPR+TNR}{2} \quad 2\text{-}5$$

$$Precision = \frac{TP}{TP+FP} \quad 2\text{-}6$$

$$F1\ score = \frac{2*(Precision*Recall)}{Precision+Recall} \quad 2\text{-}7$$

The overall performance of an intrusion detection system in a SCADA environment can be evaluated by combining the time-to-detection (TTD) and classification accuracy. The *S* score, which is a weighted combination of these two metrics, can be used to rank the performance of different algorithms. The weighting factor, γ ($0 \leq \gamma \leq 1$), can be adjusted to reflect the relative importance of TTD and classification accuracy. When both criteria are considered to be equally important, γ is set to 0.5.

$$S = \gamma \cdot S_{TTD} + (1-\gamma) \cdot S_{CLF} \quad 2\text{-}8$$

The subsequent steps taken after the detection of an anomaly play a crucial role in determining the true nature of the anomaly [24]. This is demonstrated by the 2015 cyberattack on the Ukrainian power grid where hackers remotely switched off 30 substations and left 230,000 people without electricity. The operators had to manually switch the substations back on, which

took several hours. Therefore, the nature of the anomaly and the response of the system operators had a significant impact. It is essential to determine the type of anomaly, such as a cyberattack, hardware failure or human error in order to properly address it [7]. However, the behaviour-based intrusion detection approach is prone to generating false alarms, and therefore, this issue must also be resolved [23].

The appropriate approach to handle outliers is contingent upon the nature of the data and the problem at hand. In the case of labelled data collected from a WTS via programmable logic controllers (PLCs) and a SCADA system, supervised learning through classification is deemed an effective strategy. Several commonly used classification algorithms exist, including logistic regression, Naive Bayes, stochastic gradient descent, kNN, decision tree, random forest, and support vector machines (SVM). However, it is important to note that not all of these algorithms are suitable for anomaly detection. A plethora of anomalies detection methods have been proposed in the literature, including kNN, SVM, Neural Networks, Bayesian networks, Hidden Markov Models (HMMs), and ensemble techniques.

2.2.6 Misclassification cost

Misclassification cost put simply is the penalty put forth on the classification algorithm for errors in the classification process [25]. Often datasets in real world applications are imbalanced, that is, the dataset is skewed towards one predictor class since the number of observations in one class outweigh the number of observations in the other class. Data imbalance results in misclassification in cost-sensitive methods where class labels are incorrectly predicted due to the classification algorithm being biased towards the major class. This problem is common in many applications and one that is of interest to this study is intrusion detection [26].

In other applications consequences of misclassifying observations of one class are more significant than misclassifying observations of another class. In fraud detection by way of illustration, the cost of classifying a fraudulent transaction as non-fraudulent transactions (false negative) have more severe consequences than classifying non-fraudulent transactions as fraudulent transaction (false positive), in this case higher cost is assigned to classifying a fraudulent transaction as non-fraudulent transactions. A cost matrix for a confusion matrix is given as:

$$\boldsymbol{C} = \begin{pmatrix} \boldsymbol{C_{00}} & \boldsymbol{C_{01}} \\ \boldsymbol{C_{10}} & \boldsymbol{C_{11}} \end{pmatrix} \qquad \textit{2-9}$$

Where C_{00} and C_{11} show the cost of correct classification, this cost is set to 0, C_{01} and C_{11} are the cost of incorrect classification. In the case of the fraud detection illustration, $C_{01} > C_{10}$.

2.2.7 Validation schemes

Validation schemes are used in ML modelling where it is implemented before a model is trained with the purpose of protecting against overfitting. Along with underfitting, overfitting causes poor performance in ML and a model should be protected against both [27]. When a model learns the information together with noise in the data it is trained on to the point where it adversely affects the model's performance on new data, this is known as overfitting. A model is said to be underfit when it is unable to both model the training data and generalize to new data, in essence the model does not fit the data suitably. Two popular validation schemes observed in literature are cross validation and holdout validation.

a. Cross Validation

When using cross-validation, the model is trained on several folds (or divisions) that partition the dataset into equally sized subsets. For each validation fold the model is trained using the training-fold data and then the model performance is assessed using data from that validation fold. This method was used by Granholm et al. [27] to validate results obtained from a ML model in shotgun proteomics.

b. Holdout Validation

In holdout validation all data is given chance to be training data and then again testing data. A percentage of the data is used as a validation set and a model is trained on the training set and its performance is evaluated using the validation set [28]. This validation scheme is recommended for large datasets.

2.2.8 Multistage classification

The evaluation of anomaly detection performance is crucial in determining its efficacy. One way to evaluate the effectiveness of anomaly detection is by assessing the Time-to-Detection (TTD) and classification accuracy. The BATADAL competition provides a platform to evaluate algorithms for detecting cyber-physical attacks. The competition design focuses on three goals: (1) to disclose the presence of an ongoing attack in the shortest amount of time possible, (2) to minimize false alarms, and (3) to identify which components of the system have been compromised, if possible [7]. The competition employs multi-stage detection techniques and dimensionality reduction in sensory data to detect both global and local anomalies. Table 2-3 provides the results of the top 4 competitors.

The top BATADAL competitors used multi-stage detection techniques and reduced the dimensionality of sensory data to detect both global and local anomalies. The winning model-based approach utilized EPANET, a water distribution system modelling software package, to simulate the hydraulic process of the water distribution system (WDS), and then compared the simulation results with the available SCADA readings to classify the errors and detect anomalous behaviour [29]. However, this approach was criticized as it might not be effective in real-life situations, due to factors such as the variability of demand patterns, uncertainties in the hydraulic model, and unavailability of a reliable system model [7].

The second-place approach utilized a data-driven strategy that was solely based on the SCADA observations provided. This approach employed statistical fences to detect simple anomalies and utilized supervised machine learning to detect contextual outliers. To achieve this, the approach relied on the capability of artificial neural networks (ANNs) to model complex nonlinear relationships. Additionally, Principal Component Analysis (PCA) was used to address stealthy attacks. This was achieved by analyzing the combined data obtained from all sensors in the multi-dimensional space to detect global anomalies [30].

The team that took third place in the BATADAL competition proposed a two-pronged approach to anomaly detection. The first method involved verifying the integrity of the actuator rules and SCADA data through (1) comparing the SCADA readings to predefined actuator rules and (2) identifying data points for all variables that fall outside of established thresholds using normal operating conditions as a reference. The second method utilized low-dimensionality techniques to distinguish anomalies and then employed PCA to separate normal and anomalous data components [7]. The results reported in Table 4 correspond to the outcomes obtained from the second detection method.

The team who placed fourth in the BATADAL competition employed a method of district metering to address the issue of dimensionality. This approach involves dividing a water distribution system into smaller, isolated portions referred to as District Metered Areas (DMAs). Valves are installed along pipes connecting one DMA to another. For each DMA, Recurrent Neural Networks (RNNs) were developed to predict the water levels of tanks based on pump flow, upstream pressure, and the time of day, using data obtained from normal operating conditions. A statistical control process was then applied to identify any sudden changes in the time series of errors generated by the neural network [31]. This method was outlined in [7].

Table 2-3.Methods used by the top 4 BATADAL competition teams and their results on 7 attack

Position	Stage 1	Stage 2	Stage 3	Number of attacks detected	S score
1	Mixed-integer linear program	EPANET simulation	-	7	0.970
2	Statistical fences	ANN	PCA	7	0.949
3	Unveiling low dimensionality components	PCA	-	7	0.927
4	DMA	RNN	statistical control process	6	0.894

Ramotsoela et al. [32] conducted a comparison between the multi-stage attack detection algorithms implemented in the BATADAL competition and two other approaches. The first approach involved conventional algorithms, including density-based, parametric, and classification algorithms, that were trained on the entire feature space in a centralized manner. The second approach was an ensemble detection scheme, which combined the scalability of density-based techniques, such as Subspace Outlier Degree (SOD) and LOF, with the precision of parametric algorithms. This ensemble technique leveraged the dimensionality reduction capabilities of the density-based algorithms, feeding the reduced data into the parametric algorithm, Quadratic Discriminant Analysis, which is capable of discovering complex decision limits and thus leading to a more precise system.

Aarabi et al. [33] emphasize the significance of feature selection when classifying certain datasets. The feature selection methodology they propose is based on two crucial aspects, relevance and redundancy. To evaluate the correlation between features and response, they employed linear regression-based feature selection methods and a ReliefF algorithm incorporating Ranked Feature Selection (RFS) applied to parameterized EEG data obtained from neonates. After comparing the performance of these methods using an evaluation metric, the most suitable method was determined through backpropagation within a neural network. This approach demonstrated excellent results in detecting ictal activities in newborns.

2.2.9 Ensemble learners

Ensemble learning involves combining multiple base learners, as opposed to relying on a single, one-size-fits-all approach. This is done to take advantage of the diversity of the base learners, which leads to improved performance compared to using a single algorithm. [15]. Despite the existence of multiple ensemble algorithms, determining an optimal ensemble configuration for a particular dataset remains a challenging task. [34]. In the 1980s Hansen and Salamon [35] made the observation that predictions generated by a group of classifiers often exhibit higher accuracy compared to those generated by the best individual classifier. [36].

Schapire [37] demonstrated the effectiveness of ensemble learning, where weak learners can be transformed into strong ones through a process known as "boosting". This is just one of the many popular ensemble techniques, alongside "bagging" and "stacking" [38].

In a study by Amozegar and Khorasani [39], a novel ensemble of dynamic neural network identifiers was proposed for the purpose of Fault Detection and Isolation (FDI) in gas turbine engines. The authors aimed to demonstrate that an ensemble-based FDI approach would exhibit superior performance compared to using individual neural networks. To this end, three standalone neural network learning algorithms were trained individually to represent the dynamics of the engine, including a dynamic Multi-layer Perceptron (MLP), a dynamic radial-basis function (RBF) neural network, and a dynamic support vector machine. Two heterogeneous ensemble models and one homogeneous ensemble model were then created using these three algorithms. The results indicated that the heterogeneous ensemble outperformed the individual models.

The authors of the study noted that while the improved accuracy and performance of the ensemble technique came at the cost of increased computation time, they argue that this trade-off is offset by the fact that the ensemble methods do not require fine-tuning, which is necessary for a single neural network but is a labor-intensive process.

In a study by Aburomman and Reaz [34], a novel ensemble technique was proposed for intrusion detection. This method uses Particle Swarm Optimization (PSO) with Local Unimodal Sampling (LUS) as a meta-optimizer to determine the most effective behavioral parameters for generating the weights that are used to create an ensemble of classifiers. This ensemble approach results in improved accuracy by considering the opinions of both (1) data classification by six different SVM experts and (2) data classification by six different k-NN experts to reach a final decision. An expert refers to a collection of five binary classifiers that together generate a binary vector of responses. This ensemble was compared to a Weighted Majority Algorithm (WMA) approach on the same experts, using subsets of the KDD99 dataset [40]. The results indicated that the novel ensemble technique outperformed the WMA ensemble. Although the contribution of LUS resulted in improved accuracy, it also led to longer run times.

Cabrera et al. [41] investigated the utilization of ensemble methods for intrusion detection in Mobile Ad-Hoc Networks (MANETs). The intrusion detection systems are implemented locally on each node within the MANET, and network operational data is collected and

analyzed through a local anomaly index, which measures the difference between the current node operation and a normal baseline. The researchers employ Cross-feature analysis (CFA) to identify correlations between features, building 28 classifiers based on the C4.5 classifier, which are then deployed in each node. The local anomaly indexes at each time interval are calculated by taking the average of the outputs from these 28 classifiers.

2.3 Data preprocessing

Data preprocessing allows for uncovering knowledge in a dataset prior to the application of a ML algorithm. Applying these set of techniques is a great tool to deal with complex data, especially sensor data. In this section data preprocessing techniques are explored by providing a background explanation on each and then describing where they were found to be used in reviewed literature.

2.3.1 Missing data imputations

Effective learning by many ML algorithms relies on complete datasets and if the algorithm assumes that the dataset is complete where it is not can yield bad results. Data imputation means replacing missing values with estimates to complete the dataset. There are many approaches to missing data imputation such as the mean imputation where the mean of the present values is calculated and the missing values are imputed by this mean, regression imputation where a regression model is fit on the available data of that variable, and an imputation approach based on k-NN where *k*-nearest neighbours are calculated and the missing values are imputed based on the selected neighbours [42].

Quality metrics exist in missing data imputations for the approach used. For single value imputations, the RMSE (Root Mean Square Error) in Equation 2-10 and MAPE (Mean Absolute Percentage Error) in Equation 2-11 [43].

$$\boldsymbol{RMSE} = \sqrt{\frac{\sum_i (\hat{y}_i - y_i)^2}{n}} \qquad 2\text{-}10$$

$$\boldsymbol{MAPE} = \frac{100\%}{n} * \sum_i \left| \frac{y_i - \hat{y}_i}{y_i} \right| \qquad 2\text{-}11$$

Where y_i is a real value, $\hat{y}_i$ is the forecasted value and n is the number of forecasts.

For datasets with long gaps by missing values, similarity in Equation 2-12 and dynamic time warping distance metrics are appropriate [43], [44].

$$\boldsymbol{Similarity} = \frac{\mathbf{1}}{\boldsymbol{n}} * \left(\sum_{\boldsymbol{i}} \frac{\mathbf{1}}{\frac{|y_i - \hat{y}_i|}{\boldsymbol{max}(\hat{y}_i) - \boldsymbol{min}\,(\hat{y}_i)}} \right) s \qquad \textit{2-12}$$

In [45] by Bijlsma et al., a large-scale human metabolomics study was performed on 600 plasma samples to detect variances in metabolic profiles when associated with a large biological variation. To achieve this, they developed a data preprocessing strategy for samples analysed using a liquid chromatography-mass spectrometry (LC-MS) lipidomic method. As part of the preprocessing, they had to deal with missing values which were caused by missing peak values during a peak peaking process, they used a procedure called the 80% rule which resulted in reduced missing values. This procedure however still resulted in 10% of missing data represented as zeros, these were imputed using a using regularized expectation maximization method which is described in [46].

Fan et al. [47] presented a review comprehensive review of data preprocessing methods which includes an application of missing value imputations for operational data in the building sector when data is collected with faults in the data collection process. They describe two types of methods, univariate and multivariate methods, where univariate methods include mean imputation and moving average imputation where in either case the variable's data characteristics solely determine how the missing values will be inferred. Multivariate methods are more advanced methods and result in more accurate imputations and examples include k-NN and regression model-based methods.

2.3.2 Outlier filtering

In a dataset, data points that lie a peculiar/ abnormal distance from other data points are considered outliers, these values lie in the extreme ends. In many cases it takes discretion after studying the data to decide which are true outliers, meaning outliers that add value to the descriptiveness of the data as they represent natural occurrences and thus should be retained, and other outliers which are in contrast to the descriptiveness of the data and should not be retained. These outliers could occur due to errors in measurements, experimental errors or measurement variations [48].

Li et al. [49] hypothesized that classification accuracy can be improved if an algorithm is trained on data where outliers have been removed. They looked at data of Multispectral Imaging (MSI) implemented on burn tissue. They developed an outlier detection and removal method based on Z-test, a statistical hypothesis test used to check that the means of two populations are different when the variances are known, and the sample size is large, they found

that the accuracy was improved from 63% to 76%, a good accuracy which is equivalent to what burn surgeons achieve when using clinical judgement.

Liu et al. [50] proposed a novel outlier detection method, a one class learning approach that works in an unsupervised fashion and automatically removes outliers from a corrupted dataset. The method works by optimizing a kernel-based max-margin objective to learn a big enough margin one-class classifier. Upon experimenting on four image datasets, they achieved good results in removing outliers.

2.3.3 Noise reduction

Noise is meaningless data which corrupts the underlying information in a dataset and since data collected from the real-world using measurement tools is hardly perfect, many times data needs to be processed to remove or reduce the noise components, if not, there is a possibility of reduced system performance such as classification accuracy and increased time and resources required to train the classifier [51].

A popular noise removal/reduction technique is data smoothing based on linear filters such as an averaging filter which applies a moving average over a window of data points, a median filter which applies a moving median over a window of data points and Gaussian filter which applies a Gaussian-weighted moving average over a window of data points. Other examples of smoothing methods are methods based on linear and quadratic regression.

Zhu and Wu [51] performed a study on noise in ML applications where they evaluated its effects on system performance and provided possible solutions to addressing noise. They recognise noise as having two categories: class noise and attribute noise, class noise is label noise and it stems from incorrectly labelling am example, attribute noise is when one or more attributes have corrupted values. To handle class noise, they look at a classification filter and a partitioning filter where the partitioning filter demonstrated a better performance with a higher ability to deal with noise.

Kang et al. [52] applied a k-NN based filter before resampling data as a new under-sampling scheme proposed to aid in class imbalance problems caused by noisy minority example which result in a decreased performance of a classifier. The proposed scheme exhibited an improvement in all four under-sampling methods it was implemented on, namely: Adaboost, RUSBoost, UnderBagging, and EasyEnsemble.

2.3.4 Feature selection

Feature selection has proven to be a good strategy through its effectiveness in reducing overfitting. This method helps in preparing clean and comprehensible data [53]. Feature selection is essentially selecting relevant features from a dataset when developing a model. Applying feature selection before classifying reduces complexity and ultimately the computational time and resources [33].

The review provided by Fan et al. [47] outlines how feature selection techniques can be divided into three categories which are filter, wrapper and embedded methods. With the filter method features are chosen based on how well they perform in statistical tests to see how well they correlate with the end variable [54]. The Pearson's correlation method was used for numerical variables in data collected from commercial buildings, it is a method for calculating the linear dependency of two continuous variables [55], for variables *X* and *Y* Pearson's correlation is defined by the formula:

$$\boldsymbol{\rho(X,Y)} = \frac{\mathbf{1}}{\boldsymbol{N-1}}\sum_{\boldsymbol{I=1}}^{\boldsymbol{N}}\left(\frac{\boldsymbol{X_i-\mu_X}}{\boldsymbol{\sigma_X}}\right)\left(\frac{\boldsymbol{Y_i-\mu_B}}{\boldsymbol{\sigma_B}}\right) \qquad \textit{2-13}$$

Where μ_X and μ_Y are the means of *X* and *Y* respectively and σ_X and σ_Y are their standard deviations.

In wrapper methods a subset of selected features is used to train a model, based on the conclusion drawn from the previous model the search for new features can continue by adding or removing features, or end if the evaluation criterion is satisfied. They evaluated different wrapper methods on commercial buildings data, which include classification and regression trees, chi-square automatic interaction detector (CHAID) and a forward selection method. Other popular wrapper methods found in data science are backward elimination and recursive feature elimination.

Embedded methods leverage the benefits of both the filter and wrapper methods. Jovic and Brkić [56] evaluated a random forest algorithm which reduced data dimensions from 20 to 6, and a C5.0 algorithm which reduced the data dimensions from 28 to 10. Wrapper methods reduced dimensionality the furthest compared to the other methods, however they used up much computational time and known to be prone to over-fitting compared to filter methods [56].

2.3.5 Feature extraction

For a ML model to learn from a signal, the raw data needs to be transformed into a set of features, this process is called feature extraction and is especially useful when dealing with noisy data such as that collected from sensors [57]. Identification of important features for a specific problem, description of those characteristics, and implementation of a method to extract those features are all necessary for manual feature extraction. Deep learning methods have the ability to automatically extract features from signals where the initial layers of deep networks have mostly replaced feature extraction [58]. While deep learning techniques bear this advantage, they also carry their share of disadvantages such as requiring a large amount of data, being computationally expensive, and the difficulty of understanding their inner workings. The below subtopics investigate the types of manually extracted features to consider for training ML algorithms.

a. Timestamp Features

Observations with a timestamp allow for information extraction for ML models when certain events are associated to dates and times. In a water treatment system, a certain pump could be off on a certain day of the week leading to low tank level reading, or perhaps certain equipment is down for routine maintenance on a certain day of the week at a certain time. Date-time data of "2022–05–10 10:21:05", the following features can be extracted from it:

1. Hour of the day: 10
2. Weekend: No
3. Public holiday: No
4. Season: Winter
5. Day of the week: Tuesday

Rubin et al. [59] collected data of neurons in the hippocampal CA1 of mice using time lapse imaging and used time-stamped experienced events to study how time relates to different occurrences in episodic memory. They found that regardless of the spatial location in which they occurred, temporally near events had a same timestamp and thus timestamp features can be extracted to aid in the analysis.

b. Statistical features

When calculating statistical features a sliding window of a certain width, which the researcher chooses through iterations, is used on the time series data and calculating statistics for each iteration [43]. Some of the common statistical features are:

- Mean: The average of the data, found by adding the data points in the window and dividing by the number of values present. The equation for a sample mean is given by:

$$\bar{\boldsymbol{x}} = \frac{\sum \boldsymbol{x}}{\boldsymbol{n}} \quad \textit{2-14}$$

Where $\sum x$ is the sum of the values in the sample, and n is the number of values in the sample.

- Variance: This is a measurement of the spread between numbers in the dataset. It is calculated by taking the average of squared deviations from the mean. The population and sample variance equations are:

$$\boldsymbol{\sigma^2} = \frac{\sum(\boldsymbol{X}-\boldsymbol{\mu})^2}{\boldsymbol{N}} \quad \textit{2-15}$$

$$\boldsymbol{s^2} = \frac{\sum(\boldsymbol{X}-\bar{\boldsymbol{x}})^2}{\boldsymbol{n}-\boldsymbol{1}} \quad \textit{2-16}$$

Where σ^2 is the population variance, X represents each value, μ is the population mean, N is the number of values in the population, s^2 is the sample variance, $\bar{x}$ is the sample mean and n is the number of values in the sample.

- Median: This is the middle value in a dataset, ordered from smallest to largest values. The formula for calculating the median is:

$$\boldsymbol{Median\ when\ N\ is\ even} = \frac{(\boldsymbol{N}+\boldsymbol{1})^{\boldsymbol{th\ term}}}{\boldsymbol{2}} \quad \textit{2-17}$$

$$\boldsymbol{Median\ when\ N\ is\ odd} = \frac{\frac{\boldsymbol{N}^{\boldsymbol{th\ term}}}{\boldsymbol{2}}+\left(\frac{\boldsymbol{N}}{\boldsymbol{2}}+\boldsymbol{1}\right)}{\boldsymbol{2}} \quad \textit{2-18}$$

Where N is the number of values in the dataset.

- Mode: this is the value that appears the most in the dataset. This value is found by counting the number of each occurrence.

Veltink et al.[60] used the mean to identify a patients posture and activity and differentiate either activity as dynamic or static in an effort to improve rehabilitation treatment of patients. Si et al. [61] used the mean of Y-axis acceleration to determine if a user was standing or sitting and the variance to determine if a user was moving or not.

c. Temporal and Spectral features

When dealing with signals there are two types of features that can be extracted, temporal features which are time domain features and spectral features are frequency-based features.

Temporal features pose a direct visible interpretation, such as energy of signal, maximum amplitude, minimum energy, etc.

Spectral features are obtained by using Fourier Transform and wavelet transform methods to convert a time-based signal into the frequency domain. Frequency domain features such as the fundamental frequency of a signal and its frequency components, spectral flux, spectral density, etc.

To derive the Fourier Transform of a function, the Fourier Series synthesis equation is established [62], [63],

$$\boldsymbol{x(t) = \sum_{n=-\infty}^{+\infty} c_n e^{jn\omega_o t}} \qquad \textit{2-19}$$

Where c_n is given by the Fourier Series analysis equation,

$$\boldsymbol{c_n = \frac{1}{T}\int_T x(t)\, e^{-jn\omega_o t} dt} \qquad \textit{2-20}$$

The fundamental frequency, $\omega_0=2\pi/T$, becomes very small as $T\rightarrow\infty$, and $n\omega_0$ becomes a continuous quantity that can take on any value therefore a new variable $\omega=n\omega_0$ is defined. Let $X(\omega)=Tc_n$ and after making substitutions the Forward Fourier Transform results:

$$\boldsymbol{X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt} \qquad \textit{2-21}$$

Si et al. [61], discussed previously, were able to compute an FFT to reveal a power spectrum of the Y-axis acceleration and used its max to determine whether a user was running or not. The results they obtained for posture inference show a clear distinction between sitting, standing, walking, and running.

d. Feature scaling

Many ML algorithms are sensitive to the scale of data as they may have a wide range of values. As an example, if one feature is the height of a hippopotamus in centimetre s(cm) and the other feature is its weight in grams (g), the weight would have a much bigger range than the height, and because some algorithms use the Euclidean distance to calculate the distance between points, the results will not be very accurate. For this reason, data is scaled/normalized in the data preprocessing stage of ML modelling. Two of the most popular methods are max-min normalisation where for a variable the minimum value is changed to a 0 and the maximum value is to a 1, the remaining values are scaled to decimals between 0 and 1. The other method is z-score standardization where a variable is transformed so it becomes normally distributed with a mean of 0 and a standard deviation of 1 [64].

Ashouri et al. [65] implemented max-min normalisation on numerical energy consumption data of a building to obtain values between 0 and 1, they dealt with categorical values by converting them to 0,1. Nawi et al. [66] investigated improving the efficacy of MLP, an ANN based algorithm. They evaluated three normalization techniques, min-max, z-score and decimal scaling, aiming to improve the convergence of the ANN algorithm. Four gradient descent algorithms were tested on three popular datasets: Haberman dataset, wine recognition dataset and iris dataset. The results of the classification results on the iris dataset are shown in Table 2-4 below, where GD refers to gradient Decent.

Table 2-4. Classification results for IRIS dataset obtained by Ashouri et al. [65].

Normalization Method	**Accuracy**			
	GD with momentum	GD with line search	GD with gain	GD with gain and line search
Min-Max	98.41%	99.87%	99.12%	98.01%
Z-Score	96.06%	99.46%	98.73%	98.76%
Decimal Scaling	97.20%	94.47%	96.12%	95.89%

2.3.6 Dimensionality reduction

Dimensionality in ML refers to the number of input variables or features in a dataset and dimensionality reduction is then the techniques that are used to reduce them. Dimensionality reduction methods make it possible to comprehend and see the structure of large data collections [67]. Popular methods are given below.

Principal component analysis

Principal Component Analysis (PCA) is a well-known multivariate statistical technique that is widely used for reducing the dimensionality of a data set. It operates by identifying the eigenvectors that capture the interdependencies among a set of inter-correlated features, thereby explaining a significant portion of the variation in the data.

Lam et al. [68] conducted a study on the relationship between electricity usage in office buildings and weather patterns. They utilized Principal Component Analysis (PCA) to analyse five significant climatic factors, including dry-bulb temperature, wet-bulb temperature, global solar radiation, clearness index, and wind speed. With the help of PCA, they were able to identify the interdependency among the features, thereby explaining the majority of the data variation. Subsequently, they used regression models to relate the derived components to electricity use, thus demonstrating the efficacy of PCA as a dimensionality reduction technique.

In [69], a review and comparison of different dimensionality reduction techniques was carried out. The results of the study showed that nonlinear techniques, although popular, are often unable to outperform traditional linear techniques such as PCA.

2.4 Comparison and discussion

This literature review looked at many intrusion detection techniques. The data processing stage of ML modelling is of great importance and preprocessing techniques found in literature were evaluated. The standard preprocessing workflow includes corrected missing data through imputations, filtering outliers, reducing noise, selecting features, and extracting features.

Training a ML algorithm on a complete dataset yields good results and reduces bias of the model and for this reason missing data imputation must be applied where data is missing. There are simple and complex approached to performing missing data imputations found in literature and the discretion on the choice of approach relies on the extent of gaps in the data. When the dataset is complete, a common problem to deal with next is identifying data points that that stand out as being notably different from the rest of the data, this is outlier filtering and Li et al. [49] showed that if outliers are addressed prediction accuracy can improve significantly.

Noise reduction is a preprocessing step that will almost always be performed in applications of critical water systems infrastructure intrusion detection due to the data being collected from the real-world using measurement tools which are hardly perfect. While common techniques include data smoothing based on linear filters such as an averaging filter or a Gaussian filter, there are also noteworthy methods such as a classification filter and a partitioning filter used by Zhu and Wu [51].

At times a variable in a dataset will be redundant to another and this variable is irrelevant as a feature since it carries the same information as the other. For example, if water pump is turned on based on the reading of a water level sensor, the pump's data will be redundant to the sensor's data therefore it can be removed as a feature. Feature selection must be applied as it improves accuracy and reduces the training time.

Feature extraction plays a major role in providing data that ML algorithms can easily interpret and learn from and yield better results than if the model were trained on raw data. Although ML algorithms require manually extracting features such as timestamp, statistical and spectral features, and deep learning methods like neural networks automatically extracts features, ML

algorithms are still a popular choice in literature for intrusion detection applications owing to their simple structure, being less data demanding and less computationally expensive.

A comprehensive comparison of various intrusion detection techniques was presented in Table 5 by Ramotsoela et al. [1]. The results indicated that standalone algorithms, such as the Quick Discriminant Analysis (QDA) used in BATADAL, can perform well and even rank second among the evaluated techniques. However, multistage and ensemble methods can optimize the performance of individual algorithms by combining their strengths, leading to improved accuracy, reduced computation time and resources. While ensemble methods have the potential to provide superior performance, their implementation is not always straightforward. For instance, Aburomman and Reaz [34] demonstrated the challenges in creating effective ensemble models through their study, where the Local Unimodal Sampling (LUS) method was not found to significantly improve the performance of their intrusion detection system."

The techniques were tested in various environments, including water systems, medical systems, and wireless communication networks. In many cases, dimensionality reduction techniques were applied to achieve better anomaly detection results. Preprocessing techniques play a crucial role in the success of these models. Several competitors in the BATADAL competition utilized PCA for dimensionality reduction, and these methods performed well. The popularity of PCA can be attributed to its ability to remove correlated features, reduce overfitting, and improve algorithm performance.

In the field of data-driven approaches, the majority of authors achieved a fair level of success. Housh and Ohar [29] employed a model-based approach and emerged as the most successful participants in the BATADAL competition. However, the limitations detailed in the accompanying table render this approach negligible. While some researchers managed to strike a balance between accuracy and detection time, others displayed a stark contrast between their performance in these two metrics. A few methods were found to be computationally intensive, significantly prolonging the time required to produce results.

The methods reviewed in this section are compared in Table 2-5.

Table 2-6. Comparison of reviewed methods

Reference	Objective	Approach	Strengths	Weaknesses	Technique	Classifiers
Housh and Ohar [29]	Detection of complex cyberattacks on SCADA level of WDS	Hydraulic modelling of WDS combined with multilevel classification algorithm for anomaly detection	Detecting all attacks and in a timely manner with an S-score of 0.9701	Approach assumes unrealistically perfect real-life conditions.	Multistage	Novel
Abokifa et al. [30]	Recognising anomalies in monitoring and control data from WDS SCADA	ANNs trained on historical data to recognise different types of anomalies and PCA applied to uncover global anomalies	• Detecting all attacks and in a timely manner with S-score of 0.9491 • Detecting stealthy attacks	• Algorithms has feeble performance when data is noisy. • False alarms frequently raised after real attack no longer exists	Multistage	PCA, ANN
Giacomoni et al. [7]	Identification of cyber-attacks on water distribution systems by unveiling low dimensionality in the sensory data	Verification of SCADA readings then applying PCA to unveil anomalies	Achieved high TNR rate of 0.997, meaning it has good ability to detect false alarms.	S-score was lowered due to its low TPR rate of 0.838, resulting in low sensitivity of the model.	Multistage	PCA
Brentan et al. [7]	On-line cyber-attack detection in water networks	Dimensionality reduction using DMA and a RNN is created from each DMA. A statistical process is used to identify rapid shifts in the neural network error time series	Outperforms standalone classifiers like SOD, OSVM, LOF and LDA	• Computationally expensive and suffers from a low time to detection, STTD is 0.857 • Did not detect all attacks	Multistage	RNN
Ramotsoela et al. [1]	Attack detection in water distribution systems using machine learning	Ensemble that leverages strengths of density based and parametric algorithms. The method combines SOD and LOF using QDA	Achieved S-score of 0.9142 and outperformed standalone LOF and SOD which achieved 0.8773 and 0.8617 respectively	• Computationally expensive • Outperformed by standalone QDA which had an S-score of 0.9495	Ensemble	SOD, LOF, QDA
Aarabi et al. [33]	Seizure detection in newborn babies	Two linear correlation-based feature selection methods applied to EEG data and the optimized feature subsets are classified using a backpropagation neural network	High performance in detecting seizures with a seizure detection rate of 93%	• Low false seizure detection rate due to • Highly dependent on the selection of uncorrelated features 1.17/h opposed to >5.38/h achieved by other researchers	Multistage	ANN
Amozegar and Khorasani [39]	New approach for FDI of gas turbine engines using an ensemble of neural network identifiers	Three separate dynamic neural network architectures used to monitor health of gas turbine engine and then developing heterogeneous and homogeneous ensemble models.	• Ensembles perform better than their standalone models • Ensembles reduce effort of fine tunings required by their standalone models	Computationally expensive	Ensemble	Dynamic Neural Networks
Aburomman and Reaz [34]	Intrusion detection in KDD99 using a novel SVM-kNN-PSO ensemble method	PSO with LUS used as meta-optimizer to generate weights used to create an ensemble of SVM and k-NN classifiers	PSO alone provides added accuracy and reduced running time	• Using LUS reduces the overall accuracy of PSO and has longer runtimes. • PSO does not always guarantee optimal weights are found	Ensemble	SVM,kNN
Cabrera et al. [41]	Utilising ensemble methods to detect intrusions in MANETs	CFA used for feature selection	Classifier combination by averaging a large number of classifiers improves intrusion detection rate	Distributed AD focused on MANETs; implementation is on local nodes	Ensemble	C4.5

Lam et al. [68]	Investigate the relationship between electricity use in office building and weather patterns	PCA used to reveal components then used in regression models	PCA reduces dimensionality resulting in less computational effort and improved results.	No classification	Multistage	N/A

2.5 Conclusion

The field of behaviour-based intrusion detection in cyber-physical security has seen numerous techniques developed to address the challenges posed by these complex systems. The BATADAL competition provides a valuable platform for showcasing some of the best and most innovative techniques for anomaly detection (AD) in critical water systems infrastructure. Beyond this, literature review demonstrates the adaptability of these techniques to other application domains.

While model-based approaches have been shown to be effective in producing comparable results to data-driven approaches, machine learning (ML) algorithms have become increasingly appealing due to their adaptability to real-world applications. Preprocessing is an essential aspect of ML, as it has the potential to speed up computation, improve model accuracy, and preserve discriminatory information within the dataset. In the literature reviewed, multistage methods that incorporate a significant amount of preprocessing have demonstrated strong performance.

Sensor and actuator data from industrial control systems (ICS) such as SCADA is often complex, noisy, and non-linear, making advanced preprocessing techniques necessary for effective intrusion detection. Techniques such as principal component analysis (PCA) in feature transformation and k-NN for noise filtering have been applied in these cases with promising results.

Chapter 3. Data

SWaT is a water treatment testbed for cybersecurity research, it consists of a scaled down six-stage water treatment process that is almost indistinguishable to a real-world treatment plant, the SWaT testbed can be seen in Figure 3-1. SWaT became operational in 2015 and the SWaT dataset used in this study was collected from 7 days of normal operation and 4 days of operation with attack scenarios. During the data collection all network traffic, sensor and actuator data were collected. The work presented in this study looks only at sensor and actuator data, the sensors and actuators used in SWaT are described in Table 3-1. Stage 1 of the physical process begins by taking in raw water, followed by chemical dosing (Stage 2), filtering it through an Ultrafiltration (UF) system (Stage 3), dechlorination using UV lamps (Stage 4), and then feeding it to a Reverse Osmosis (RO) system (Stage 5). A backwash process (Stage 6) cleans the membranes in UF using the RO permeate. The six sub-processes, referred to as stages P1 to P6 are controlled by a set of dual Allen-Bradley PLCs, a primary and a redundant hot-standby. The operation status of the PLCs is monitored by the SCADA system [70]. These sub-processes are shown in Figures 3-2 and 3-3. Data was collected from an empty state and it took 5 hours for SWaT to stabilise while taking an extra hour to completely fill up the tanks in stages 3 and 4, this is to be accounted for in the modelling process of the ML algorithm.

Figure 3-1.Picture of SWaT testbed [70]

Table 3-1.Description of sensors and actuators used in SWaT [8]

No	Name	Type	Description
1	FIT-101	Sensor	A flow meter which measures water inflow into raw water tank.
2	LIT-101	Sensor	A level Transmitter which measures the water level in the raw water tank.
3	MV-101	Actuator	A motorized valve which regulates the raw water tank's water flow.
4	P-101	Actuator	A pump for water from the raw water tank to the chemical dosing stage.
5	P-102 (backup)	Actuator	A pump for water from the raw water tank to the chemical dosing stage.
6	AIT-201	Sensor	A conductivity analyser that measures NaCl level in the water.
7	AIT-202	Sensor	A pH analyser that measures HCl level in the water.
8	AIT-203	Sensor	An Oxidation Reduction Potential (ORP) analyser that measures NaOCl level in the water.
9	FIT-201	Sensor	A flow transmitter used to control the pumps used for dispensation of chemicals.
10	MV-201	Actuator	Motorized valve; Controls water flow to the UF feed water tank.
11	P-201	Actuator	A dosing pump used to dispense NaCl.
12	P-202 (backup)	Actuator	A dosing pump used to dispense NaCl.
13	P-203	Actuator	A dosing pump used to dispense HCl.
14	P-204 (backup)	Actuator	A dosing pump used to dispense HCl.
15	P-205	Actuator	A dosing pump used to dispense NaOCl.
16	P-206 (backup)	Actuator	A dosing pump used to dispense NaOCl.
17	DPIT-301	Sensor	A transmitter that indicates differential pressure. It is used to controls the back-wash process.
18	FIT-301	Sensor	A flow meter that measures the flow of water in the UF stage.
19	LIT-301	Sensor	A level Transmitter for the UF feed water tank.
20	MV-301	Actuator	A motorized valve used in the UF-Backwash process.
21	MV-302	Actuator	A motorized valve that allows water flow from the UF process to the dechlorination unit.
22	MV-303	Actuator	A motorized valve that allows for UF-Backwash drain.
23	MV-304	Actuator	A motorized valve that that allows for UF drain.
24	P-301 (backup)	Actuator	A pump which supplies water from the UF feed water tank to RO feed water tank through UF filtration.
25	P-302	Actuator	A pump for water from the UF feed water tank to the RO feed water tank via UF filtration.
26	AIT-401	Sensor	A meter that measures RO hardness of water.
27	AIT-402	Sensor	An ORP meter that controls the dispensing of NaHSO3 using pump P203 and dispensing of NaOCl dosing using pump P205.
28	FIT-401	Sensor	A flow transmitter used to regulate the UV dechlorinator.
29	LIT-401	Actuator	A level transmitter for RO feed water tank level.
30	P-401 (backup)	Actuator	A pump for water in the RO feed tank to the UV dechlorinator.
31	P-402	Actuator	A pump for water from the RO feed tank to the UV dechlorinator.
32	P-403	Actuator	Sodium bi-sulphate pump.
33	P-404 (backup)	Actuator	Sodium bi-sulphate pump.
34	UV-401	Actuator	A dechlorinator that gets rid of chlorine in the water.
35	AIT-501	Sensor	A RO pH analyser used to measure HCl level.
36	AIT-502	Sensor	A RO feed ORP analyser used to measure NaOCl level.
37	AIT-503	Sensor	RO feed conductivity analyser used to measure NaCl level.
38	AIT-504	Sensor	RO permeate conductivity analyser used to measure NaCl level.
39	FIT-501	Sensor	A flow meter for the RO membrane water inlet.
40	FIT-502	Sensor	A RO Permeate flow meter.
41	FIT-503	Sensor	A RO Reject flow meter.
42	FIT-504	Sensor	A RO re-circulation flow meter.
43	P-501	Actuator	A Pump for dechlorinated water to RO stage.
44	P-502 (backup)	Actuator	A Pump for dechlorinated water to RO stage.
45	PIT-501	Sensor	A RO feed pressure meter.
46	PIT-502	Sensor	A RO permeate pressure meter.

47	PIT-503	Sensor	A RO reject pressure meter.
48	FIT-601	Sensor	A UF Backwash flow meter.
49	P-601	Actuator	A pump for water from the RO permeate tank to raw water tank. (Data is not available as it was not used in the data collection process).
50	P-602	Actuator	A pump for water from the UF back wash tank to the UF filter. This is for the purpose of cleaning the membrane.
51	P-603	Actuator	SWaT has not yet incorporated this pump.

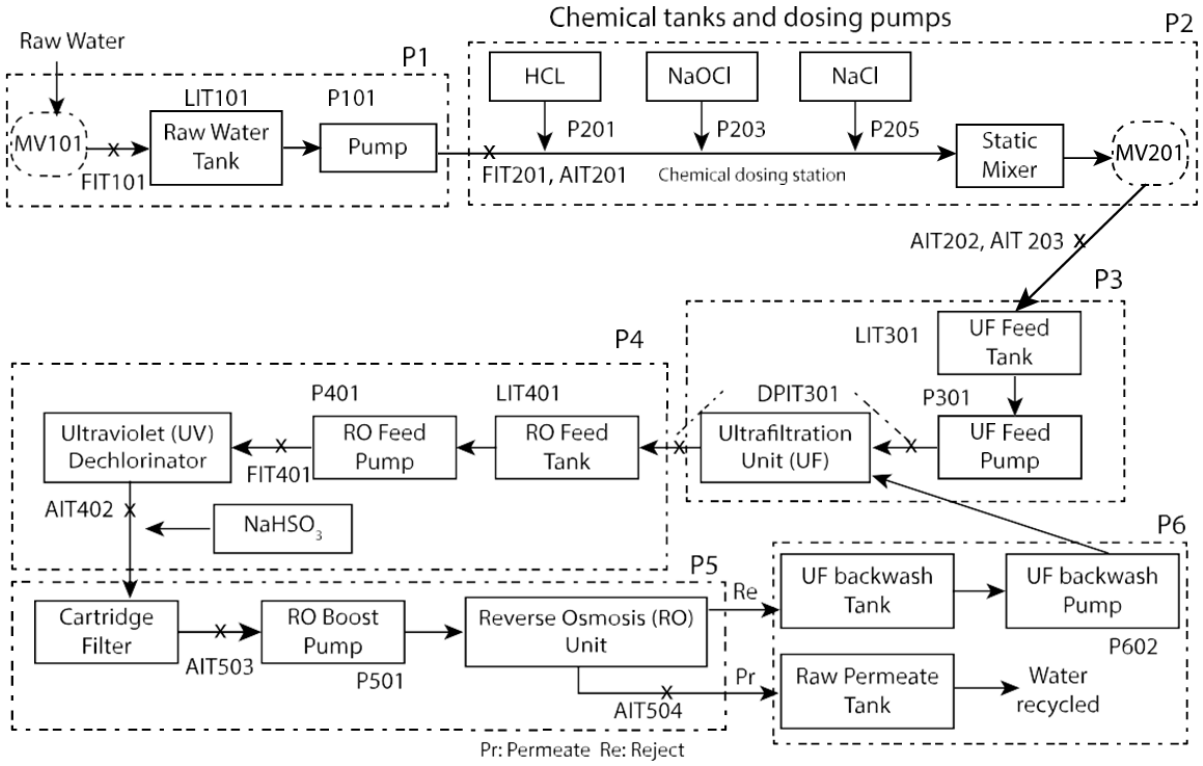

Figure 3-2.SWaT's six stage processes [71]

Where:

- P1: Raw water supply & storage
- P2: Chemical dosing
- P3: UF
- P4: Dechlorination
- P5: RO
- P6: RO permeate transfer, UF backwash
- AITx0y: Analyser Indicator Transmitter
- DPITTx0y: Differential Pressure Indicator Transmitter
- FITx0y: Flow Indicator Transmitter
- LITx0y: Level Indicator Transmitter
- MVx0y: Motorised Valve

- Px0y: Pump
- x = component #; y = process module#

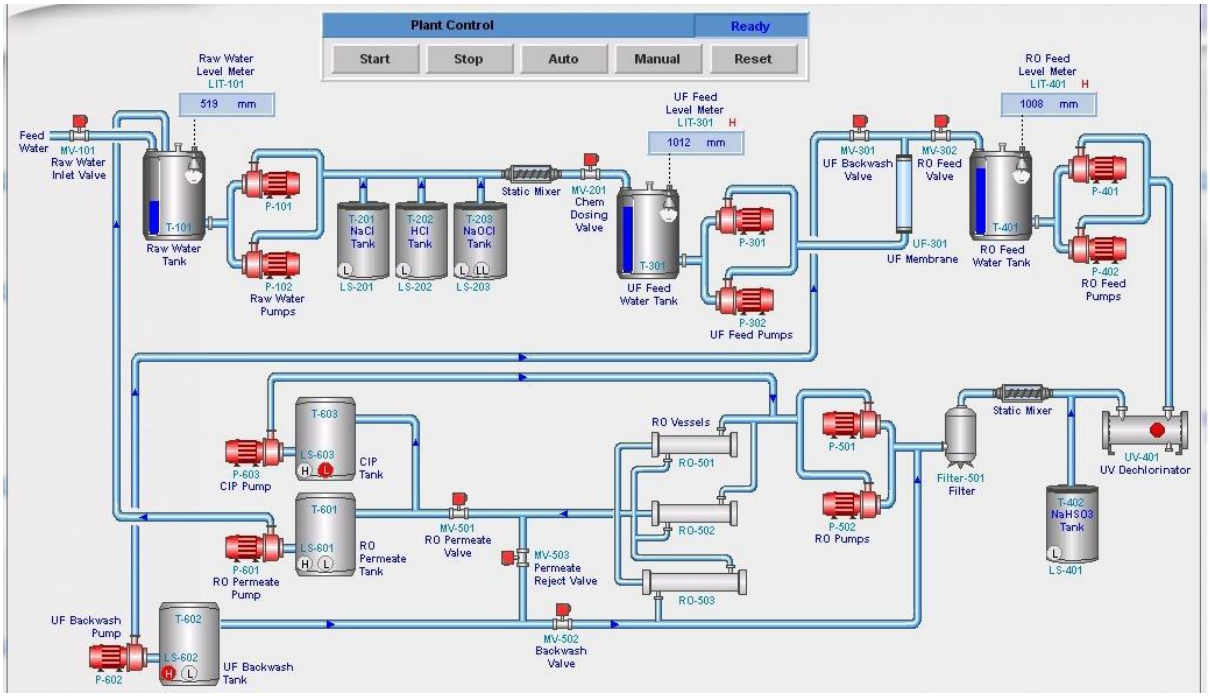

Figure 3-3. SWaT's HMI/SCADA interface [71]

Attack scenarios

The system was attacked using an approach that considers the physical aspects of a CPS, as opposed to existing attack models outlined in network security which do not consider the physical aspects of a CPS [72]. The models used are attacker and attack models described in more detail in [73].

The attacker model is an abstract model that captures the domains of possible attacker intents. The components of the abstract domain model are component, property, and performance. The *Component* dimension includes the physical, cyber and logical elements of the CPS. The *Property* dimension includes physical properties of the products being produced or controlled by the CPS. The *Performance* dimension includes performance characteristics of a CPS.

An attack model may be used to create procedures and functions that are particular to a CPS and in this case the SWaT testbed [8]. An attack model for a given CPS is a sextuple (M,G,D,P, S0, Se), where M is potentially infinite set of procedures that use a knowledge of attack vectors and system properties to launch attacks, G is a set of attacker intents (or goals), D is a finite or an infinite set of domains obtained from the concrete attacker model for C, P is a finite set of

attack points, e.g., a wireless link, and S0 and Se are possibly infinite sets of states of C, that denote, respectively, the start and end states of interest [73].

The resultant attacks in SWaT, based on attack points in each of the six stages, are divided into four types, all of which are expressed in detailed in the SWaT website [70].

A total of 36 attacks in total were introduced into the SWaT system throughout the duration of the data collection process. The durations of the attacks and the magnitude of the effects on the system dynamics vary. Some of the attack's effects are not instant. The attack types are described as:

1. Single Stage Single Point (SSSP): This attack type focuses on exactly one point in a CPS. There are 26 of these attacks in the dataset.

2. Single Stage Multi Point (SSMP): This attack type focuses on two or more attack points in a CPS but on only one stage. In this case set, *P* consists of more than one element in a CPS selected from any one stage. There are 4 of these attacks in the dataset.

3. Multi Stage Single Point (MSSP): This attack type is similar to an SSMP attack, but it targets multiple stages in the CPS. There are 2 instances of this attack in the dataset

4. Multi Stage Multi Point (MSMP): This attack type is an SSMP attack implemented at two or more stages of the CPS. There are 4 of these attacks in the dataset.

All the attacks over the total duration can be visualised in Figure 3-4 below.

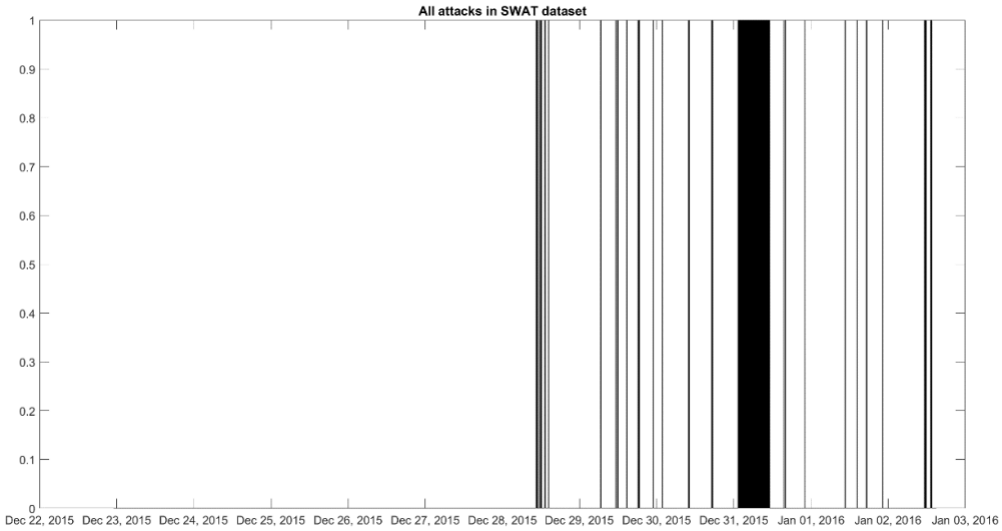

Figure 3-4. Attacks in SWaT dataset

Chapter 4. Methodology

The literature review in chapter 2 documented explored literature relevant to machine learning modelling when dealing with data collected from critical water systems infrastructure. In the data section, chapter 3, an overview of the dataset selected for use in this study was provided. This section provides an outline on the methodology used in the ML modelling approach using sensor and actuator data to detect attacks in the data. Although the dataset is from a water treatment system, the approach discussed here can be adopted for all critical water systems infrastructure. As this study focuses on data preprocessing, there is a heavy emphasis on that part of ML modelling, however it is appropriate to discuss the entire modelling workflow.

4.1 Modelling approach

When designing a machine learning model, a model workflow is followed, this creates a clear structure of the modelling process and provides preliminary insights before modelling. The Model workflow is shown in Figure 4-1 below. Since the work conducted in this study to a great extent focuses on data preprocessing and not the entire ML modelling process, part 4 is the focus of the study and is discussed in great detail and in support of this, parts 1 to 3 are also discussed.

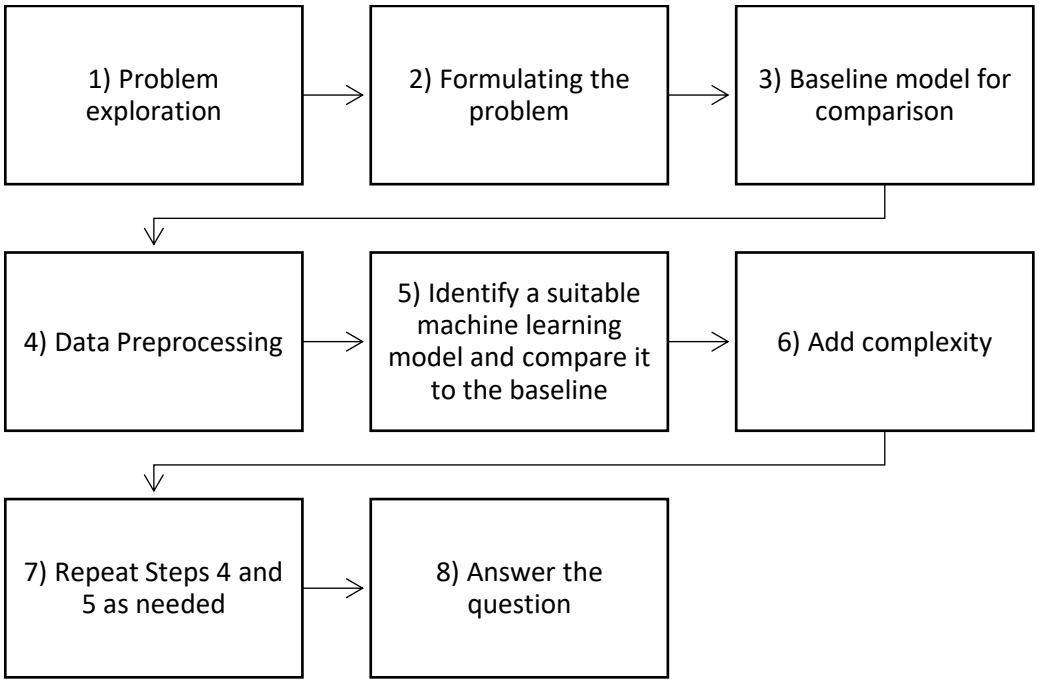

Figure 4-1. ML modelling workflow

4.1.1 Problem Exploration

Problem exploration entails using numerical and visual exploration to better understand the problem and how to solve it. When doing exploration some of the things that can be inspected are, but not limited to, data types, outliers, distributions, variance imbalances, histograms, and plots. This step is done before formulating the problem to provide enough context to make the decision.

Specifically for this study, first the attacks in the sensor and actuator data are visualised. Visualisation of the attacks allows the 4 types of attacks to be viewed against the data and prepare the analyst to apply discretion on how to preprocess the variables, in this way visual analysis adds context on the types of attacks present and how they affect the data. As an example, an attack on LIT101 data between times 17:04:56 and 17:29:00 on the 30th of December 2015 can be seen in Figure 4-2 below and is observed to alter the trajectory of the data points which are expected to rise in value but are set at 800mm, and so the model can be trained on the type of data to expect at that point in time.

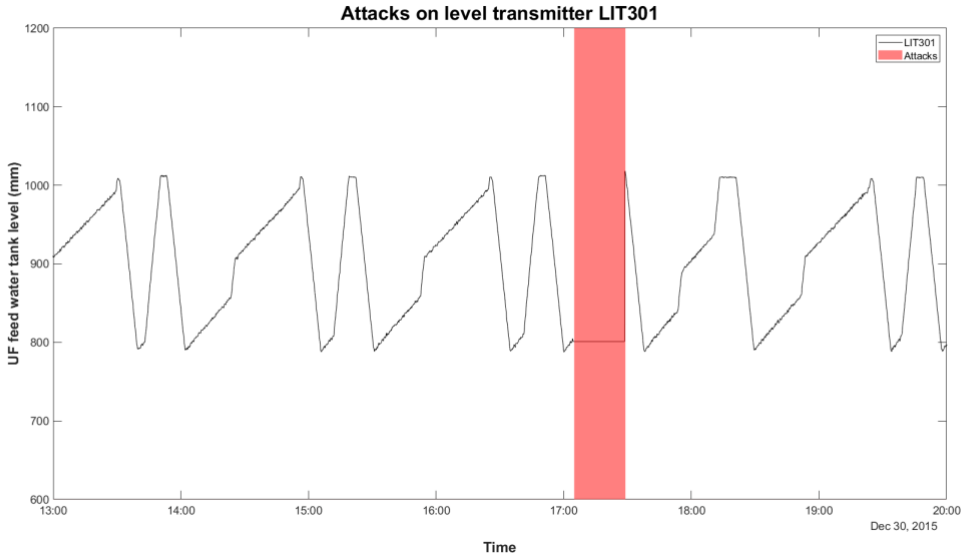

Figure 4-2. Attack observed on LIT301 sensor data

A finding made when exploring the data is that the pumps go in pairs thus having a main and a backup pump where one comes on when the other is off and therefore the pair can be merged into one variable resulting in a smaller feature set for training the model.

Some variables in the dataset carry the same information as other variables and thus are redundant, this means that they can be removed from the total set of features. For this exercise a correlation analysis can be applied where Pearson correlation coefficients are generated to

identify closely related variables. Histograms and scatter plots of the variables being compared are a good way to visually attest to the relationship between the variables.

4.1.2 Problem formulation

This section aims to answer the question: What are you trying to achieve? (1) Is it a regression, clustering, or classification problem? (2) Are you seeking any specific insights? (3) is there a pattern expected? (4) Is there a specific format for the response. Using information gathered from the problem exploration stage this question can be answered.

4.1.3 Baseline models

A baseline model is a simple model that will act as a reference point for evaluating the improvement of subsequent machine learning models. Two popular models have been chosen for classification problems: Fine Decision Trees and Boosted Trees Ensemble.

1. Model 1: Fine Decision Tree

Decision Trees are popular in classification problems due to their simplicity and interpretability [74]. They work by recursively splitting the data into smaller subsets, with each split corresponding to a decision rule that best separates the target class [75].

2. Model 2: Boosted Trees Ensemble

This model creates an ensemble of medium decision trees using the AdaBoost algorithm [76]. Ensemble learners, which combine multiple base models, have been shown to perform relatively well as compared to a single algorithm. This is because the combination of multiple base models reduces the variance of the overall prediction, leading to better generalisation performance [77].

Having a second baseline model is done to verify the experiment results found in chapter 6. The baseline models have no added complexity in the modelling process and serve as a benchmark for comparison with the same models trained on preprocessed data. The modelling approach for training and evaluating ML models is depicted in Figure 4-3 and describes how the data is first spit into training and testing data, followed by the application of the model and then their evaluation.

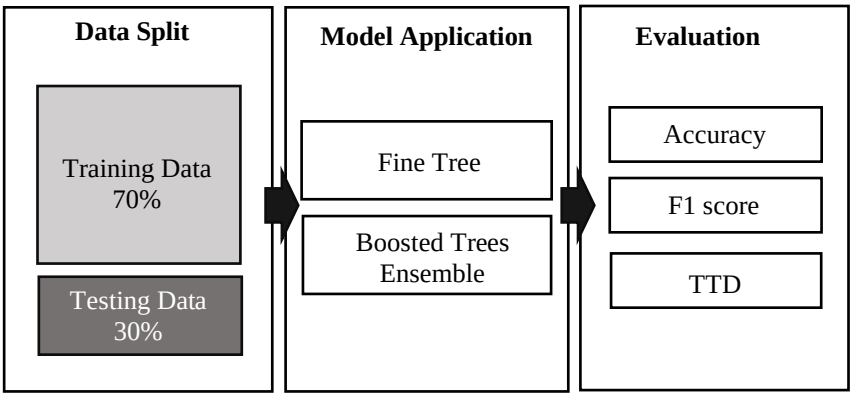

Figure 4-3. Modelling approach for training and evaluating ML models

a. Data Split

The dataset is split into two parts namely the training dataset which is used to fit the model and the testing data is used to measure the performance of the model [78]. The data is split by a 7:3 ratio for the training and testing data respectively, where the training data is the section from the beginning of the dataset up its 70% mark and the testing data is the remaining 30%.

b. Model application

The three selected models are applied in this stage. The models are trained solely with the training data. Typically, hyperparameters would be used when training a model to improve its learning ability however they are not used in this section as these are baseline models, neither are they used in later chapters as the aim of this study is to evaluate preprocessing techniques and using hyperparameters will produce uninterpretable results. A cross-validation scheme is used to protect the model against overfitting, similar to the application in [27], however for this application only 2 folds are used on the training data.

c. Evaluation

The model's performance in evaluated in this stage, where the test data is applied on the trained models to determine how well it performs on new data. The evaluation measures used are chosen for their relevance to intrusion detection in SCADA systems, these measures are:

- Accuracy - This evaluation depicts the percentage of correctly predicted values for the test data. A confusion matrix provides the false-negative, false-positive, true-negative, and true-positive values. The accuracy is then determined by the following equation:

$$\boldsymbol{accuracy} = \frac{\boldsymbol{correct\ predictions}}{\boldsymbol{all\ predictions}} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FN+FP}} \qquad 4\text{-}1$$

- F1 score - Accuracy alone is not enough to evaluate the cogency of a model since its disadvantage is that in an imbalanced dataset it does not accurately represent the true

performance of the model, to overcome this shortcoming it will be used alongside the F1 score which is a weighted average of Precision and Recall. This matric has a range of 0 to 1, with 1 indicating a perfectly precise classifier and 0 indicating the worst. The F1 score is given by Equation 2-7 in chapter 2.

  - Precision - This metric indicates the proportion of accurately predicted positive observations to all expected positive observations, specific to this study it indicates that of the total number of predictions made out to be attacks, the ones that were indeed attacks. Precision of 1 is the best possible outcome and 0 is the lowest.
  - Recall - Also referred to as sensitivity or true positive rate, it gives the proportion of correct predictions in the positive predictions class, in this study it evaluates the model on its ability to identify threats, similar to precision, recall too has a range of 0 to 1.

- Time to Detection – In intrusion detection the TTD is a very important measure of the effectiveness of a trained model because the longer the attack goes undetected the greater the harm that is caused on the system thus increasing the attacker's chances of success. It is the difference between the times at which the attack is detected and when it starts. TTD has a range of 0, the best outcome which means the attack was detected immediately, to infinity, the worst outcome which means the attack was not detected. The TTD for the baseline Tree classifier is shown in Figure 4-4 to be 2447 seconds.

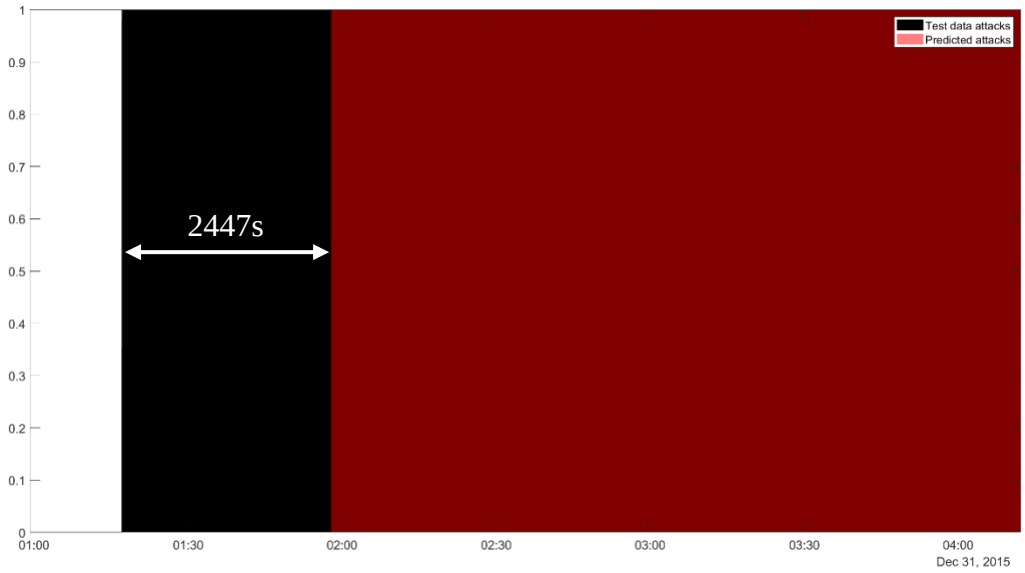

Figure 4-4. Classification Tree TTD

4.1.4 Data Preprocessing

A systemic approach is used to preprocess the remaining variables after the correlation reduction process, where each variable is analysed and the appropriate technique is applied. Many of the variables exhibit unique characteristics and thus can be evaluated and analysed individually. Data preprocessing steps to be applied are given in Figure 4-5 and for each step common techniques are listed in bullet form. These steps are not necessarily applied to each variable as each variable has its own unique characteristics and discretion is required.

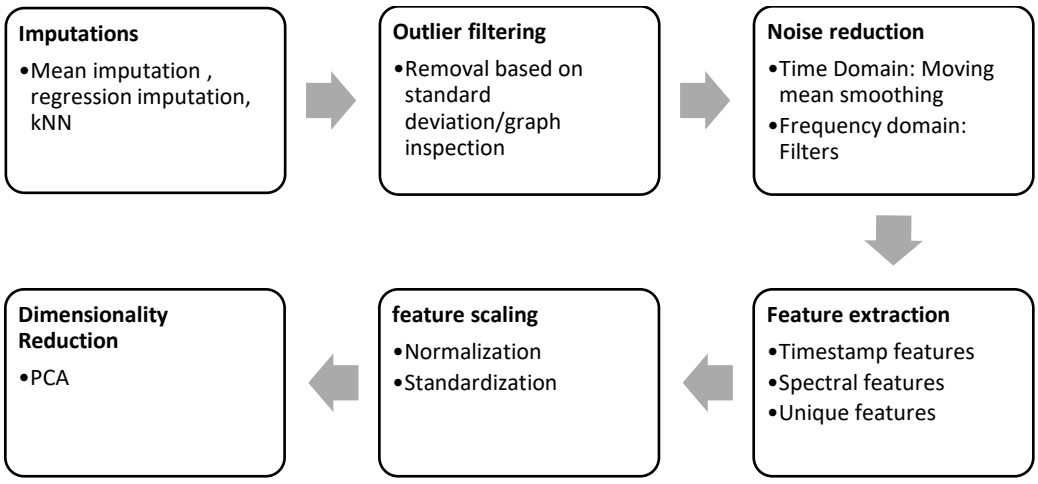

Figure 4-5. Data processing steps

a. Imputations

Missing values can be found programmatically using different platforms, in MATLAB the function *ismissing(A)* returns a logical array that indices the positions in the input data *A* that contain missing values. Missing values can be in the form of not-a-number (NaN) values, not-a-datetime (NaT), an undefined value, or an empty space, and discretion is applied to determine what constitutes missing values in that dataset. Where missing values are found, imputations can be made by means of mean imputation, regression imputation, or a k-NN based imputation method, these approaches are covered in chapter 2. As a finding, the dataset used in this study has no missing values and thus no missing value imputations are made.

b. Outlier filtering

To observe outliers in a variable it must be visualised either by plotting the data points against time to observe values that deviate from the general pattern of the signal, or a histogram to observe values that are more distant from the mean. Outliers are removed programmatically by removing values that fall within a specified range.

c. Noise Reduction

Linear filters are an effective way to remove/reduce noise. Filters used in this study are: median filter, 1-dimensional digital filter and a Gaussian-weighted moving average filter. The choice of a filter requires that the variable be visualised, and the type of filter selected based on the desired result.

Median filter smoothing

Median filtering is a nonlinear signal processing technique that is very useful in data processing to suppress noise. The center value in a sliding window is replaced by the median of the values in the window. Given a discrete sequence $a_1, a_2, ..., a_N$ where N is odd, the median is the number in the sequence where $(N-1)/2$ elements are smaller or equal in value and $(N-1)/2$ elements are larger or equal in value [79].

In MATLAB the built-in function *medfilt1* is used to perform this task. The function applies an n^{th} order one-dimensional median filter to the input vector. The order of the filter impacts its performance, and an iterative process is used to determine the optimal order of the filter.

Digital filter

A 1-dimensional digital filter is used to filter using a rational transfer function defined by the numerator and denominator coefficients b and a [80]. The rational transfer function for the filter operates the input-output vectors in the Z-transform and is of form:

$$Y_{(z)} = \frac{b(1) + b(2)z^{-1} + \cdots + b(n_b + 1)z^{-n_b}}{1 + a(2)y(n-1) - \cdots - a(n_a + 1)y(n - n_a)} X(z)$$

The rational transfer function handles both Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. Its representation using its direct-form II transposed implementation is shown in Figure 4-6 below, where $n_a = n_b$.

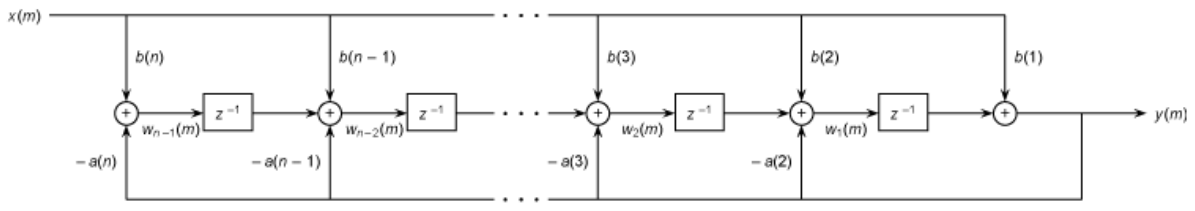

Figure 4-6. Representation of rational transfer function using its direct-form II transposed implementation [81].

In MATLAB the built-in function *filter* is used to apply the filter on data.

Gaussian-weighted moving average filter

A Gaussian filter is a filter whose impulse response is a Gaussian function, and its impulse response is given by:

$$\boldsymbol{g(x)} = \sqrt{\frac{a}{\pi}}\boldsymbol{e^{-ax^2}} \qquad 4\text{-}2$$

In MATLAB the built-in function *smoothdata* is used to apply this filter.

d. Feature extraction

Timestamp Features

Timestamp features can be extracted in MATLAB using built in functions. In this study, the hour numbers of the datetime and the number representing the day of the week are extracted. In MATLAB the built-in functions *hour* and *weekday* are used to perform this task.

Spectral features

To extract frequencies within a time window, an FFT is computed from the data points. The *fft* function in MATLAB computes a discrete Fourier transform (DFT) of the input data using a FFT algorithm. The maximum value of the output represents the fundamental frequency, and this is the value that is populated in the new data set for the computed data window.

Unique features

Understanding the physical knowledge of the system aids in establishing distinguishing characteristics of the variables in the dataset, the system from which the data was collected must be understood such that when the variables are visualised and studied there is an idea of what distinguishing features can be extracted from the raw data. Features can be a measure of shape such as the slopes or the number and values of local minima/maxima, statistical measurements such as the mean over a moving window, or other measurements that can be thought of upon visualising and studying the data.

e. Feature scaling

For data standardisation, a z-score is used, where a data point's distance from the mean is measured in terms of the standard deviation. This is done so distance-based algorithms can be trained on features with different units and ranges. To compute the z-score for a value *x* part of a variable *X* which has mean μ and standard deviation σ, the following equation is used:

$$z = \frac{(x-\mu)}{\sigma} \qquad 4\text{-}3$$

In MATLAB the built-in function *zscore* is used to perform this task.

f. Dimensionality reduction

PCA is used to reduce dimensionality in the dataset. The principal components have to explain at least 99% of the variance to preserve much of the information.

Chapter 5. Application

The previous chapters provided an outlook on the research in intrusion detection that other researchers have conducted, the data selected for this study and the methodology followed in this study. This chapter gives a detailed description of the application of knowledge from the previous sections for ML modelling. The approach is to preprocess SCADA level data, apply it to a machine learning model and evaluating the results with the purpose of detecting attacks. The code developed for this study is stored in a GitHub repo [82] and references to code files are provided in relevant sections of this thesis.

5.1 Computing resources and parallel computing

Computational resources are an important part of machine learning as they dictate how quickly an algorithm takes to learn. The technical work in this study was conducted on a Proline desktop computer running a Windows 10 Pro operating system with the following specifications:

- Device name: DESKTOP-KU5HAKB
- Processor: Intel(R) Core (TM) i7-4790 CPU @ 3.60GHz 3.60 GHz
- CPU Cores: 4
- GPU logical processors: 8
- Installed RAM: 16,0 GB (15,8 GB usable)
- Device ID: E39D8E46-6C50-43C5-8151-0700D1EBFF21
- Product ID: 00331-10000-00001-AA568
- System type: 64-bit operating system, x64-based processor

Parallel computing allows for splitting of large problems into smaller ones and carrying out calculations simultaneously. The main benefits of using parallel computing for this study are: (1) saving time by distributing tasks and executing them simultaneously and, (2) solving big data problems by distributing data. The software used in this study is MATLAB® which supports parallel computing through the Parallel Computing Toolbox which allows machine learning tasks to be divided and run on multiple workers which are the cores in the CPU, in this case the maximum is 4. Figure 5-1 shows the partitioning of a CPU to have multiple parallel workers.

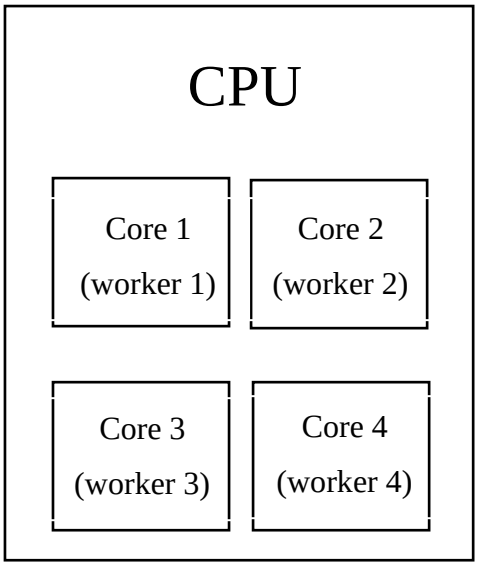

Figure 5-1. CPU partition

5.2 Modelling approach

The modelling approach discussed in chapter 4 is applied in this chapter.

5.2.1 Problem Exploration

In this section the data is explored to provide an understanding of the data before moving on to further steps in the model workflow.

a. Visualising attacks against sensor and actuator data

As mentioned in the data section above, there are 4 types of attacks in the SWAT dataset, namely SSSP, SSMP, MSSP and MSMP. To give an idea of how each of the attacks affect the data, an example of each attack has been selected and visualised against data of the components under attack below. It is clear from observing the attacks that a deviation from the normal trajectory of the data points is created. This exercise is done to understated how the attacks affect the various variables in the dataset and this insight dictates the selection of appropriate data preprocessing techniques.

On the 28th of December 2015 between times 11:22:00 and 11:28:22, an SSSP attack happens on level transmitter sensor LIT101 which measures the raw water tank level in raw water supply & storage stage (P1). The attack increases the water level by 1mm every second to underflow the tank and also damage pump P101. The attack is visualised in Figure 5-2 below.

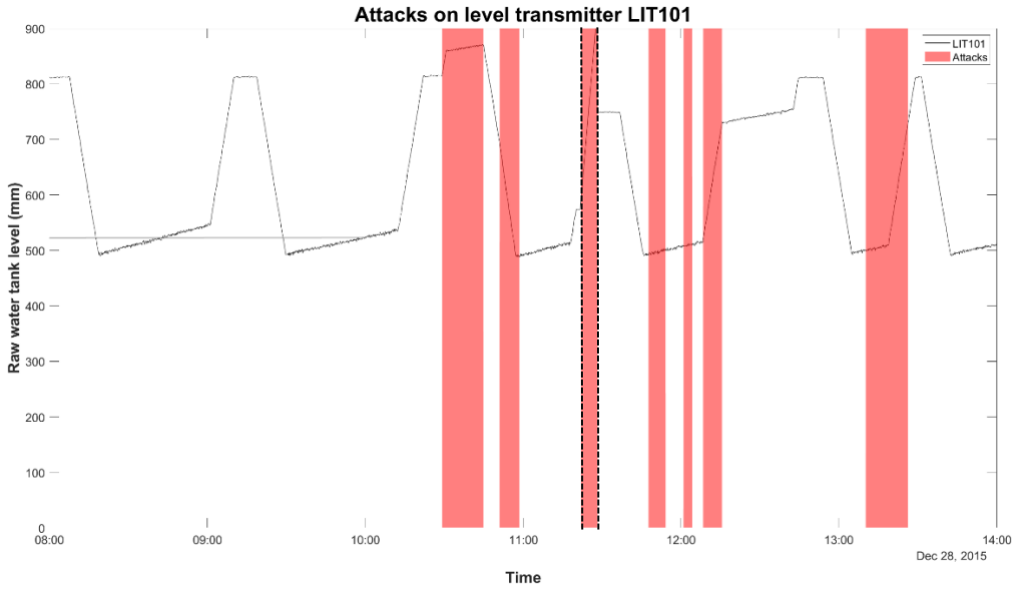

Figure 5-2. Attacks on level transmitter

On the 29th of December 2015 between times 18:30:00 and 18:42:00, an SSMP attack happens in raw water supply & storage stage (P1) on motorised valve MV101 which controls water flow to the raw water tank and level transmitter sensor LIT101 mentioned previously. The attack keeps MV101 on continuously and sets the value of LIT101 to a constant of 700mm to overflow the raw water tank. The attack is visualised in Figure 5-3 below.

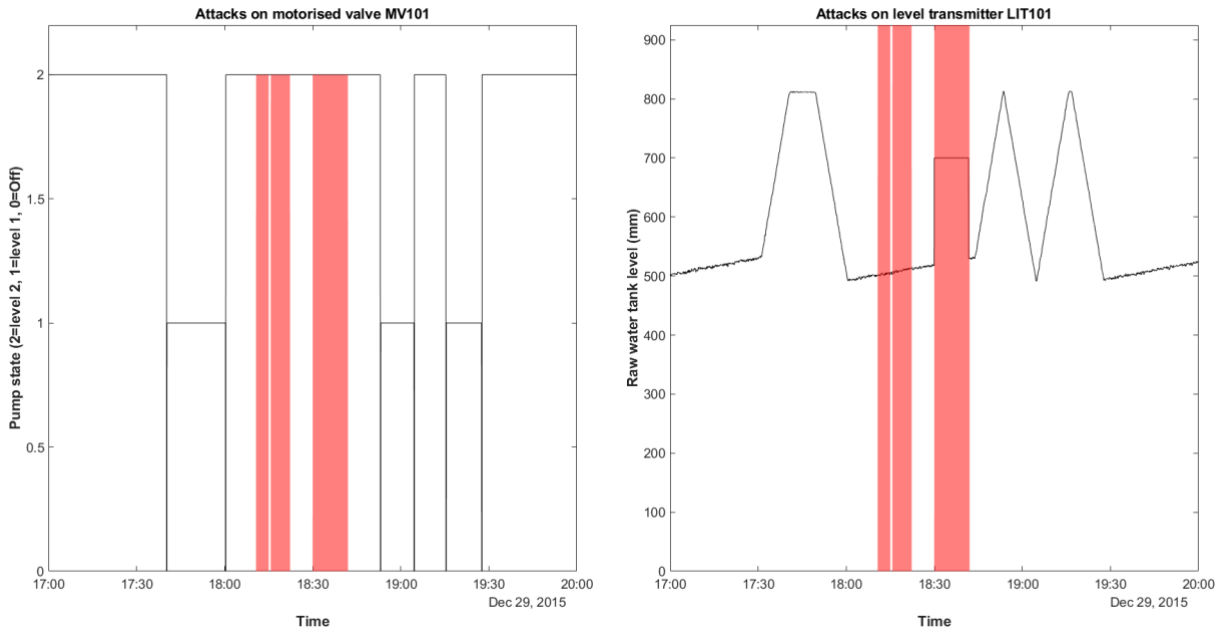

Figure 5-3. Left: Attacks on motorised valve MV10. Right: Attacks on level transmitter LIT101

On the 30th of December 2015 between times 17:04:56 and 17:29:00, a MSSP attack happens in both raw water supply & storage stage (P1) and UF stage (P3). In stage P1 the attack happens

on pump P101 which Pumps water from raw water tank to second stage (stage P2). In stage P3 the attack happens on level transmitter sensor LIT301 which measures the UF feed water tank level. The attack keeps pump P101 on continuously and sets the value of LIT301 to a constant of 801mm to underflow the raw water tank and overflow UF feed water tank. The attack is visualised in Figure 5-4 below.

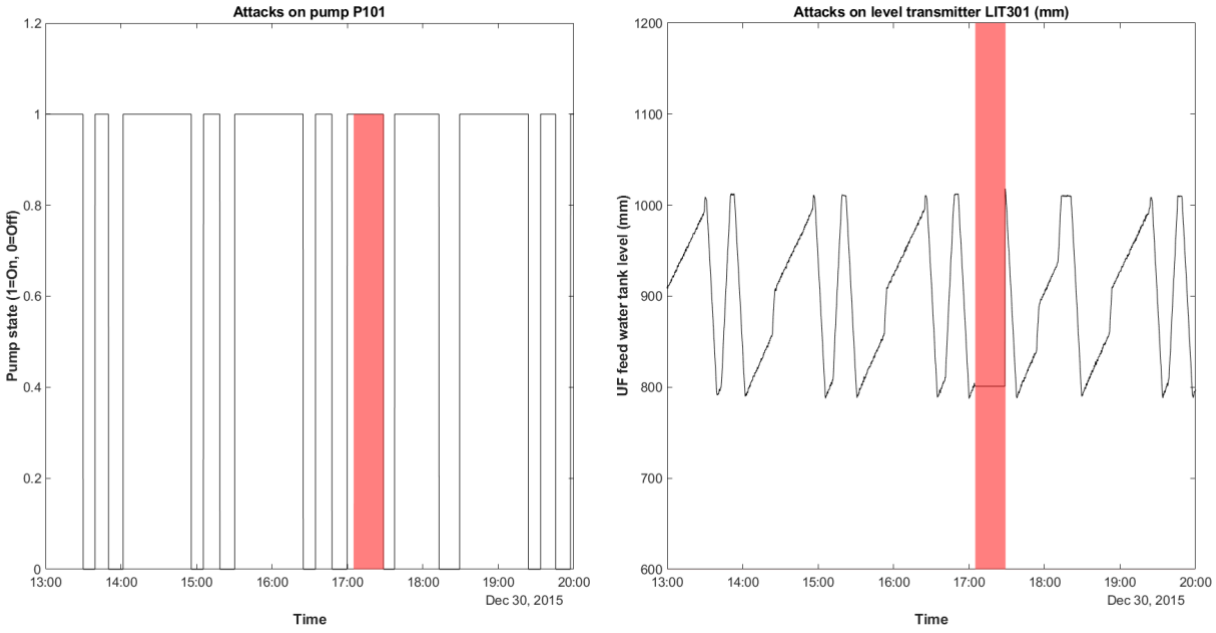

Figure 5-4. Left: Attacks on pump P101. Right: Attacks on level transmitter LIT301

On the 29th of December 2015 between times 22:55:18 and 23:03:00, an MSMP attack happens in dechlorination stage (P4) and RO stage (P5). In stage P4 the attack happens on the dechlorinating actuator UV401 which removes chlorine from water. In stage P5 the attacks happen on both the RO pH analyser sensor AIT501 which measures HCl level, and pump P501 which pumps dechlorinated water to RO stage. The attack stops UV401 from operating, sets the value of AIT501 to a constant of 150 and forces P501 to remain on. The attack is visualised in Figure 5-5 below.

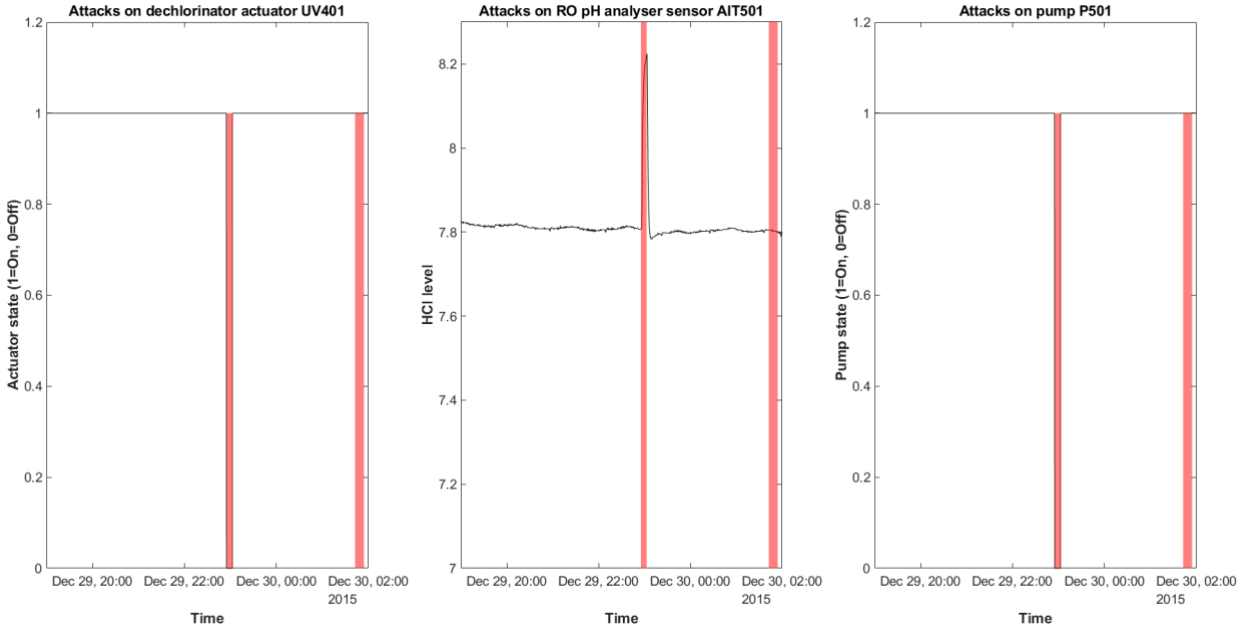

Figure 5-5. Left: Attacks on dechlorinator actuator UV401. Middle: Attacks on RO pH analyser sensor AIT501. Right: Attacks on pump P501

b. Pump actuators

The pumps used in SWaT go in pairs, one is the main pump, and the other is a backup pump. The backup pump either pumps alongside the main pump or pumps when the main pump fails. This purports that they can be considered as a single pump and their values merged. Since the pump data is logical with a value of 0 indicating that the pump is off and a value of 1 indicating that it is on, to merge data from 2 pumps an Inclusive OR operation is used where the output is 1 when at least one of the values is 1 and 0 if both values are 0. The resultant variable of the merged pumps data takes on the name of the pump whose name has the least numeric value. i.e., Pump P101 merged with pump P102 is a new variable called P101. The pump pairs and resultant variable can be seen in Table 5-1 below.

Table 5-1.Table showing pump pairs (main and backup) and resultant variable.

Main pump	**Backup pump**	**Merged data variable name**
P101	P102	P101
P201	P202	P201
P203	P204	P203
P205	P206	P205
P302	P301	P301
P402	P401	P401
P403	P404	P403
P501	P502	P501

c. Dataset reduction by correlation analysis

Correlation analysis is a study to determine whether there is a linear relationship between variables. This method is used in the study to identify closely related variables in the dataset, further investigate the relationship and drop variables from the dataset should they be directly correlated to others. This exercise ensures that processing techniques are only applied to a reduced dataset with non-redundant variables. Correlation matrices of plots are generated in MATLAB for this task, scatter plots of variable pairs appear in the off-diagonal region, while histograms of the variables are plotted along the matrix diagonal. The slopes of least squares reference lines for these scatter plots are equal to displayed correlation coefficients, high correlation is indicated by Pearson correlation coefficient values less than -0.8 and greater than 0.8. Equation 2-13 in Section 2.3.4 showed how Pearson correlation coefficients are computed.

a. Stage P1: Raw water supply & storage

A correlation matrix of plots showing correlation among variables in the raw water supply & storage stage (P1) is presented in Appendix B and the results are presented in Table 5-2 below. The variables explored are FIT101, LIT101, MV101 and P101.

Table 5-2. Correlation coefficients among variables in stage P1.

	FIT101	**LIT101**	**MV101**	**P101**
FIT101	-	-0.67	0.97	0.22
LIT101	-0.67	-	-0.67	-0.68
MV101	0.97	-0.67	-	0.23
P101	0.22	-0.68	0.23	-

Observations:

- FIT101 measuring flow into raw water tank is highly correlated to MV101 which controls water flow to the raw water tank at a correlation coefficient of 0.97.
- LIT101 measuring the raw water tank level and MV101 share a negative correlation since they share an inversely proportional relationship.
- LIT101 and P101 which pumps water from raw water tank share a negative correlation since they share an inversely proportional relationship.

Features to be dropped:

- MV101 is redundant to FIT101 and thus can be dropped since it is actuator data and not much explanatory data can be extracted from it.

b. Stage P2: Chemical dosing

Correlation matrices of plots showing correlation among variables in chemical dosing stage (P2) are presented in presented in this section. Due to computing resource limitations, the *corrplot* function in MATLAB can only accept 7 of the variables at a time, as a result 3 correlation matrix plots were generated and are presented in Appendix B and the results are presented in Table 5-2 below. The variables explored are AIT201, AIT202, AIT204, FIT201, MV201, P201, P203, P205.

Table 5-3. Correlation coefficients among variables in stage P2.

	AIT201	**AIT202**	**AIT203**	**FIT201**	**MV201**	**P201**	**P203**	**P205**
AIT201	-	-0.76	0.26	0.06	0.06	-0.38	0.06	0.03
AIT202	-0.76	-	-0.05	-0.30	-0.30	0.28	-0.30	-0.25
AIT203	0.26	-0.05	-	0.11	0.11	-0.13	0.11	-0.19
FIT201	0.06	-0.30	0.11	-	0.98	0.12	1.00	0.91
MV201	0.06	-0.30	0.11	0.98	-	0.12	0.98	0.90
P201	-0.38	0.28	-0.13	0.12	0.12	-	0.12	0.14
P203	0.06	-0.30	0.11	1.00	0.98	0.12	-	0.91
P205	0.03	-0.25	-0.19	0.91	0.90	0.14	0.91	-

Observations:

- AIT201 measuring NaCl water content level and AIT202 measuring HCl water content level have high correlation but within acceptable limits at a correlation coefficient of -0.76.
- MV201, an actuator that controls water flow to the UF feed water tank is highly correlated to FIT201, a sensor that measures water flow, at a correlation coefficient of 0.91.
- P203, an NaCl dosing pump, is highly correlated to MV201 at a correlation coefficient of 0.98.
- P203 and FIT201 are perfectly correlated at a correlation coefficient of 1.
- P205, an NaOCl dosing pump, is strongly correlated to 3 features: FIT201, MV201 and P203 at correlation coefficients of 0.91, 0.9 and 0.91 respectively.

Features to be dropped:

- FIT201 explanation of the events in stage P2 is tantamount to variables MV201, P203 and P205 and due to the redundancy only FIT201 is retained.

c. Stage P3: Ultrafiltration

Correlation matrices of plots showing correlation among variables in ultrafiltration stage (P3) are presented in this section. Due to computing resource limitations mentioned previously, 3 correlation matrix plots were generated and are presented in Appendix B and the results are presented in Table 5-4 below. The variables explored are DPIT301, FIT301, LIT301, MV301, MV302, MV303, MV304, P302.

Table 5-4. Correlation coefficients among variables in stage P3.

	DPIT301	**FIT301**	**LIT301**	**MV301**	**MV302**	**MV303**	**MV304**	**P302**
DPIT301	-	0.96	-0.49	-0.12	0.93	-0.25	-0.34	0.95
FIT301	0.96	-	-0.49	-0.11	0.90	-0.23	-0.26	0.98
LIT301	-0.49	-0.49	-	-0.02	-0.47	-0.02	0.17	-0.48
MV301	-0.12	-0.11	-0.02	-	-0.10	0.42	-0.01	-0.11
MV302	0.93	0.90	-0.47	-0.10	-	-0.22	-0.42	0.90
MV303	-0.25	-0.23	-0.02	0.42	-0.22	-	0.17	-0.28
MV304	-0.34	0.26	0.17	-0.01	-0.42	0.17	-	-0.24
P302	0.95	0.98	-0.48	-0.11	0.90	-0.28	-0.24	-

Observations:

- FIT301, a flow sensor measuring the flow of RO reject, is highly correlated to three features at correlation coefficients of 0.96, 0.9 and 0.98 respectfully, namely: (1) DPIT301, a sensor that transmits differential pressure to control the back wash process, (2) MV302, a valve actuator that controls UF backwash drain, and (3) P302, a pump actuator which pumps water from UF feed water tank to RO feed water tank via UF filtration.

Features to be dropped:

- FIT301 explanation of the events in stage P3 is tantamount to variables DPIT301, MV302 and P302 and due to the redundancy only FIT301 is retained, this variable can be processed to extract informative features.

d. Stage P4: Dechlorination

A correlation matrix of plots showing correlation among variables in the dechlorination stage (P4) is presented in Appendix B and the results are presented in Table 5-5 below. The variables explored are AIT401, AIT402, FIT401, LIT401, P402, P403, UV401.

Table 5-5. Correlation coefficients among variables in stage P4.

	AIT401	**AIT402**	**FIT401**	**LIT401**	**P402**	**P403**	**UV401**
AIT401	-	-0.19	-0.03	-0.04	-0.05	0.00	-0.05
AIT402	-0.19	-	-0.86	-0.67	-0.85	0.03	-0.85
FIT401	-0.03	-0.86	-	0.79	0.98	-0.00	0.99
LIT401	-0.04	-0.67	0.79	-	0.380	-0.03	0.79
P402	-0.05	-0.85	0.98	0.80	-	0.00	0.98
P403	0.00	0.03	-0.00	-0.03	0.00	-	0.00
UV401	-0.05	-0.85	0.99	0.79	0.98	0.00	-

Observations:

- Four variables are closely correlated considering that they form part of the dechlorination process however are not close enough to drop either. Namely: (1) AIT402, a sensor that measures ORP to control the $NaHSO_3$ dosing by pump P204 and the NaOCl dosing by pump P205, (2) FIT401, a sensor that measures the flow in order to control the UV dechlorinator, (3) P402, a pump actuator that pumps water from the RO feet tank to the UV dechlorinator, and (4) UV401, a dechlorinator actuator that removes from the water.
- FIT401 is responsible for P402 UV401 are closely correlated because one controls the other.

Features to be dropped:

- P402 and UV401 will be dropped since they are redundant to FIT401.

e. Stage P5: Reverse osmosis

Correlation matrices of plots showing correlation among variables in the reverse osmosis stage (P5) are presented in this section. Due to computing resource limitations mentioned previously, 5 correlation matrix plots were generated and are presented in Appendix B and the results are presented in Table 5-6 below. The variables explored are AIT501, AIT502, AIT503, AIT504, FIT501, FIT502, FIT503, FIT504, P501, PIT501, PIT502, and PIT503.

Table 5-6. Correlation coefficients among variables in stage P5 *(table too wide and is split in 2).*

	AIT501	AIT502	AIT503	AIT504	FIT501	FIT502
AIT501	-	-0.31	0.17	-0.22	0.70	0.69
AIT502	-0.31	-	-0.10	-0.13	-0.63	-0.65
AIT503	0.17	-0.10	-	0.04	-0.05	-0.05
AIT504	-0.22	-0.13	0.04	-	-0.08	-0.08
FIT501	0.70	-0.63	-0.05	-0.08	-	1.00
FIT502	0.69	-0.65	-0.05	-0.08	1.00	-
FIT503	0.70	-0.58	-0.05	-0.11	0.99	0.99
FIT504	0.68	-0.58	-0.05	-0.10	0.98	0.98
P501	0.68	-0.61	-0.05	-0.09	0.99	0.99
PIT501	0.71	-0.56	-0.06	-0.11	0.99	0.98
PIT502	0.52	-0.16	-0.10	-0.13	0.60	0.59
PIT503	0.72	-0.54	-0.05	-0.12	0.99	0.98

	FIT503	FIT504	P501	PIT501	PIT502	PIT503
AIT501	0.70	0.68	0.68	0.71	0.52	0.72
AIT502	-0.58	-0.58	-0.61	-0.56	-0.16	-0.52
AIT503	-0.05	-0.05	-0.05	-0.06	-0.10	-0.05
AIT504	-0.11	-0.10	-0.09	-0.11	-0.13	-0.12
FIT501	0.99	0.98	0.99	0.99	0.60	0.99
FIT502	0.99	0.98	0.99	0.98	0.59	0.98
FIT503	-	0.99	0.99	1.00	0.61	1.00
FIT504	0.99	-	0.99	0.99	0.60	0.98
P501	0.99	0.99	-	0.99	0.60	0.98
PIT501	1.00	0.99	0.99	-	0.61	1.00
PIT502	0.61	0.61	0.60	0.61	-	0.61
PIT503	1.00	0.98	0.98	1.00	0.61	-

Observations:

- Seven variables are perfectly correlated at a correlation coefficient of 0.99 between each other and can all be explained by just one of the variables. These variables are: (1) FIT501, a sensor measuring inlet flow of RO membrane, (2) FIT502, a sensor measuring RO Permeate flow, (3) FIT503, a sensor measuring RO Reject flow, (4) FIT504, a sensor measuring RO re-circulation flow, (5) P501, a pump actuator that pumps dechlorinated water to RO, (6) PIT501, a sensor measuring RO feed pressure, and PIT503, a sensor measuring RO permeate pressure.

Features to be dropped:

- Upon observation of the abovementioned variables, it can be hypothecated that the simple data of P501 can best describe the events of stage P5 in place of variables FIT501, FIT502, FIT503, FIT504, PIT501 and PIT503, which will be dropped from the dataset.

f. Stage P6: RO permeate transfer, UF backwash

In raw water supply & storage stage (P1), only pump P602 which pumps water from UF back wash tank to UF was used in the data collection process. The other variable, pump P601 which pumps water from RO permeate tank to raw water tank was not used and therefore can be dropped from the dataset. A correlation matrix plot are not performed since there is only one variable in this stage.

Resultant dataset

Following the process undertaken above to reduce the dataset by means of correlation, a reduced dataset is resultant and is described in detail in Table 5-7 below which shows the dropped and retained variables.

Table 5-7. Dataset variables at this stage.

No	Type	Retained variables	Type	Dropped variables
1	Sensor	FIT101	Actuator	MV101
2	Sensor	LIT101	Actuator	MV201
3	Actuator	P101	Actuator	P203
4	Sensor	AIT201	Actuator	P205
5	Sensor	AIT202	Sensor	DPIT301
6	Sensor	AIT203	Actuator	MV302
7	Sensor	FIT201	Actuator	P302
8	Actuator	P201	Actuator`	P402
9	Sensor	FIT301	Actuator	UV401
10	Sensor	LIT301	Sensor	FIT501
11	Actuator	MV301	Sensor	FIT502
12	Actuator	MV303	Sensor	FIT503
13	Actuator	MV304	Sensor	FIT504
14	Sensor	AIT401	Sensor	PIT501
15	Sensor	AIT402	Sensor	PIT503
16	Sensor	FIT401	Actuator	P601
17	Sensor	LIT401		
18	Actuator	P403		
19	Sensor	AIT501		
20	Sensor	AIT502		
21	Sensor	AIT503		
22	Sensor	AIT504		
23	Actuator	P501		
24	Sensor	PIT502		
25	Sensor	FIT601		
26	Actuator	P602		

5.2.2 Formulating the problem

In this case we have a classification problem, as the dataset includes attacks that need to be identified. Supervised learning is a commonly used approach for anomaly detection in water treatment systems data. The use of supervised learning is driven by several factors, including the availability of labelled data, interpretability, high accuracy, fast detection, and scalability.

a. The success of anomaly detection models depends on the availability of labelled data with normal and abnormal behaviour, as this allows for the training of supervised learning models to differentiate between the two [83].

b. Supervised learning models are easy to understand making it easy to determine the cause of anomalies. This interpretability is important in real-world applications where understanding the source of anomalies is crucial.

c. Supervised learning models have been shown to be accurate in anomaly detection, especially when the data is well labelled and the model is designed and optimized properly [84]. Additionally, these models can detect anomalies in real-time, making them suitable for online monitoring and control applications [84].

d. Supervised learning models are capable of handling large amounts of data and can be scaled to suit large-scale water treatment systems. This scalability is necessary to ensure that the anomaly detection system can continue to function effectively as the water treatment system grows [74].

5.2.3 Baseline Models

For this exercise two ML algorithms namely: Fine Tree and Boosted Trees Ensemble, are trained on raw data, which is the resultant dataset from the correlation analysis in the previous section. The models are evaluated on metrics identified in the literature review, they are: Test accuracy, F1 score which is calculated from Precision and Recall, and the time it takes to detect an attack, referred to as the TTD.

Attacks in the test data are shown in Figure 5-6 below and the time to detection is evaluated on the attack that happened between the 31st of December 2015 at 01:17:08 and the 31st of December 2015 at 11:15:27.

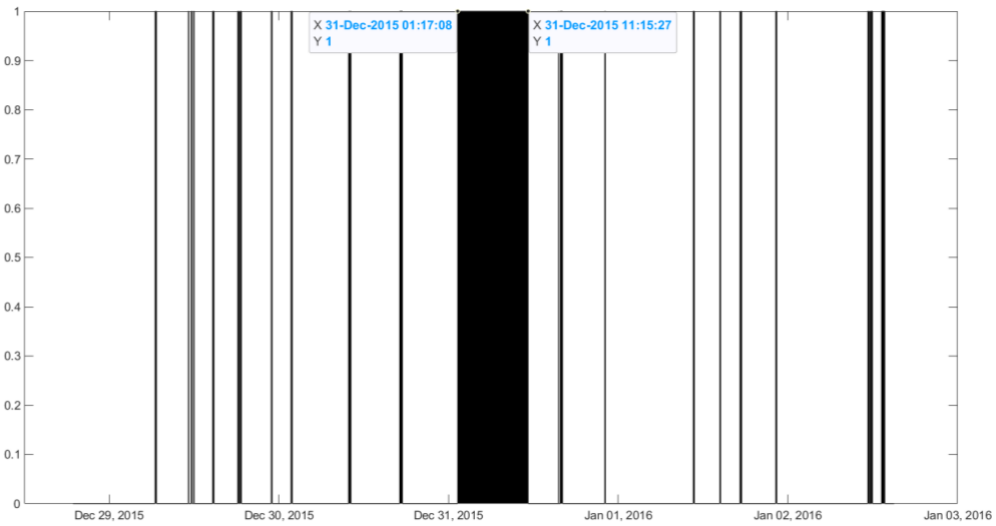

Figure 5-6. Test data attacks

The parameters applied when training the models are given in Table 5-8 below. These model parameters are carried over to the experiments to maintain consistency and have fair results. The choice of the parameters is based on the learnings from the literature review.

Table 5-8.Model parameters:

Training data	70% of dataset
Testing data	30% of dataset
Validation scheme	Cross validation with 2 folds
Misclassification costs	TP = 0, FN = 5, FP = 1, TN = 0

5.2.4 Data Preprocessing

A systemic approach for preprocessing outlined in chapter 4 is applied in this section, where each variable is studied, and the appropriate preprocessing steps are applied on it. Data preprocessing steps outlined in chapter 4 form a standard approach to preprocessing a variable and not all the steps are to be applied as each variable has its own unique form.

Timestamp data preprocessing

Certain events may be described by the time they occur and thus timestamp features are extracted from the datetime feature in the dataset. The hour and weekday provide enough frequency for the data to be informative and are the two features extracted. The code used to perform this task can be found in [85].

LIT101 data preprocessing

In stage P1 of the SWaT system, LIT101 is a sensor that measures raw water tank level. Figure 5-3 showed an attack where the aim is to overflow the raw water tank by setting the water level

to a constant 700mm. The steps taken to process the LIT101 sensor data are given below, and the code can be found in [86].

1. Raw data is replaced with a smoothed signal. A median filter is chosen for its ability to keep edges.
2. The signal appears to be periodic thus high prominence peaks are be extracted and differences between them populated as a new feature.
3. The signal is composed of lines and vertices and therefore the slopes of the derivatives are calculated to determine the changing slopes and populated as a new feature. This implementation is illustrated in Figure 5-7.

Figure 5-7. LIT101 sensor data process for extracting slopes.

Median filter smoothing

In applying a median filter to LIT101 data, a 600th order and a truncate zero-padding method is applied to computes medians of smaller segments as it reaches the signal edges.

```
LIT101MedFilt = medfilt1(LIT101,600,'truncate');
```

The above MATLAB code was used to filter the signal and the result can be seen in Figure 5-8 below where the noise was suppressed, and a smooth signal resulted.

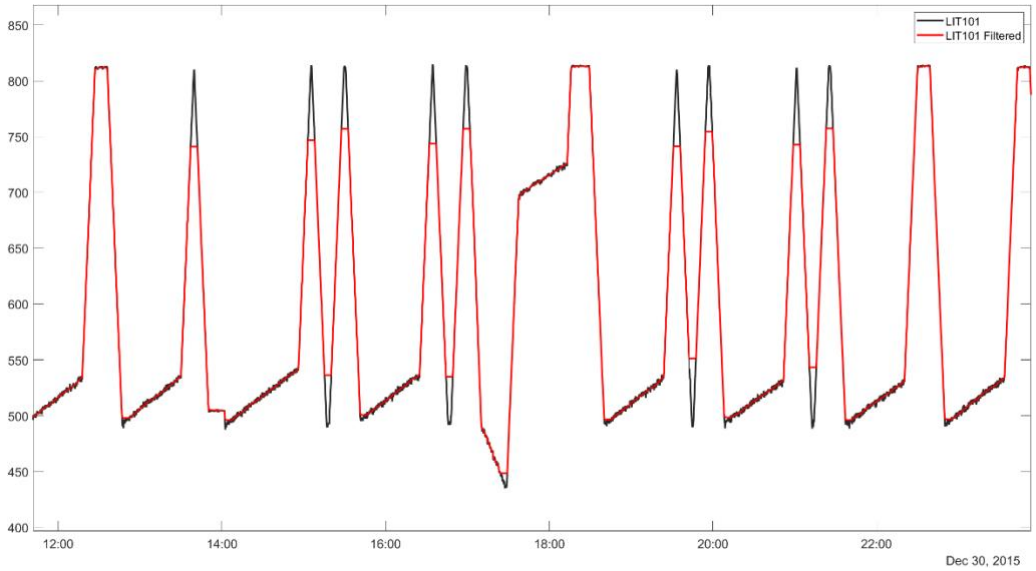

Figure 5-8. Plots of LIT101 raw and filtered data

Extracting Peaks

In Figure 5-9 local minima of high prominence for a filtered LIT101 signal are seen marked in red. The extraction of the minima is achieved by a custom code which can be found in GitHub link provided above.

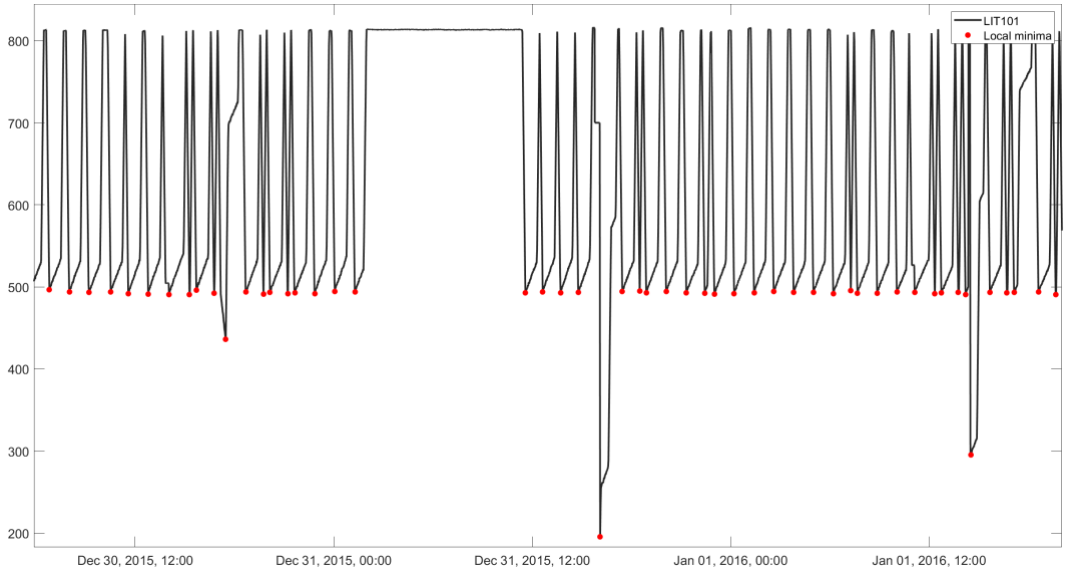

Figure 5-9. LIT101 local minima of high prominence

Approximating slopes: dYdT=Δy/Δt

To derive the slopes the process in Figure 5-7 is used. The data is first smoothed using a 200th order one-dimensional median filter with a truncate zero-padding method applied. The data is the resampled at a factor of 1/10. The difference between the resulting data points is computed and then the difference data is resampled at a factor of 10. The MATLAB code to achieve this is shown below.

```
LIT101MedFilt = medfilt1(LIT101,200,'truncate');
reLIT = resample(LIT101MedFilt,1,10);
LITdata = medfilt1(reLIT ,10,'truncate');
dX = diff(LITdata);
dXLIT101 = resample(dX,10,1);
```

FIT101 data preprocessing

This sensor is part of stage P1 of the secure water treatment system and measures inflow of raw water into raw water tank. FIT101 can be seen in Figures 5-10 and 5-11.

The processing of this feature results in 3 new features, namely: Smoothed signal without outliers, Peaks, and frequencies. The code can be found in [87].

1. The signal is composed of sinusoidal segments when there is inflow into the tank. The segments can be inspected individually, and the individual frequencies are extracted. The following steps are used to achieve this.
    a. Outliers below the value 2.4 are removed.
    b. The signal is smoothed to get a smooth sinusoid
        i. Moving mean smoothing is used. This method reduces the peaks however this is not an issue when only focusing on frequency.
    c. Extract frequencies in segments and combine in a new cell
2. The signal is smoothed, outliers are removed, and local maxima are extracted.
    a. Median smoothing used. This method does not give a near perfect sinusoid however the peaks are maintained.

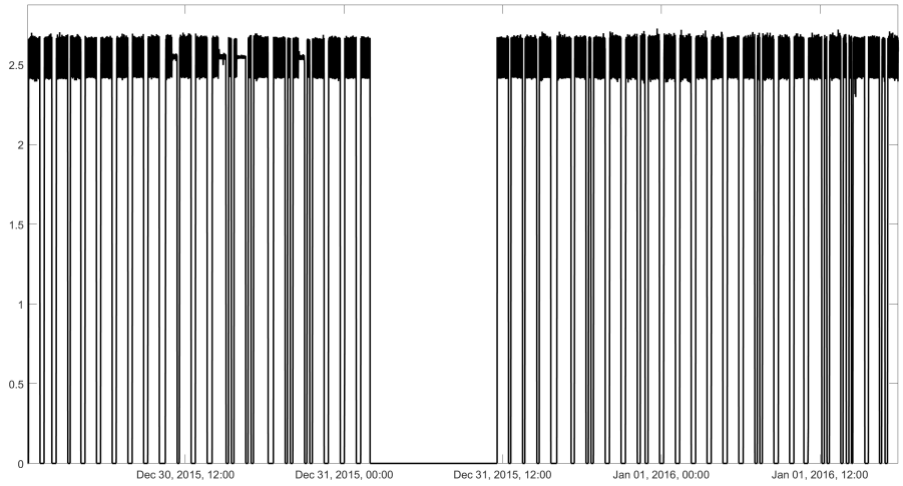

Figure 5-10. Plot of FIT101

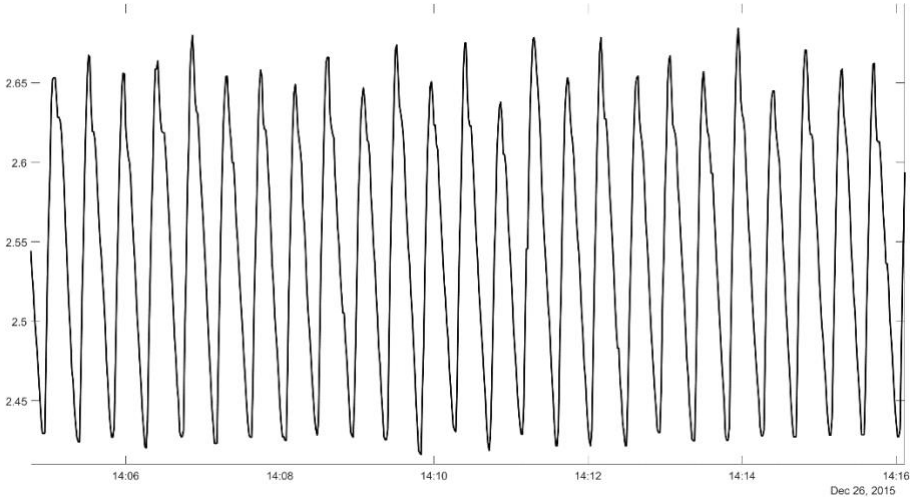

Figure 5-11. FIT101 zoomed in to show sinusoidal structure

Smoothing and frequency extraction

When inflow into the tank is turned on, the inflow rate rises to a point where it starts to stabilize. Similarly, when it is turned off, the inflow rate drops to 0. Values observed before the stability is reached can then be removed for the purposes of frequency extraction, values below 2.4 are thus removed from the data.

The data is smoothed by moving mean smoothing, a technique commonly used in time series forecasting. A sliding window is passed through the time series data to calculate the average values in the new series. The formula for a simple moving average is:

$$\bar{y}_t = \frac{y_t + y_{t-1} + \cdots + y_{t-n-1}}{n} \qquad 5\text{-}1$$

Where *y* is the variable, *t* is the time period, *n* is the number of time periods in the averaging window. The MATLAB code for this task uses the *smoothdata* built-in function which is set to use a moving mean technique.

```
FIT101Smoothed = smoothdata(FIT101,'movmean');
```

The resultant signal is a smooth sinusoid, shown in Figure 5-12, which will yield good results when an FFT is applied to extract the fundamental frequency.

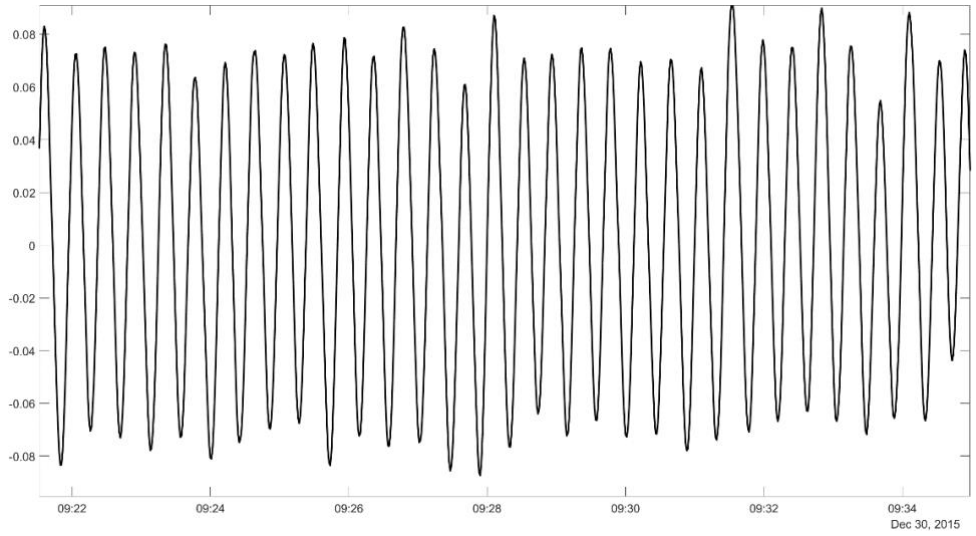

Figure 5-12. FIT101 signal after moving mean smoothing.

To extract the frequencies between states of water inflow, custom code that uses the built in *fft* function in MATLAB was written and can be found in the GitHub link provided above. Figure 5-13 is a power spectrum computed for FIT101 samples 279053 to 279729, which shows the signals frequency at 0.078Hz normalized frequency.

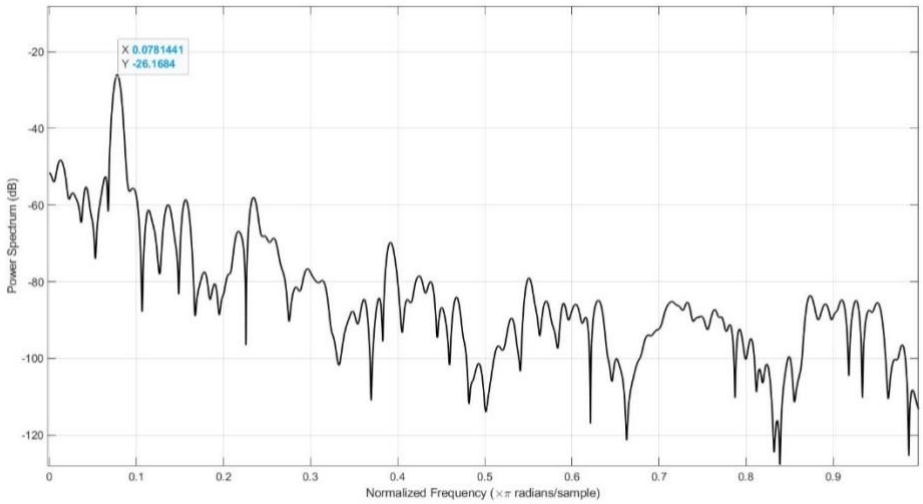

Figure 5-13. Power spectrum computed from a window on FIT101 data.

Median Smoothing

Median smoothing is used as a smoothed signal feature extraction method, this method is chosen as it has a better ability of maintaining peaks in the data as compared to moving mean smoothing. The same procedure as with LIT101 processing is used.

Peak extraction

Similar to LIT101 processing, the peaks can be a good indication of how the system is behaving. Local maxima are extracted is a similar manner to LIT101 processing.

AIT201 data preprocessing

This sensor is part of stage P2 of the secure water treatment system. It is a conductivity (μS/cm) analyser which measures NaCl level in the water which is dosed into the water by dosing pump P201 and P202 as a backup. A plot of AIT201 data is show in Figure 5-14.

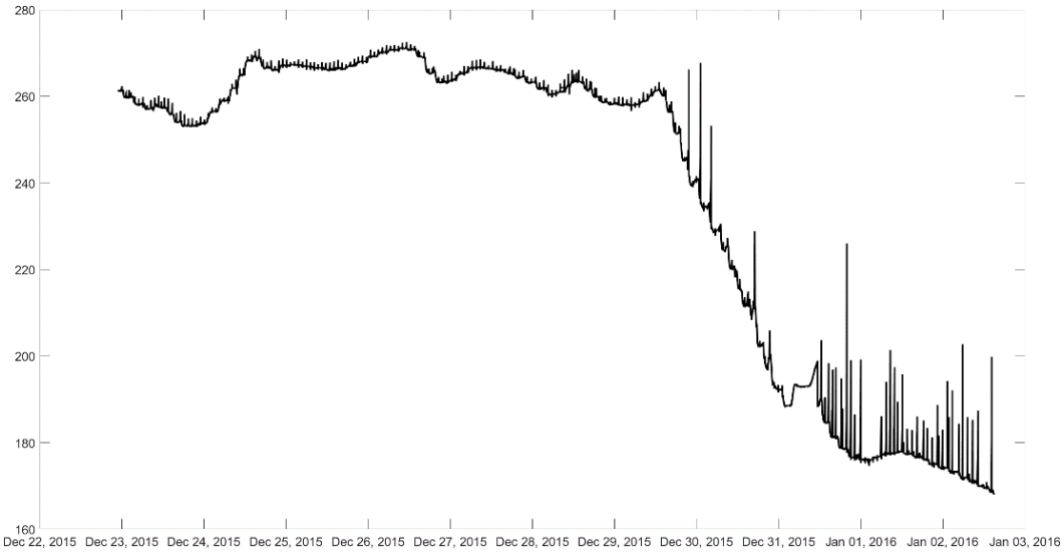

Figure 5-14. Plot of AIT201

The data is noisy, and the processing of this feature results in one feature which is data that has been filtered. A 1-dimensional digital filter is used to filter input AIT201 data using a rational transfer function described in chapter 4. The implementation of the filter on the data vector in MATLAB is achieved by a built-in *filter* function. The coefficient *b* is defined by an equation that uses the window length of a sliding window which is set to 600 samples. Coefficient *a* is set to 1 and the code can be seen below, and the output signal is shown in Figure 5-15 where it is compared to the input signal.

```
windowSize = 600;
b = (1/windowSize)*ones(1,windowSize);
a = 1;
AIT201Filtered = filter(b,a,AIT201);
```

Figure 5-15. Plot showing AIT201 raw and filtered data.

AIT202 data preprocessing

This sensor is part of stage P2 of the secure water treatment system and is a pH analyser which measures HCL level in the water dosed by dosing pump P203 and P204 as a backup. A plot of AIT202 data is show in Figure 5-16.

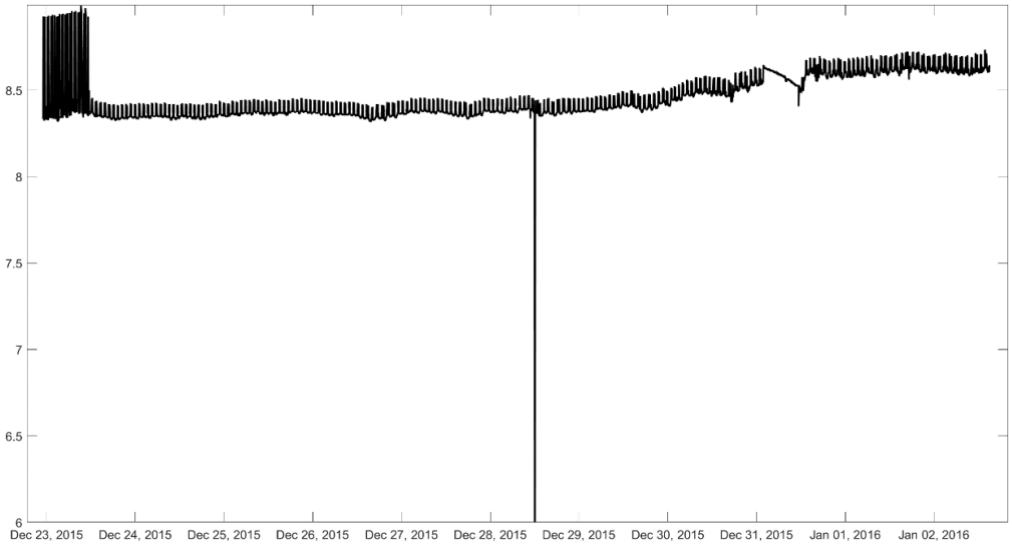

Figure 5-16. Plot of AIT202

The processing of this data results in 4 new features, namely: Smoothed and detrended signal, local minima, and local maxima. The code can be found in [88]:

1. The signal is linearly detrended to reveal any subtrends.
2. The signal is smoothed using Gaussian smoothing
3. Local minima and local maxima are extracted.
4. The time differences between peaks are extracted.

Detrending and Smoothing

In order to see subtrends in the time series data which appears to trend, a detrend can be done. The built in MATLAB function *detrend* is used to remove a linear trend in the data by removing the best straight-fit line from the data. The following command is used to perform this task:

```
AIT202Detrended = detrend(AIT202,'linear');
```

To smooth the data, a Gaussian-weighted moving average filter is applied to the noisy data. To achieve this in MATLAB, the following command is used:

```
AIT202Filtered = smoothdata(AIT202Detrended,'gaussian');
```

The resultant data is shown in Figures 5-17 and 5-18 where it is compared to the input raw data.

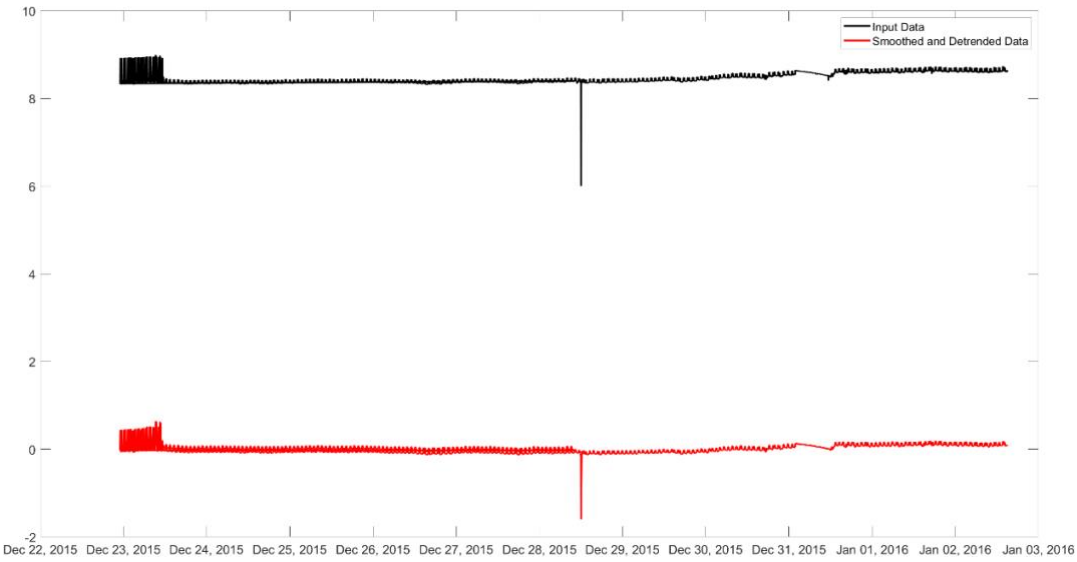

Figure 5-17.Plot showing AIT202 raw and smoothed and detrended data.

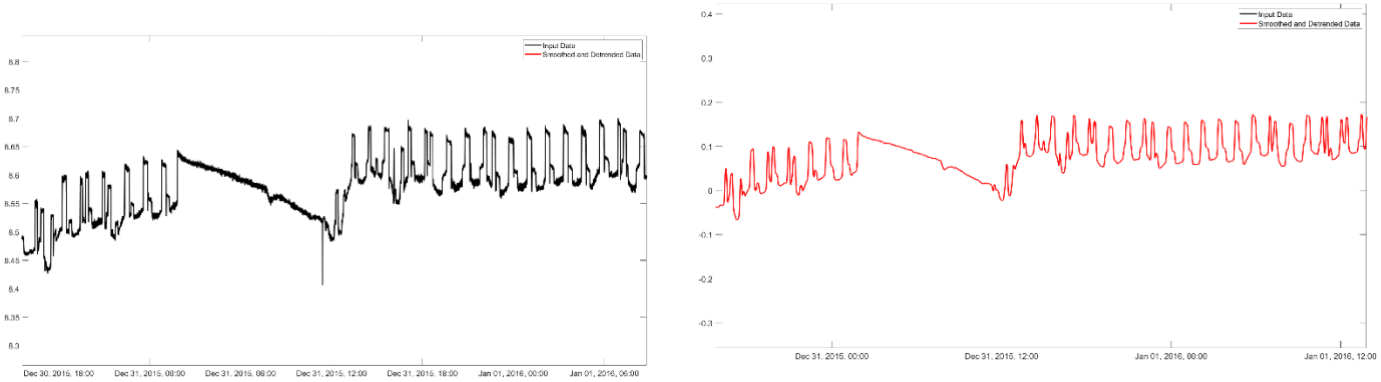

Figure 5-18. Left: Plot of raw AIT202 data. Right: Plot of smoothed and detrended AIT202 Data.

Local minima and local maxima

The local minima and local maxima points in the data as seen in Figure 5-19 are extracted as before into 2 new features, this information can be useful in training the model. The time difference between the peaks can be a differentiator between normal and abnormal behaviour, to extract this information custom code was written and can be found in the aforestated link.

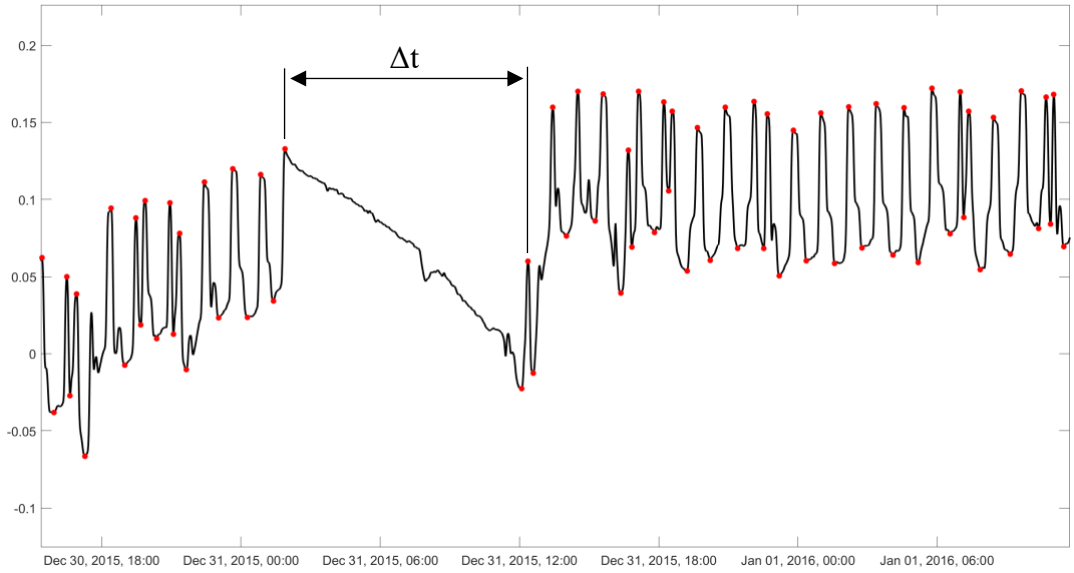

Figure 5-19. Plot of processed AIT201 data showing local minimum and maximum extrema.

AIT203 data preprocessing

This sensor is part of stage P2 of the secure water treatment system and is an ORO analyser which measures NaOCl level in the water. A plot of AIT203 data is show in Figure 5-20.

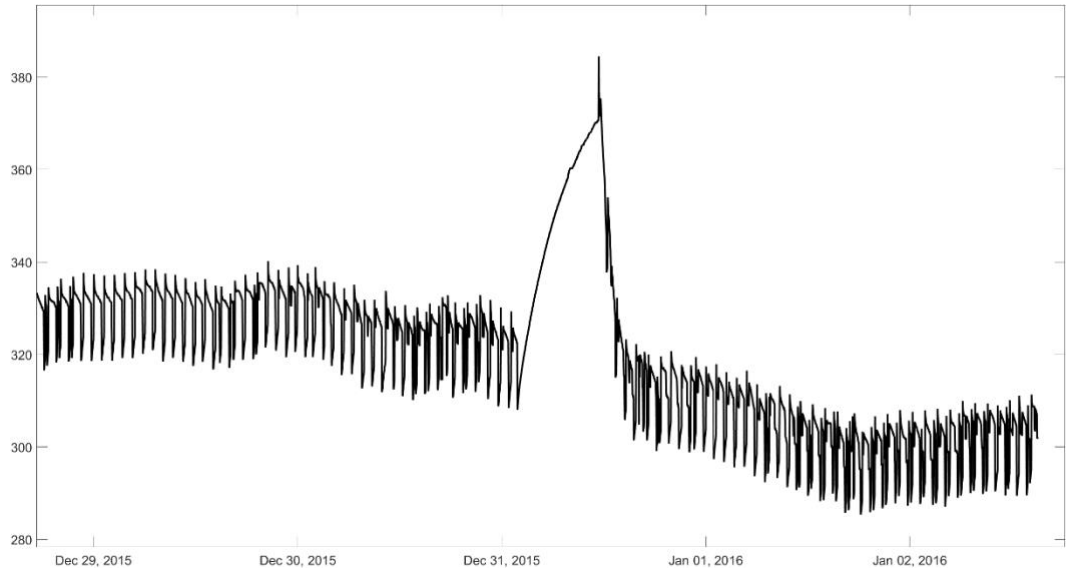

Figure 5-20. Plot of AIT203.

The processing of this data results in 3 new features, namely: A smoothed signal, local minima, and local maxima.

1. The data is smoothed by moving mean smoothing.
2. Local minima and local maxima are extracted.

A moving mean with a smoothing factor of 0.03 is used on the data using the following MATLAB command:

```
AIT203smoothed = smoothdata(AIT203,"movmean","SmoothingFactor",0.03);
```

The local minima and maxima points are extracted as before. The resultant signal is shown in Figure 5-21 along with local extrema.

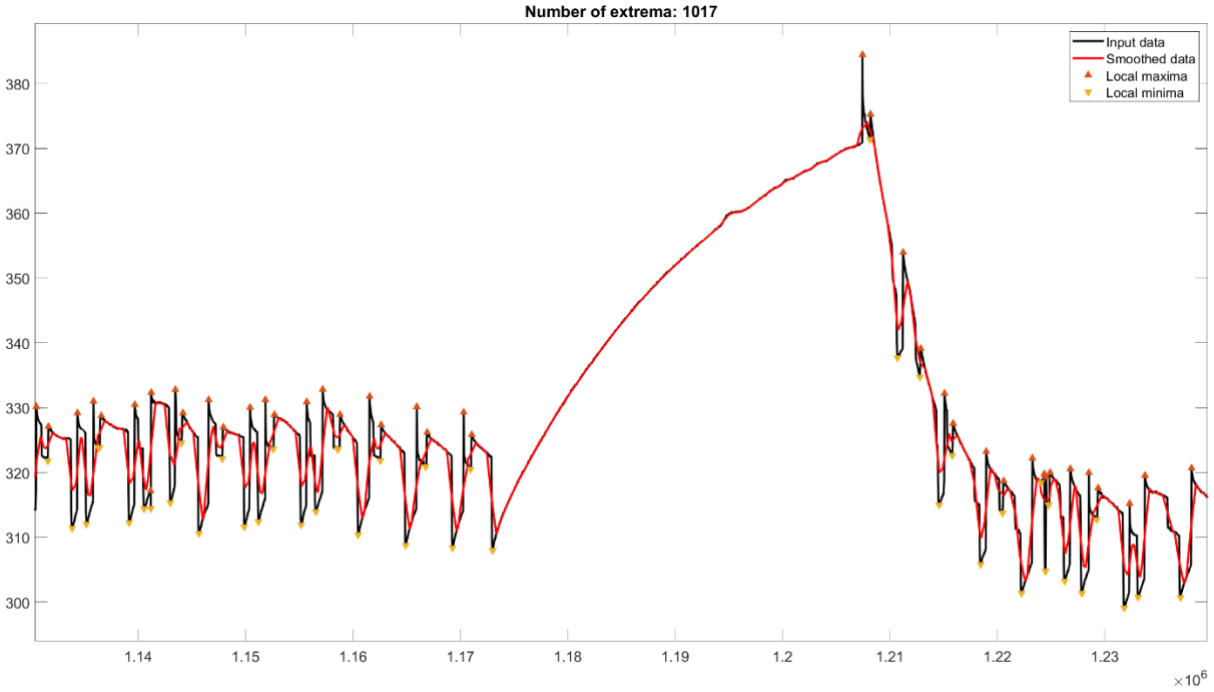

Figure 5-21. Plot showing AIT203 and extracted features.

FIT201 data preprocessing

This sensor is part of stage P2 of the secure water treatment system and is a flow transmitter which controls the chemical dosing pumps. A zoomed in plot of FIT201 data is show in Figure 5-22.

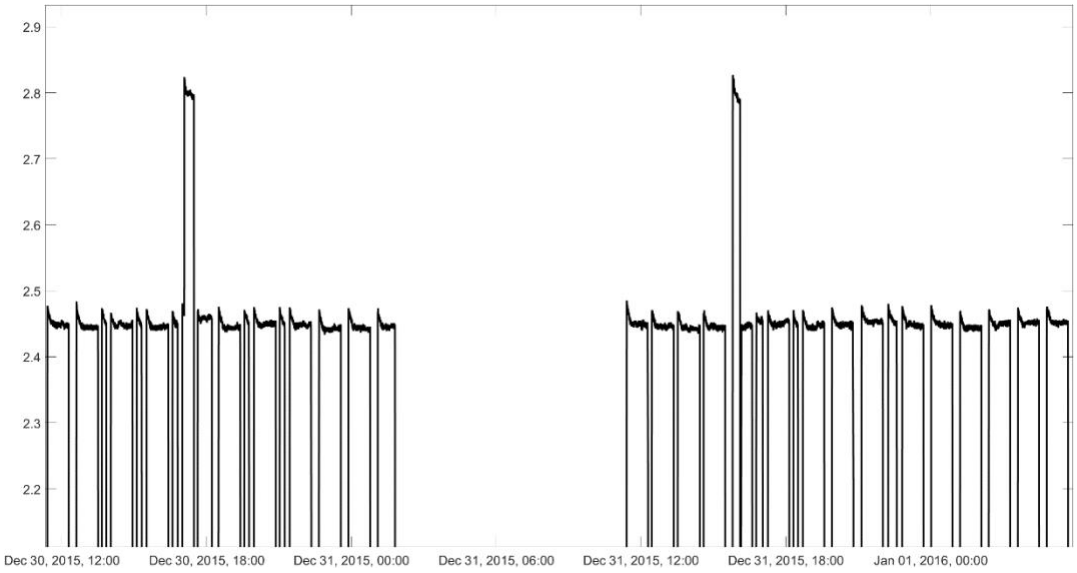

Figure 5-22. Plot of FIT201

The processing of this data results in 2 new features, namely: local maxima, and the time difference between peaks. The code can be found in [89].

1. Local maxima are extracted
2. The time differences between peaks are extracted.

The local extrema are extracted as before using the *islocalmax* built in MATLAB function. The time difference between the peaks is extracted using the code already written for AIT202 processing. The local maxima points in the data are shown in Figure 5-23.

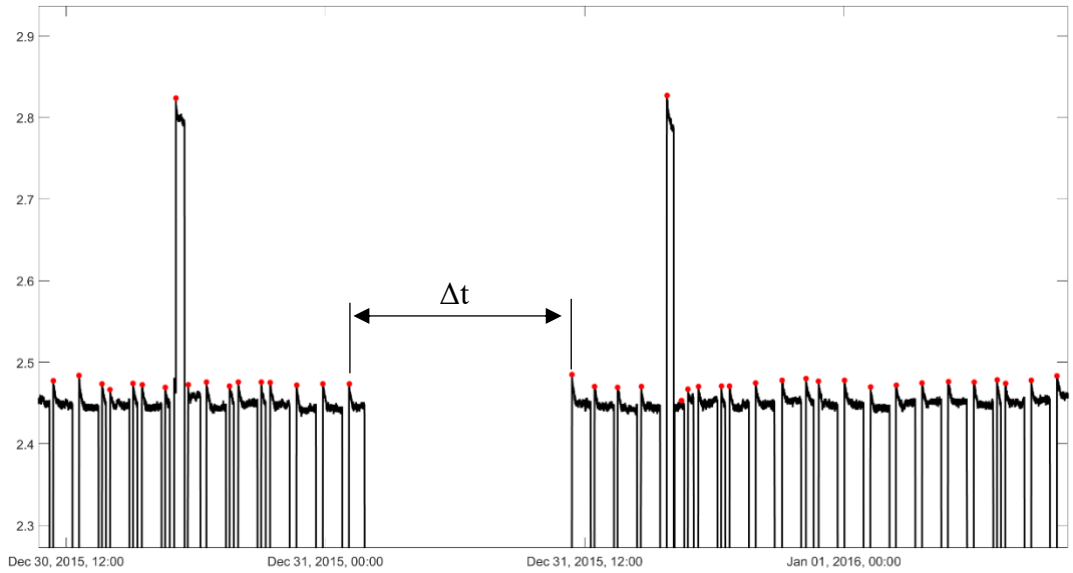

Figure 5-23. Plot of processed FIT201 data showing local maximum extrema.

FIT301 data preprocessing

This sensor is part of stage P3 of the secure water treatment system and is a flow Transmitter which measures the flow of water in the UF stage. A zoomed in plot of FIT301 data is show in Figure 5-24.

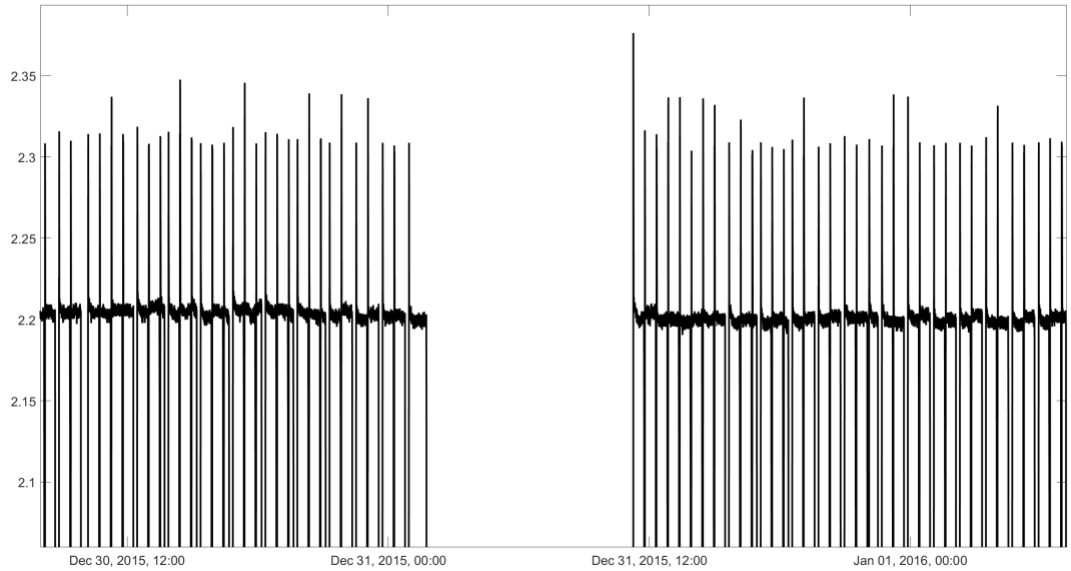

Figure 5-24. Plot of FIT301

The processing of this data results in 3 new features, namely: Smoothed signal, local maxima and the time difference between peaks. The code can be found in [90].

1. The signal is smoothed using a moving mean smoothing technique.
2. Local maxima are extracted.
3. The time differences between peaks are extracted.

An underlying signal is masked by the noise observed in the signal. To remove the noise the data is smoothed with a moving average as done previously. Since this signal is similar to previously processed signal FIT201 the rest of the steps are similar. The local extrema are extracted as before using the *islocalmax* built-in MATLAB function and the time difference between the peaks are also extracted. The resultant signal is shown in Figure 5-25 along with local extrema.

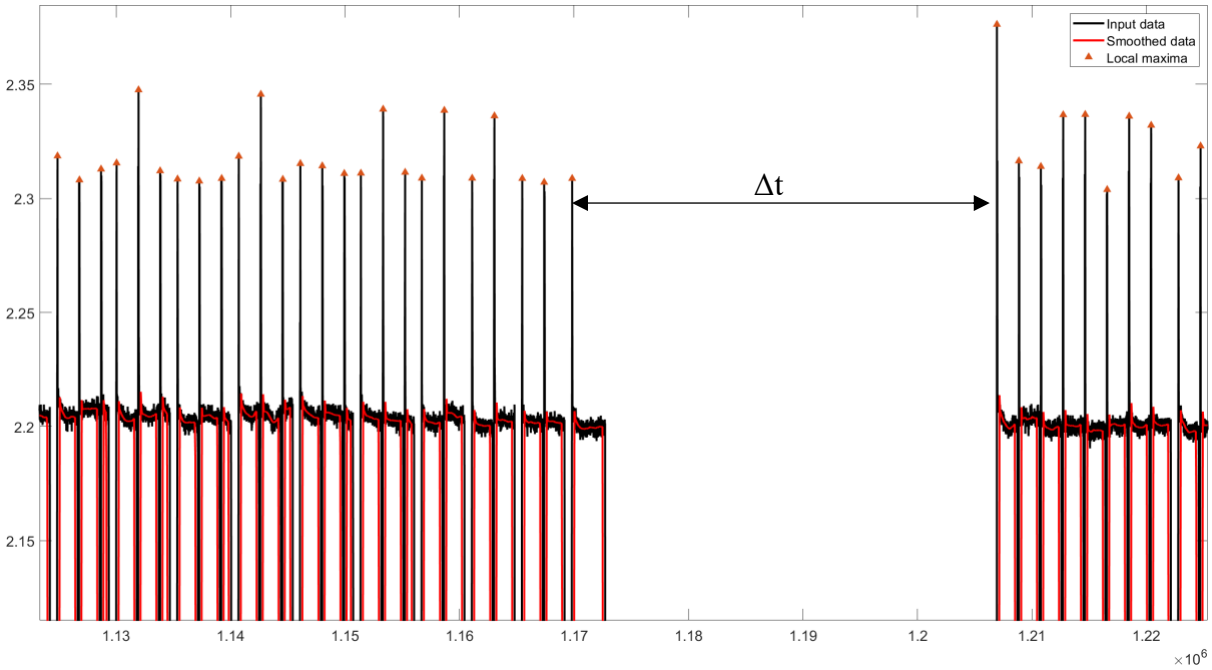

Figure 5-25. Plot showing FIT301 and extracted features.

LIT301 data preprocessing

This sensor is part of stage P3 of the secure water treatment system and is a level Transmitter for the UF feed water tank level. A plot of FIT301 data is show in Figure 5-26.

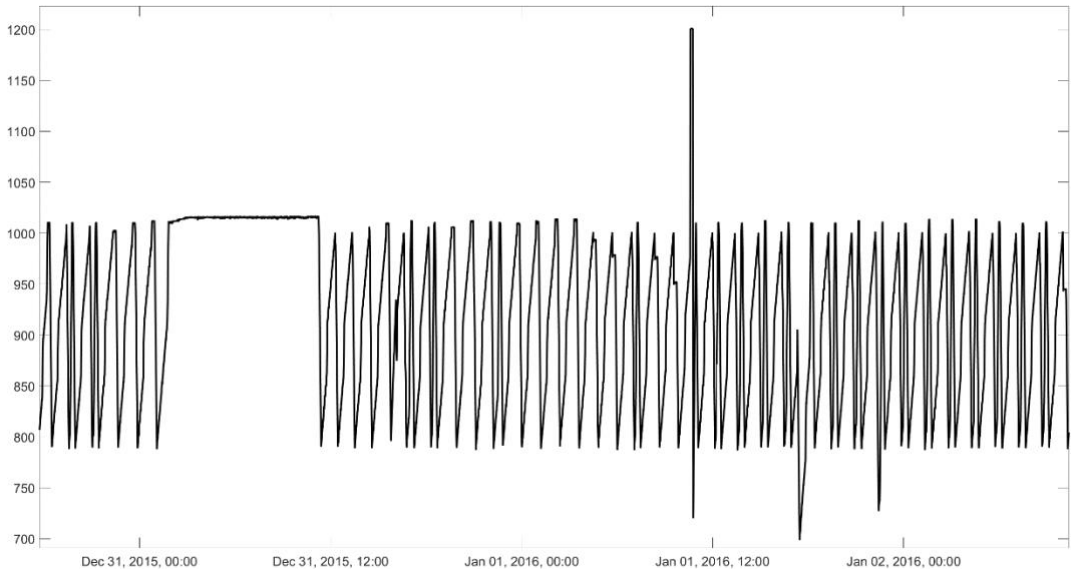

Figure 5-26. Plot of LIT301.

The signal can be defined by its shape, it's peaks and its frequency. The processing of this data results in 3 features, namely: A smoothed signal, local minima and maxima, time difference between peaks and slope values. The code can be found in [86].

1. Raw data is replaced with a smoothed signal. A median filter is chosen for its ability to keep edges.
2. The signal appears to be periodic thus high prominence peaks are be extracted and differences between them populated as a new feature.
3. To learn the shape of the signal, the shape is defined by its slopes, which indicate the rate of change of the signal value (y) over a small time interval (Δt). The slope is calculated as the ratio of the change in y to the change in t. This technique was applied in LIT101.

To derive the slopes the data is first smoothed using a 50th order one-dimensional median filter with a truncate zero-padding method applied. The data is downsampled at a factor of 50. The difference between the resulting data points is computed and then the difference data is upsampled at a factor of 50. The local extrema and time distance between them are extracted as before. Figure 5-27 below shows a plot of LIT301 along with a smoothed signal and its local extrema.

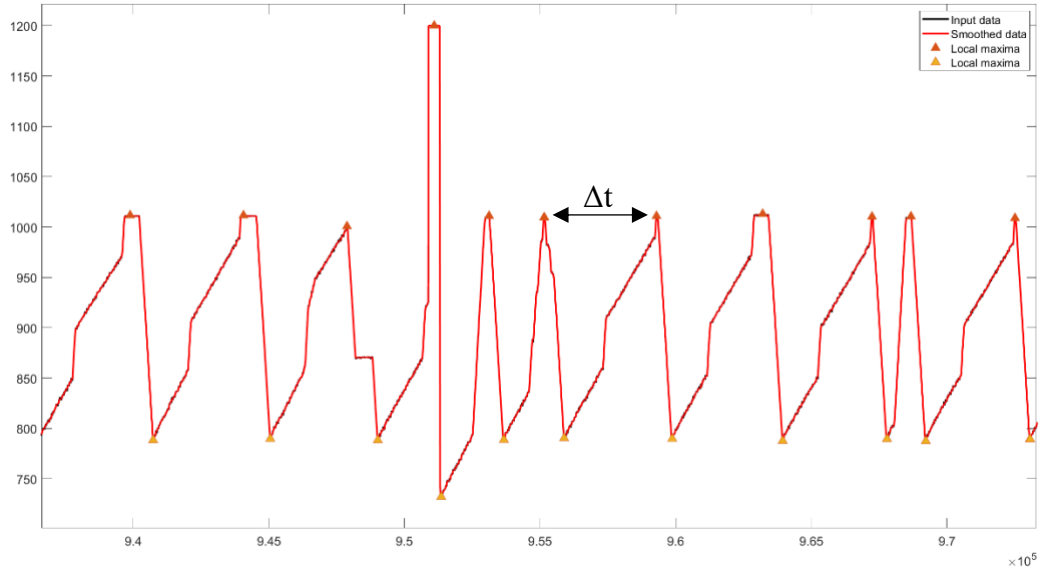

Figure 5-27. Plot showing LIT301 and extracted features.

AIT401 data preprocessing

This sensor is part of stage P4 of the secure water treatment system and it measures the RO hardness in the water. A plot of AIT401 data is shown in Figure 5-28.

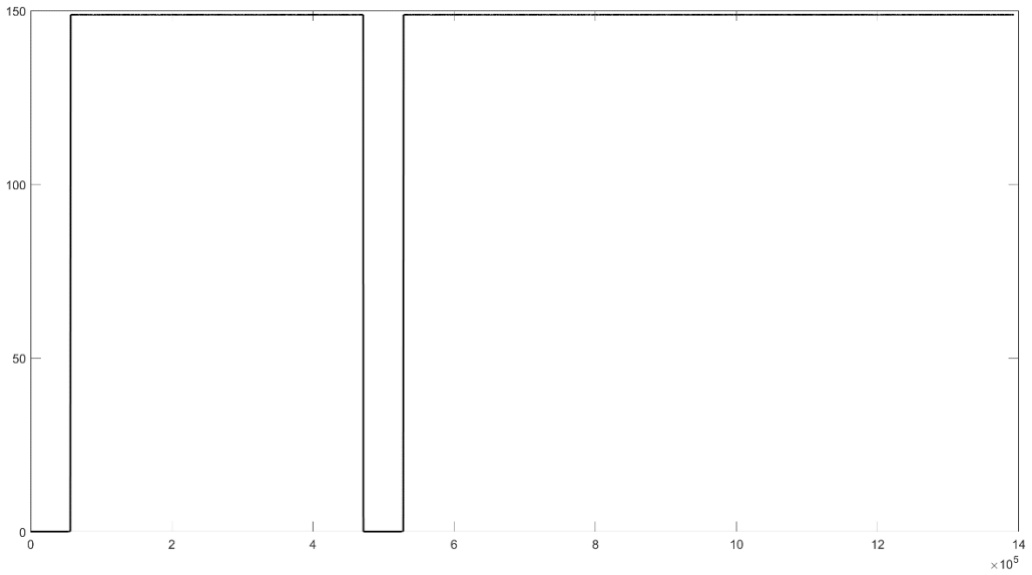

Figure 5-28. Plot of LIT301

The data takes on 2 ranges of values, values close to 0 and values close to 148.8. The date is divided into sections, lows and highs where lows take on the value of 0 and highs take on the value of 1, this new data results in a feature that replaces AIT401 data. The code used for this task is shown in Figure 5-29 below and can be found in [91].

```
min(AIT401) %minimum value
mean(AIT401(1:4e4)) %mean of low values
mean(AIT401(1.4e5:2.8e5)) %mean of high values
max(AIT401) %maximum value
```

```
ans = 0
ans = 0
ans = 148.8044
ans = 148.8561
```

Figure 5-29. AIT402 MATLAB processing code

AIT402 data preprocessing

This sensor is part of stage P4 of the secure water treatment system and is an ORP meter which controls the NaHSO3 dosing by pump P203 and NaOCl dosing by pump P205. The data is noisy and is treated using the moving mean smoothing approach with a smoothing factor of 0.001 to decrease the noise. The resultant smoothed data replaces the raw AIT402 data. Plots of AIT401 and the smoothed data are shown in Figure 5-30 and the code used for this task can be found in [91].

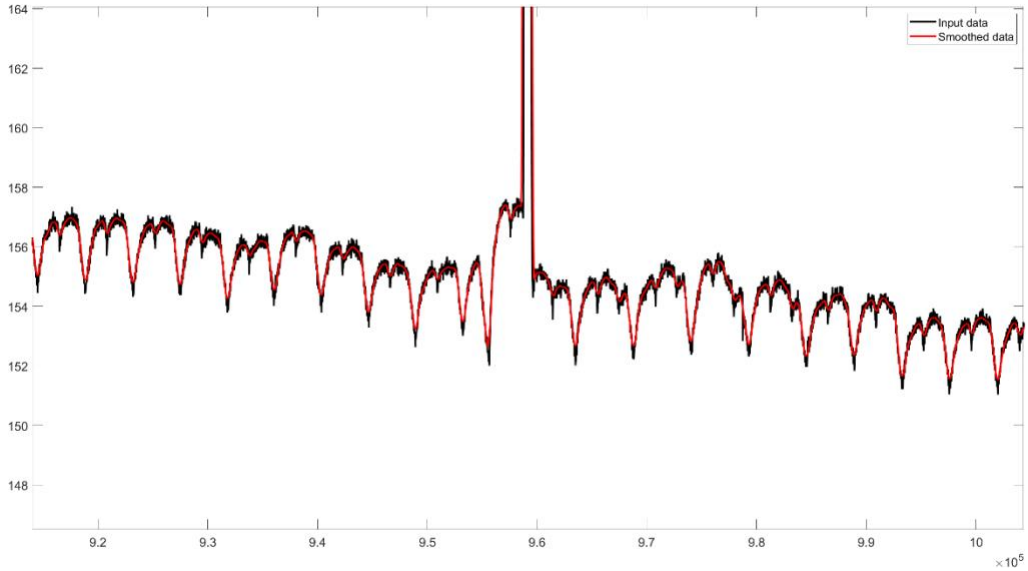

Figure 5-30. Plots of raw and smoothed AIT402 data.

FIT401 data preprocessing

This sensor is part of stage P4 of the secure water treatment system and is a flow Transmitter which controls the UV dechlorinator actuator UV-401. The data is very noisy and so it is processed to reduce the noise using a moving mean smoothing method at a smoothing factor of 0.1. The resultant data is a feature that is used in place of the raw FIT401 data. Plots of FIT401 and the smoothed data are shown in Figure 5-31 and the code used for this task can be found in [92].

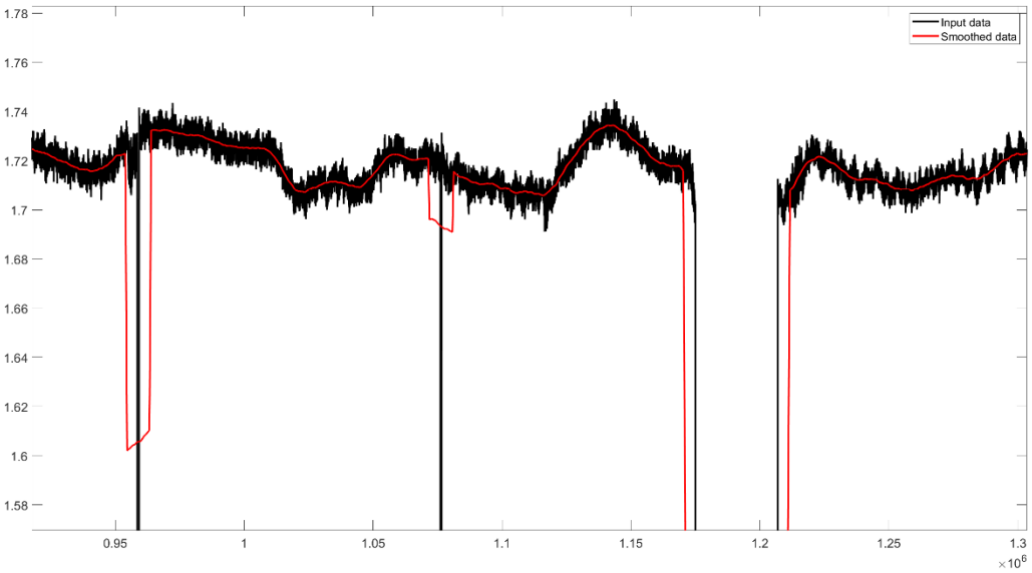

Figure 5-31. Plots of raw and smoothed FIT401 data.

LIT401 data preprocessing

This sensor is part of stage P4 of the secure water treatment system and is a level Transmitter for the RO feed water tank. A close-up plot of LIT401 data is shown in Figure 5-32.

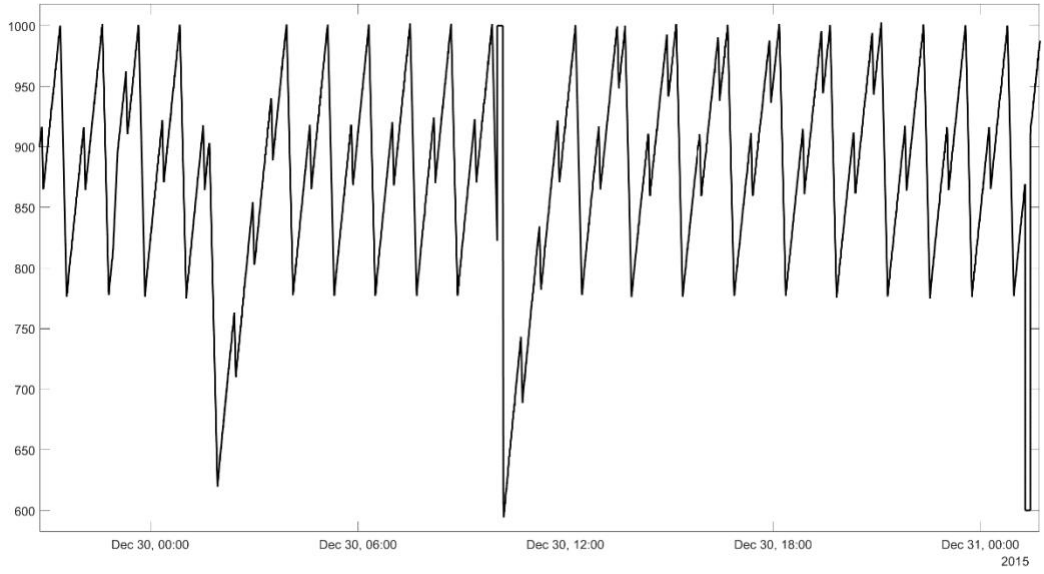

Figure 5-32. Plot of LIT401

As was LIT301, this signal can be defined by its shape, it's peaks and its frequency and is processed in a similar manner to result in 3 features, namely: A smoothed signal, local minima and maxima, time difference between peaks and slope values. Close-up plots of LIT401 with local extrema and the smoothed data are shown in Figure 5-33. The code can be found in [93].

1. The data is smoothed using moving mean smoothing at a low smoothing factor of 0.0005.
2. Since the signal is dominantly periodic the frequency can be approximated by using the time differences between high prominence peaks.
3. To learn the shape of the signal, the shape slopes are extracted by downsampling followed by a difference calculation and then upsampling the data, as was done LIT301.

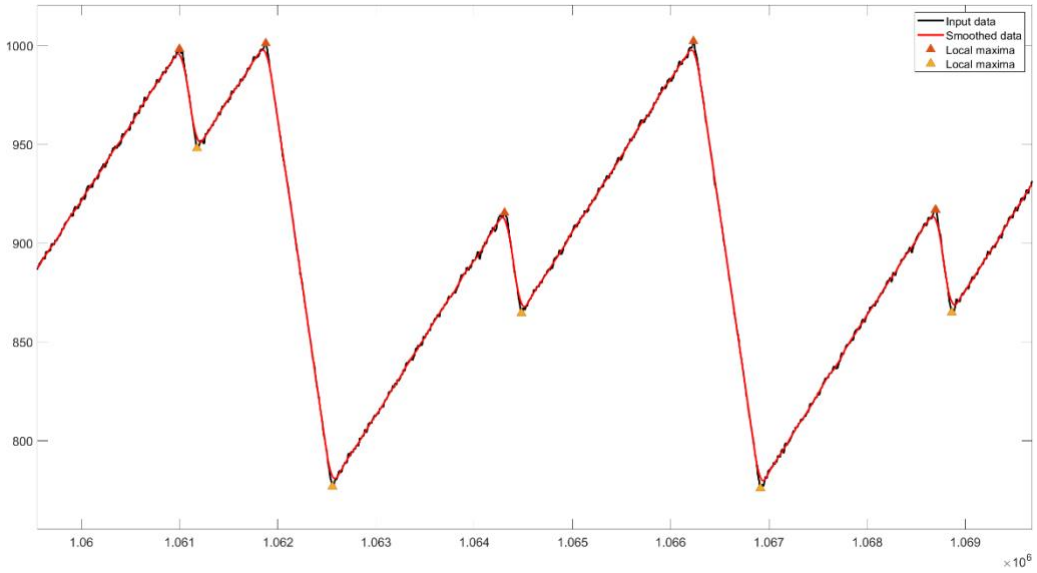

Figure 5-33. Plot showing LIT401 and extracted features.

AIT501 data preprocessing

This sensor is part of stage P5 of the secure water treatment system and is a RO pH analyser which measures HCl level in the water. The data is very noisy and so it is processed to reduce the noise using a moving mean smoothing method at a smoothing factor of 0.02. The resultant data is a feature that is used in place of the raw AIT501 data. Plots of AIT401 and the smoothed data are shown in Figure 5-34 and the code used for this task can be found in [94].

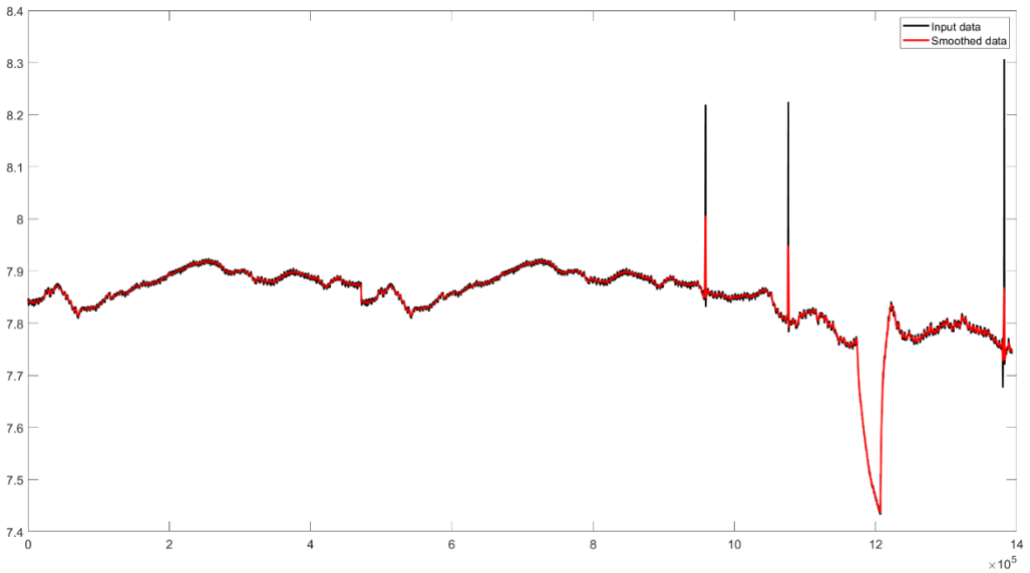

Figure 5-34. Plots of raw and smoothed AIT401 data.

AIT502 data preprocessing

This sensor is part of stage P5 of the secure water treatment system and is a RO feed ORP analyser which measures NaOCl level in the water. Similar to AIT401 processed previously, AIT501 data is also noisy, and the underlying trend is exposed by reducing the noise using a moving mean smoothing method, this time at a smaller smoothing factor of 0.001. Close-up plots of AIT502 and the smoothed data are shown in Figure 5-35 and the code used for this task can be found in [91].

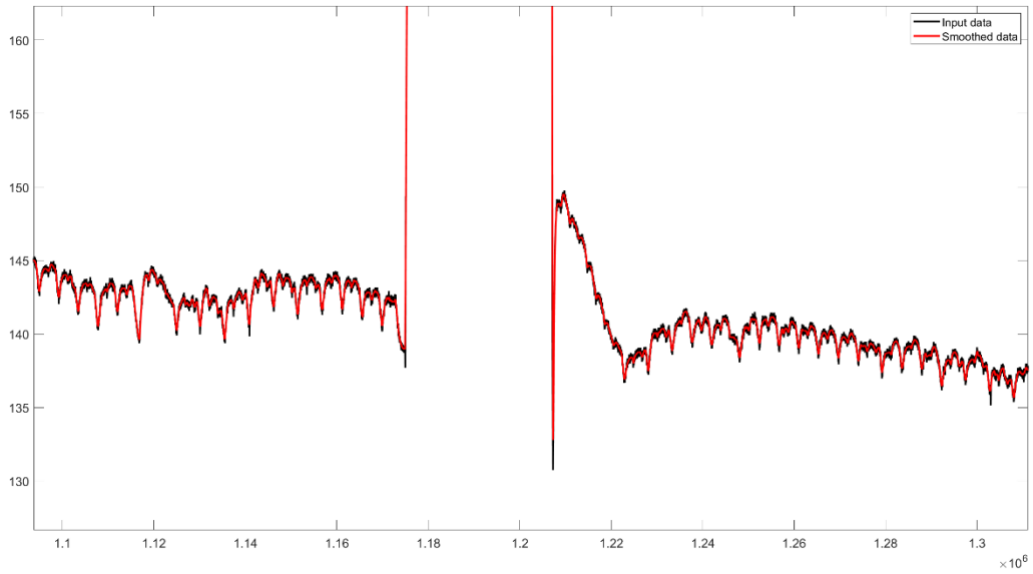

Figure 5-35. Plots of raw and smoothed AIT502 data.

AIT503 data preprocessing

This sensor is part of stage P5 of the secure water treatment system and is a RO feed conductivity analyser which measures NaCl level in the water. As with the other analyser sensors in stage P5 processed previously, AIT503 is a noisy signal, and is processed by reducing the noise using a moving mean smoothing method at a smoothing factor of 0.001. Close-up plots of AIT503 and the smoothed data are shown in Figure 5-36 and the code used for this task can be found [95].

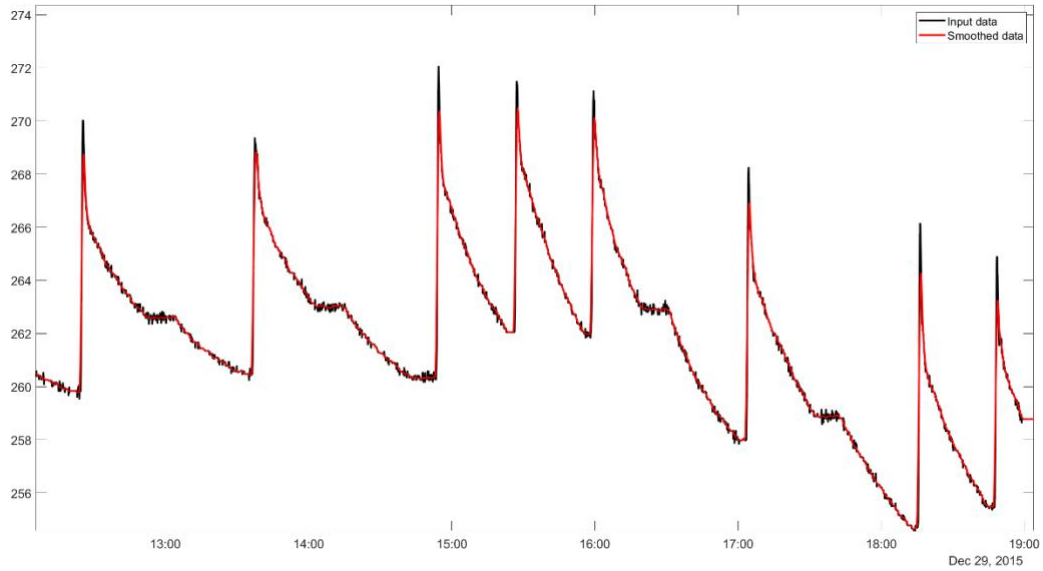

Figure 5-36. Plots of raw and smoothed AIT503 data.

AIT504 data preprocessing

This sensor is part of stage P5 of the secure water treatment system and is a RO permeate conductivity analyser which measures NaCl level in the water. The data is processed to reduce noise using a moving mean smoothing method at a smoothing factor of 0.05. Close-up plots of AIT504 and the smoothed data are shown in Figure 5-37 and the code used for this task can be found in [96].

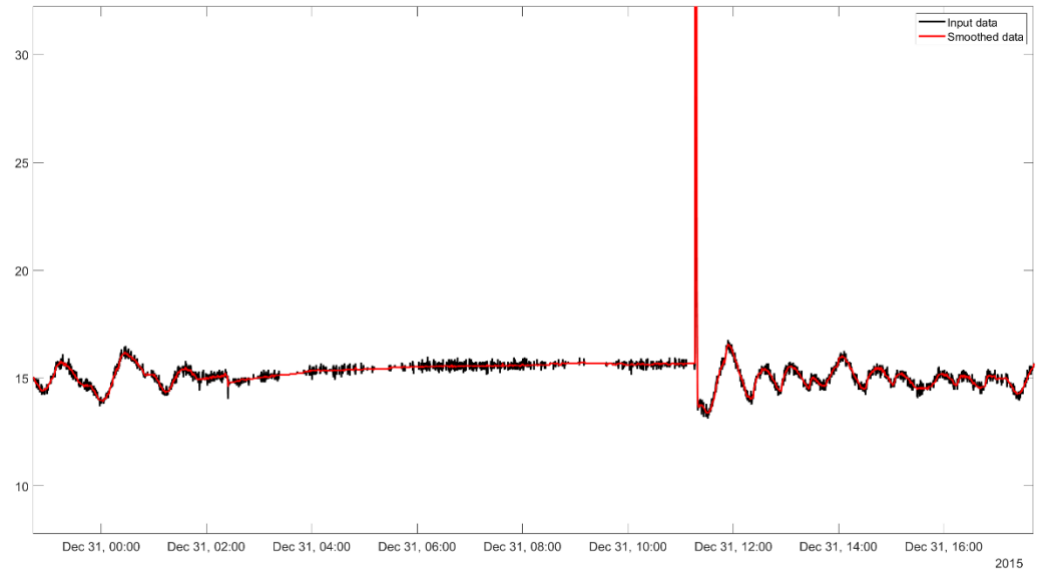

Figure 5-37. Plots of raw and smoothed AIT504 data.

PIT502 data preprocessing

This sensor is part of stage P5 of the secure water treatment system and is a pressure meter which measures RO permeate pressure. A plot of PIT502 data is show in Figure 5-38 and is a very noisy signal, it too is processed to reduce the noise using a moving mean smoothing method at a smoothing factor of 0.03. Figures 5-38 and 5-39 illustrate the level of noise reduced from the signal. The code can be found in [97].

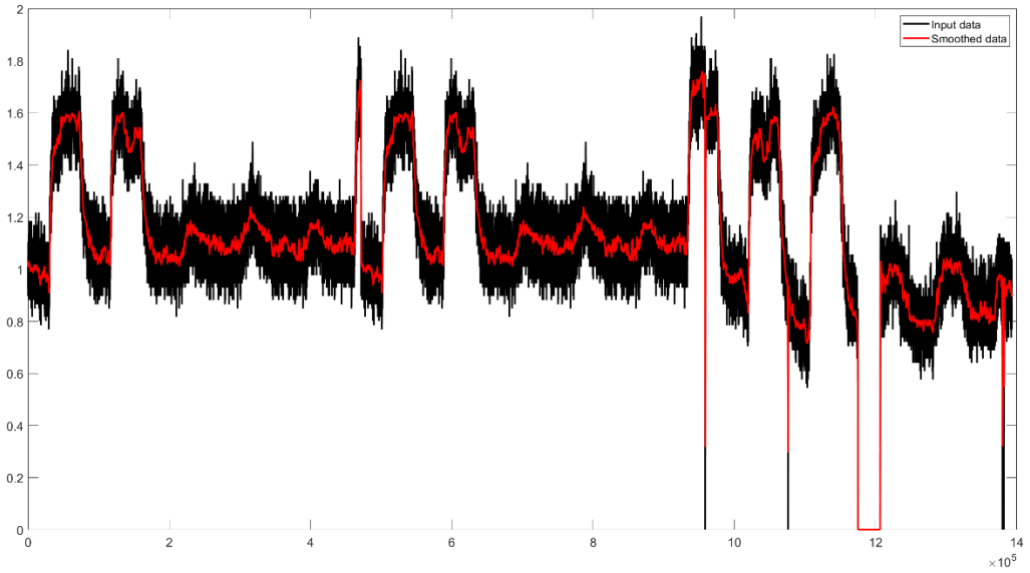

Figure 5-38. Full plots of raw and smoothed PIT502 data.

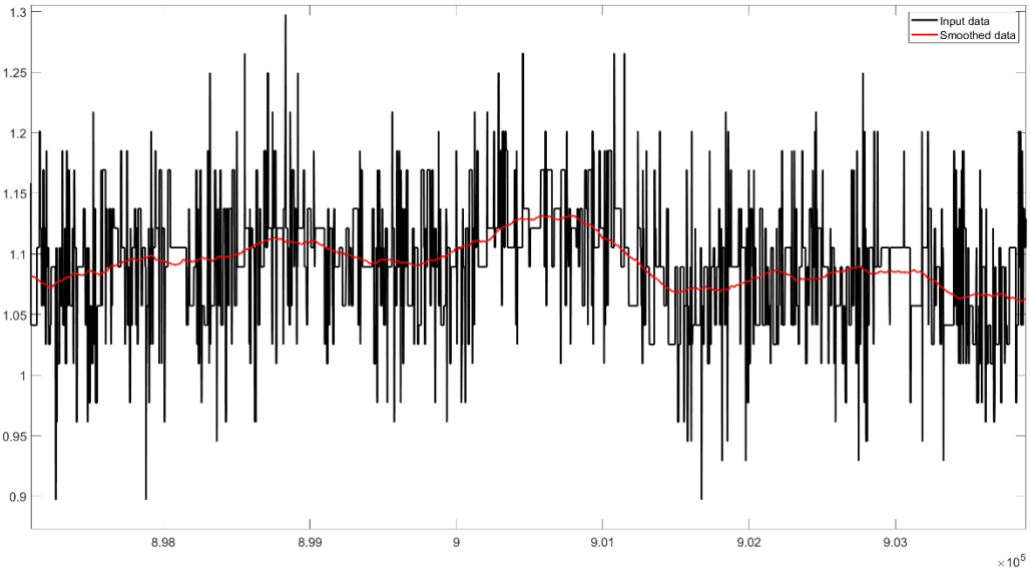

Figure 5-39.Close-up plots of raw and smoothed PIT502 data.

FIT601 data preprocessing

This sensor is part of stage P6 of the secure water treatment system and is a Flow meter for UF Backwash flow. This data required no additional processing is used in the ML algorithm training as is. A close-up plot of FIT601 data is show in Figure 5-40.

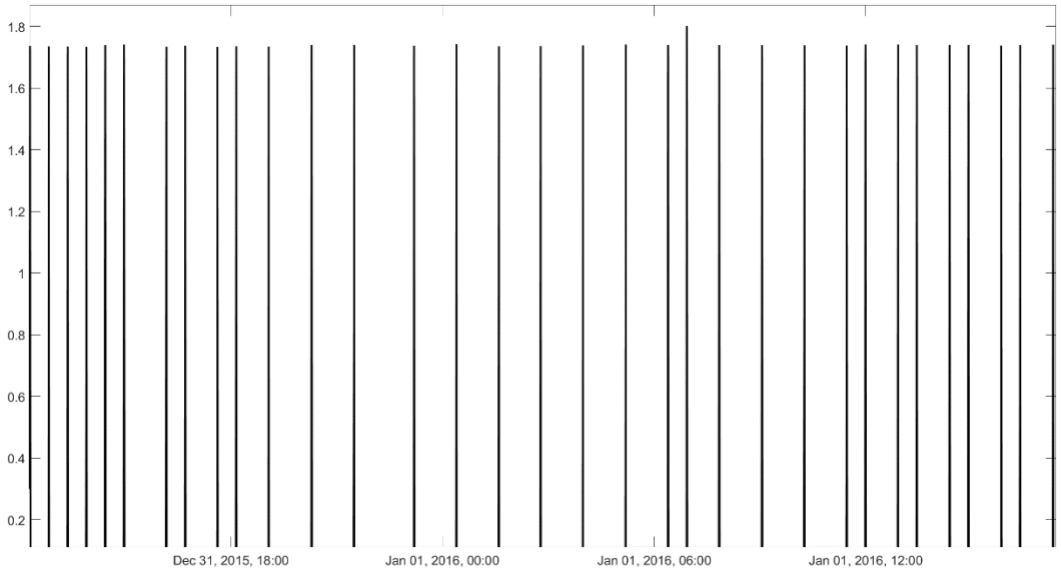

Figure 5-40.Close-up plot of FIT601 data.

Resultant Dataset

The feature extraction process resulted in a much bigger set than the initial. The dataset is further reduced later in the paper using PCA dimensionality reduction technique. At this stage the increased dimensionality is meant to increase the informativeness of the dataset and thus improving the learning ability of the ML algorithm trained on it, resulting in a high classification performance. The resultant dataset is summarised in Table 5-9.

Table 5-9. Dataset variables at this stage.

Stage	Original variable	Extracted features
	Timestamp (unused)	Weekday, Hour
P1: Raw water intake	FIT101	FIT101Filtered, FIT101Peaks, FIT101Frequencies
	LIT101	LIT101Filtered, LIT101Slopes, LIT101Range, LIT101Peaks
	P101	
P2: Chemical dosing	AIT201	AIT201Filtered
	AIT202	AIT202SmoothDetrend, AIT202Minima, AIT202Maxima, AIT202PeakRange
	AIT203	AIT203Smoothed, AIT203Minima, AIT203Maxima
	FIT201	FIT201Maxima, FIT201Range
	P201	
P3: Ultrafiltration	FIT301	FIT301Smoothed, FIT301Maxima, FIT301Range
	LIT301	LIT301Smoothed, LIT301Maxima, LIT301Minima, LIT301Range, LIT301Slopes
	MV301	
	MV303	
	MV304	
P4: Dechlorination	AIT401	AIT401Processed
	AIT402	AIT402Smoothed
	FIT401	FIT401Smoothed
	LIT401	LIT401Smoothed, LIT401Maxima, LIT401Minima, LIT401Range, LIT401Slopes
	P403	
P5: Reverse Osmosis	AIT501	AIT501Smoothed
	AIT502	AIT502Smoothed
	AIT503	AIT503Smoothed
	AIT504	AIT504Smoothed
	P501	
	PIT502	PIT502Smoothed
P6: Reverse Osmosis permeate transfer and ultrafiltration backwash	FIT601	
	P602	
	26	40
Total	**66**	

5.3 Summary

In this section, a systematic process for preparing critical water system infrastructure data to detect intrusion attacks using machine learning was examined. The process commenced with the exploration of the problem, wherein patterns, trends, and relationships in the data were identified, leading to an understanding of redundancies in the features which could pose issues of overfitting or increased variance. This information was then utilized in the Problem Formulation stage to formulate a specific hypothesis regarding the anomalies being detected and determined that supervised learning was required. Baseline models were introduced to set a performance target and evaluate the efficacy of various machine learning algorithms and models, serving as a starting point for future improvement. Preprocessing followed, which involved filling in any missing values using methods such as mean imputation or k-NN based imputation if necessary, removing outliers, and reducing noise through linear filters. Feature extraction was performed to create a set of informative features for training a machine learning model, these included timestamp, spectral, and unique features. The data was then normalized through z-score normalization, dimensionality reduction through PCA could be carried out if necessary and this is discussed. The workflow described is depicted in Figure 12 and may be useful for researchers working with similar data.

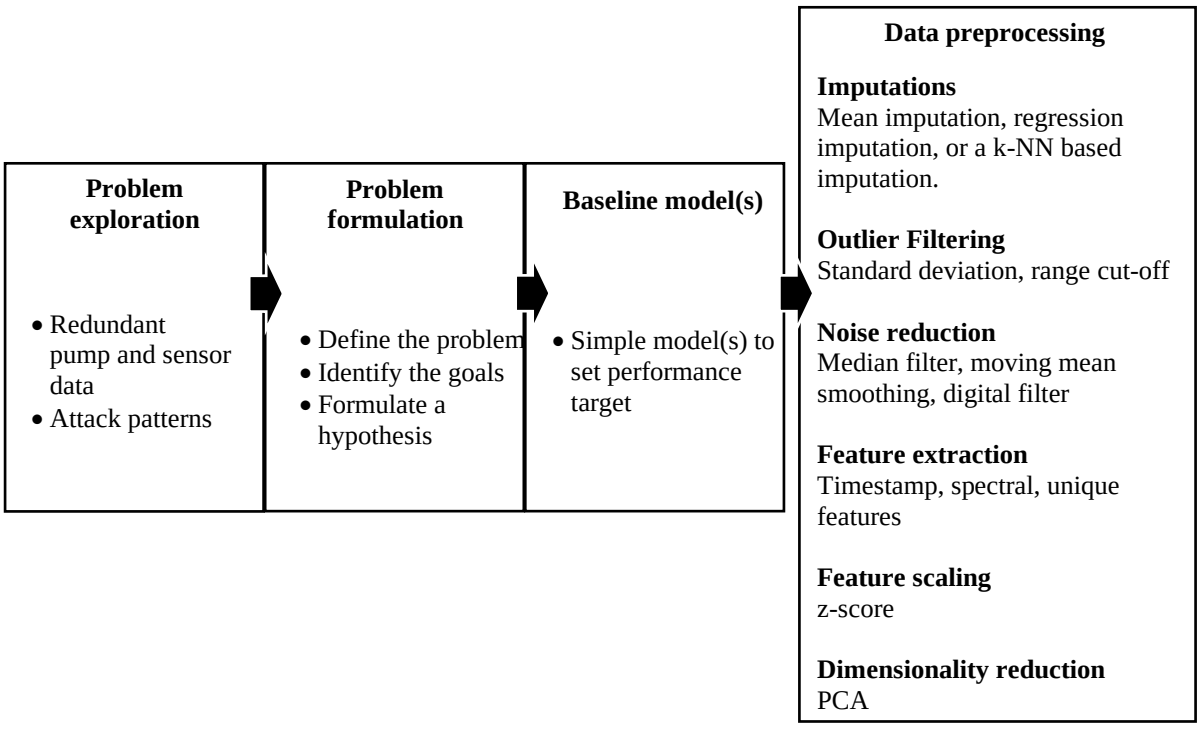

Figure 5-41. Workflow for treating process data from industrial control systems.

Chapter 6. Experiments and results

The work presented in this dissertation aims to provide ideal critical water systems infrastructure specific data preprocessing techniques for a resultant informative dataset that will yield high results when applied on machine learning (ML) classification models. The output of this study is a data preprocessing framework that can be applied by researchers working with similar data. To provide a good framework, an investigative experimental process is applied to answer research questions.

This section applies the resultant dataset from the previous section on ML models through a set of experiments designed to determine the impact that the data preprocessing strategies used have on the evaluation criteria. Just as was done the baseline models, the models in the individual experiments are evaluated on test accuracy, F1 score which makes use of precision and recall, and time to detection. In the results tables of the experiments, green is used to indicate better performance over the baseline while red indicates a worsened performance. Supporting images for the results can be found in appendix A which includes the models' training and testing summary, confusion matrix and the TTD results.

6.1 Baseline models

The trained baseline models described in chapter 5 yielded the results provided in Table 6-1 below. The success of the data preprocessing steps is evaluated on the ability of the experiment models to perform better than the baseline models.

Table 6-1. Baseline results

Model	Accuracy	Precision	Recall	F1 Score	TTD
Tree	95.9%	85.9%	78.4%	82%	2447s (40.8mins)
Ensemble	95.3%	91.7%	66.6%	77.2%	4117s (68.6mins)

6.2 Experiment 1: Models trained on new dataset

Data preprocessing steps in chapter 5 resulted in a larger dataset with increased dimensionality which is meant to increase the informativeness of the dataset and thus improving the performance of the classifier. In this experiment the models are trained on the new dataset and approach for conducting this experiment is depicted in Figure 6-1 where the dataset is first split using a 70-30 time-based splitting method then the training data is then used to train the models and the results are evaluated and documented in Table 6-2.

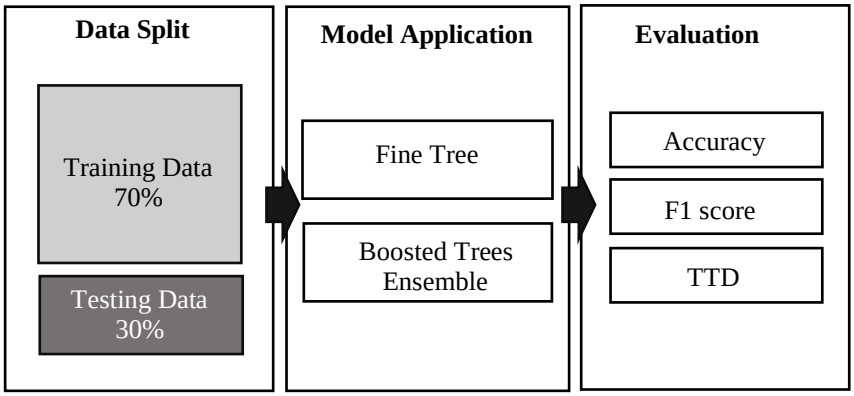

Figure 6-1. Experiment 1 approach

Table 6-2. Experiment 1 results

Model	Accuracy	Precision	Recall	F1 Score	TTD
Tree	96.5%	99.1%	71.8%	83.3%	3932s (65.53mins)
Ensemble	96.5%	99.1%	71.8%	83.3%	3932s (65.53mins)

6.3 Experiment 2: Feature scaling

This experiment is conducted to determine the impact of standardisation on the algorithms' performances. To perform data standardisation the built-in function *zscore* is used, where the input is the data matrix, and the output is the z-scores for each element in the input matrix, which is the standardised data. The command is shown here:

```
standardisedData = zscore(preprocessedData);
```

The approach for conducting this experiment is depicted in Figure 6-1 and describes how the dataset is first standardised, thereafter it follows the same process in experiment 1 described previously. The models and the results are evaluated and documented in Table 6-3.

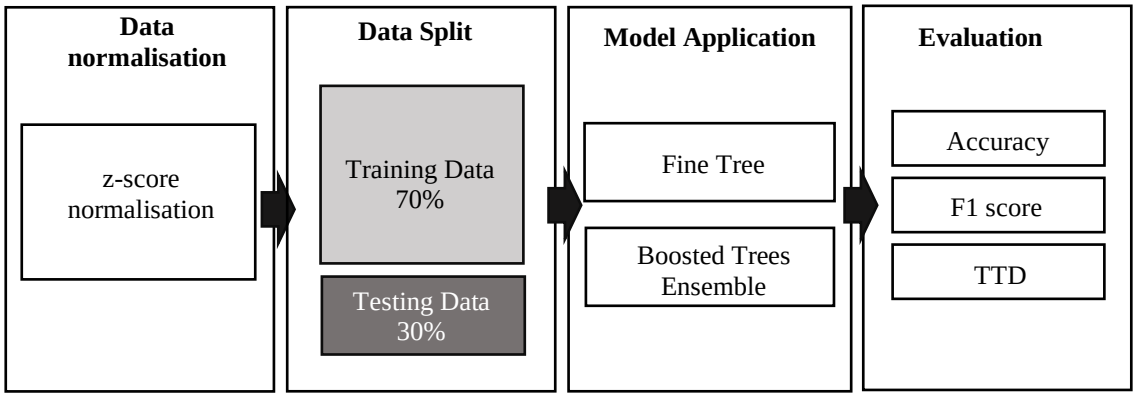

Figure 6-2. Experiment 2 approach

Table 6-3. Experiment 2 results

Model	Accuracy	Precision	Recall	F1 Score	TTD
Tree	96.5%	99.1%	71.8%	83.3%	-7s (-0.1mins)
Ensemble	96.5%	99.1%	71.8%	83.3%	3932s (65.5mins)

6.4 Experiment 3: PCA applied

This experiment aims to determine the effect of PCA. PCA is used to reduce the increased dimensionality that resulted from the data preprocessing steps and this experiment aims to find out if this can be achieved without reducing the performance of the model. The approach for conducting this experiment is depicted in Figure 6-3 where PCA is applied on training data to explain 99% of the variance in the data before it is used to train the models. The trained models are evaluated on testing data with matching features to the training data with reduced dimensionality. The results of this experiment are evaluated and documented in Table 6-4.

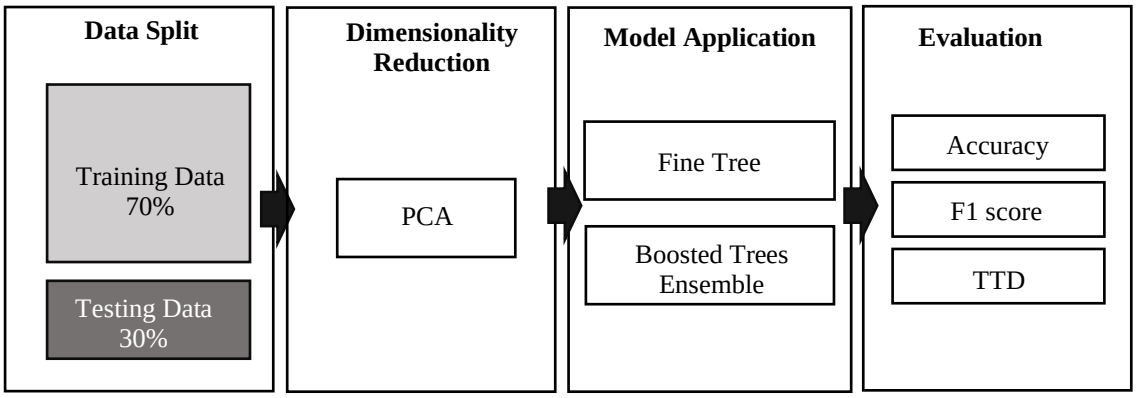

Figure 6-3. Experiment 3 approach

Table 6-4. Experiment 3 results

Model	Accuracy	Precision	Recall	F1 Score	TTD
Tree	86.7%	25.3%	5.6%	9.2%	33705s (561.8mins)
Ensemble	95.5%	93.1%	67.4%	78.2%	3932s (65.5mins)

6.5 Experiment 4: Randomly partitioned data

This experiment aims to determine randomly splitting the data rather than time-based data split used previously. The data is randomly split to 70% as training data and 30% as testing data. This method results in the datasets that do not follow a time sequence and thus evaluating the TTD is not as straight forward anymore. To determine the TTD a second part of the experiment is set up where the attack is isolated from the dataset before it is randomly partitioned and then

added to the test data. The first part of the experiment gives an accurate representation of the accuracy as the same lengths of training and testing data as prior experiments is used to train the models, the second part of the experiment gives a satisfactory time to detection evaluation. The approach of this experiment's is depicted in Figure 6-4 and the results evaluated and documented in Table 6-5 below.

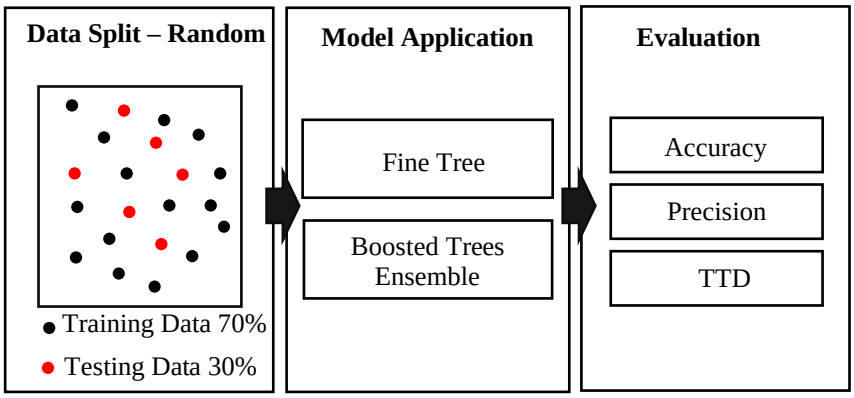

Figure 6-4. Experiment 4 approach

Table 6-5. Experiment 4 results

Model	Accuracy	Precision	Recall	F1 Score	TTD
Tree	96.5%	97.8%	97.8%	97.8%	0s (0mins)
Ensemble	96.5%	95.1%	97.5%	96.3%	0s (0mins)

Chapter 7. Discussion

This section concludes the study by discussing the results and findings. In the methodology baseline models were set up, these models formed a benchmark for comparison with the models applied in the different experiments. The baseline models were trained on unprocessed data. The aim of the study was to preprocess the data so that it yields improved results when applied to ML algorithms compared to the baseline models. The results of the 4 experiments discussed previously are discussed here.

7.1 Experiment 1: Models trained on new dataset

Feature extraction resulted in a larger dataset, and the question is whether the new dataset will result in improved performance of the models or not, and his experiment aimed to answer this question. This resulted in an improved test accuracy and F1 score for both models. This is a good result considering that it can still be improved using hyperparameters. The Fine Tree algorithm's TTD did however worsen, unveiling a trade-off between classification accuracy and time to detection when using the new dataset. This can be attributed to the model being trained on irrelevant features leading to overfitting and resulting in reduced performance, feature selection is to be applied iteratively to determine the key features to train the model on. Regardless of this, the results are satisfactory, and the ML modelling approach requires a careful consideration of classification algorithm to achieve the best results.

7.2 Experiment 2: Feature scaling

This experiment was conducted to determine the impact of data standardisation on the performance of the algorithm. Z-score standardisation was used for this application. The results showed that all 3 models performed better than the baseline models in accuracy, F1 score and TTD. The worsened TTD of the Tree algorithm in experiment 1 was improved in this experiment as the attack is now detected 7 seconds before it happens and this can be interpreted as that the attack was detected immediately, meaning feature scaling improved the performance of the model.

Feature scaling will however not always give better results, in some cases such as this, it yields unchanged results for the Boosted Trees Ensemble algorithm but and improved TTD for the Fine Tree algorithm. The careful choice of ML algorithms is important when feature scaling is to be applied, some algorithms may not be affected at all by scaling, others may be negatively affected, and others may be positively affected. Distance based methods such as k-NN and SVM, as well as deep learning methods such as neural networks are positively affected by

feature scaling [43]. An interesting observation is that feature scaling improved the training time compared to the non-scaled data, for this reason feature scaling should be applied when time is for the essence, especially when computational resources are limited.

7.3 Experiment 3: PCA applied

Dimensionality reduction is an important part of ML modelling and this experiment aimed to determine the effect of PCA on reducing dimensionality while keeping the informativeness of the dataset. The results show that the Tree algorithm performed worse in all evaluation criteria than the baseline while Ensemble performed better in all aspects. The reduced dimensionality meant that the Tree algorithm had less data to learn on than it required while the Ensemble technique proved to be more robust than a single tree as the literature review suggested.

7.4 Experiment 4: Random partitioned data

In previous experiments data had been split using a time-based method, while this experiment applied random splitting of the dataset into 70% and 30% randomly selected training and testing datasets respectively and this yielded the best results of all the experiments. All algorithms had improvements in accuracy, F1 score and TTD.

7.5 What can be learnt from the results?

- Data preprocessing in its entirety is a crucial part of ML modelling as it has a direct effect on the results.
- Feature scaling can have a positive, negative, or neutral effect on the performance of the model and this requires careful judgement on whether it should be used or not.
- While PCA reduces a dataset's number of features by retaining only the important ones, there is still a risk of losing useful information which can affect other algorithms negatively.
- Feature selection can be applied after feature extraction on the dataset that contains both original and new features.

7.6 Limitations and recommendations

This study focused on the data preprocessing stage of ML modelling, only 3 models were applied and hyperparameters were not applied. An extension on this research must include more models evaluated and the use of hyperparameters.

Deep learning algorithms are beyond the scope of the work conducted in this study and were therefore not applied, they bear an advantage of automatic feature extraction and as such in future work it is worth comparing manually extracted features to the automatically extracted features from deep leaning algorithms.

Although the SWaT dataset was crucial to the fulfilment of this study, it bears limitations as the data was collected over a set number of days and constituting a set number of attacks. A virtual model or digital twin of a water treatment system can be created with an inclusion of disturbances and custom attacks that the user can configure. This way future researchers can have access to unlimited and unique data.

Chapter 8. Conclusion

This study aimed to improve research in cyber security for cyber-physical systems in critical water system infrastructure by providing a preprocessing workflow and techniques that can be applied to similar data collected from a similar environment for the purpose of intrusion detection. The study focused on data collected from common process sensors and actuators in water treatment systems, such as flow, level, chemical analyser sensors, and pump actuators.

The study utilized a dataset collected under ideal conditions and found that data preprocessing can have a significant impact on the models, even when the data is already of high quality. The study highlights the importance of feature scaling in data preprocessing, as it can potentially enhance the performance of wide range of machine learning algorithms including distance-based methods and neural networks, and therefore should be taken into consideration when conducting data preprocessing. Having features that are on a similar scale can help to speed up the training time of these methods, making feature scaling a critical aspect to consider in the preprocessing stage.

The study also highlights the role of Principal Component Analysis (PCA) in data preprocessing, but notes that it may not always have a positive effect. PCA is often considered a useful tool for reducing the dimensionality of a dataset, especially when the variables in the data are strongly correlated. However, when the correlation between variables is weak, the results of PCA may be less effective as it becomes difficult to identify patterns effectively. Additionally, applying PCA and discarding data may result in loss of information from the original dataset, potentially reducing its overall usefulness.

It was found that training models on a randomly split dataset yields the best results, but also presents a challenge in accurately evaluating the model's performance. The study suggests that future research should focus on identifying a suitable model for critical water system infrastructure data, as well as exploring evaluation techniques for data preprocessing, machine learning modelling, and algorithm evaluation.

In closing, this study provides valuable insights and techniques for novice and expert researchers in cyber security research in critical water system infrastructure. The data preprocessing workflow and techniques outlined in the study can be applied to similar data collected from similar environments to improve the accuracy and effectiveness of machine learning modelling in cyber security research.

References

[1] D. T. Ramotsoela, G. P. Hancke, and A. M. Abu-Mahfouz, “Attack detection in water distribution systems using machine learning,” *Human-centric Comput. Inf. Sci.*, vol. 9, no. 1, 2019, doi: 10.1186/s13673-019-0175-8.

[2] C. Talcott, “Cyber-physical systems and events,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5380 LNCS, pp. 101–115, 2008, doi: 10.1007/978-3-540-89437-7_6.

[3] A. Humayed, J. Lin, F. Li, and B. Luo, “Cyber-Physical Systems Security - A Survey,” *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, 2017, doi: 10.1109/JIOT.2017.2703172.

[4] G. N. Ericsson, “Cyber security and power system communicationessential parts of a smart grid infrastructure,” *IEEE Trans. Power Deliv.*, vol. 25, no. 3, pp. 1501–1507, 2010, doi: 10.1109/TPWRD.2010.2046654.

[5] Y. Zhang, W. Lee, and Y. A. Huang, “Intrusion detection techniques for mobile wireless networks,” *Wirel. Networks*, vol. 9, no. 5, pp. 545–556, 2003, doi: 10.1023/A:1024600519144.

[6] C. Pfleeger, *Security in computing, 1997*, 5th ed. 1997.

[7] R. Taormina *et al.*, “Battle of the Attack Detection Algorithms: Disclosing Cyber Attacks on Water Distribution Networks,” *J. Water Resour. Plan. Manag.*, vol. 144, no. 8, p. 04018048, 2018, doi: 10.1061/(asce)wr.1943-5452.0000969.

[8] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, “A dataset to support research in the design of secure water treatment systems,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10242 LNCS, no. August, pp. 88–99, 2017, doi: 10.1007/978-3-319-71368-7_8.

[9] I. F. Akyildiz and M. Can Vuran, “Wireless Sensor Networks,” *Wirel. Sens. Networks*, vol. 16, no. 1, pp. 266–282, 2010, doi: 10.1002/9780470515181.

[10] A. A. Cárdenas, S. Amin, and S. Sastry, “Secure control: Towards survivable cyber-physical systems,” *Proc. - Int. Conf. Distrib. Comput. Syst.*, pp. 495–500, 2008, doi: 10.1109/ICDCS.Workshops.2008.40.

[11] B. Schneier, “Managed security monitoring: Network security for the 21st century,” *Comput. Secur. J.*, vol. 17, no. 2, pp. 1–12, 2001.

[12] B. S.M, “Security Problems in the TCP/IP Protocol Suite,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 19, no. 2, pp. 32–48, 1989.

[13] H. Cao, P. Zhu, X. Lu, and A. Gurtov, “A layered encryption mechanism for networked critical infrastructures,” *IEEE Netw.*, vol. 27, no. 1, pp. 12–18, 2013, doi: 10.1109/MNET.2013.6423186.

[14] B. Zhu, A. Joseph, and S. Sastry, “A taxonomy of cyber attacks on SCADA systems,” *Proc. - 2011 IEEE Int. Conf. Internet Things Cyber, Phys. Soc. Comput. iThings/CPSCom 2011*, pp. 380–388, 2011, doi: 10.1109/iThings/CPSCom.2011.34.

[15] A. Zimek and P. Filzmoser, “There and back again: Outlier detection between statistical reasoning and data mining algorithms,” *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 6, 2018, doi: 10.1002/widm.1280.

[16] E. M. Knorr, R. T. Ng, and V. Tucakov, “Distance-based outliers: Algorithms and applications,” *VLDB J.*, vol. 8, no. 3–4, pp. 237–253, 2000, doi: 10.1007/s007780050006.

[17] M. Breunig, H.-P. Kriegel, and J. Sander, “LOF: Identifying Density-Based Local Outliers,” *ACM SIGMOD Rec.*, 2000, doi: 10.1145/335191.

[18] C. C. Aggarwal, “Data classification: Algorithms and applications,” *Data Classification: Algorithms and Applications*. pp. 1–675, 2014, doi: 10.1201/b17320.

[19] J. Singh, S. P. Mohanty, and D. K. Pradhan, *Introduction to SRAM*. 2013.

[20] N. Ben Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes vs decision trees in intrusion detection systems,” *Proc. ACM Symp. Appl. Comput.*, vol. 1, no. January 2004, pp. 420–424, 2004, doi: 10.1145/967900.967989.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, no. 2. Stanford, California: Springer, 2008.

[22] B. Subba, S. Biswas, and S. Karmakar, “Intrusion Detection Systems using Linear Discriminant Analysis and Logistic Regression,” *12th IEEE Int. Conf. Electron. Energy,*

Environ. Commun. Comput. Control (E3-C3), INDICON 2015, no. December, 2016, doi: 10.1109/INDICON.2015.7443533.

[23] M. Labonne, "Anomaly-based network intrusion detection using Maxime Labonne To cite this version : HAL Id : tel-02988296 Anomaly-Based Network Intrusion Detection Using Machine Learning," 2020.

[24] R. S. Society, "The Study of Outliers : Purpose and Model Author (s): Vic Barnett Published by : Wiley for the Royal Statistical Society Stable URL : http://www.jstor.org/stable/2347159 The Study of Outliers : Purpose and Model," vol. 27, no. 3, pp. 242–250, 2018.

[25] H. Lu, Y. Xu, M. Ye, K. Yan, Z. Gao, and Q. Jin, "Learning misclassification costs for imbalanced classification on gene expression data," *BMC Bioinformatics*, vol. 20, no. Suppl 25, pp. 1–10, 2019, doi: 10.1186/s12859-019-3255-x.

[26] X. Y. Liu and Z. H. Zhou, "The influence of class imbalance on cost-sensitive learning: An empirical study," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 970–974, 2006, doi: 10.1109/ICDM.2006.158.

[27] V. Granholm, W. S. Noble, and L. Käll, "A cross-validation scheme for machine learning algorithms in shotgun proteomics.," *BMC Bioinformatics*, vol. 13 Suppl 1, no. Suppl 16, 2012, doi: 10.1186/1471-2105-13-S16-S3.

[28] F. Tempola, R. Rosihan, and R. Adawiyah, "Holdout Validation for Comparison Classfication Naïve Bayes and KNN of Recipient Kartu Indonesia Pintar," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1125, no. 1, p. 012041, 2021, doi: 10.1088/1757-899x/1125/1/012041.

[29] M. Housh and Z. Ohar, "Model-based approach for cyber-physical attack detection in water distribution systems," *Water Res.*, vol. 139, pp. 132–143, 2018, doi: 10.1016/j.watres.2018.03.039.

[30] A. A. Abokifa, K. Haddad, C. Lo, and P. Biswas, "Real-Time Identification of Cyber-Physical Attacks on Water Distribution Systems via Machine Learning–Based Anomaly Detection Techniques," *J. Water Resour. Plan. Manag.*, vol. 145, no. 1, p. 04018089, 2019, doi: 10.1061/(asce)wr.1943-5452.0001023.

[31] S. Alvisi and M. Franchini, “A heuristic procedure for the automatic creation of district metered areas in water distribution systems,” *Urban Water Journal*, vol. 11, no. 2. Taylor & Francis, pp. 137–159, 2014, doi: 10.1080/1573062X.2013.768681.

[32] D. T. Ramotsoela, G. P. Hancke, and A. M. Abu-Mahfouz, “Attack detection in water distribution systems using machine learning,” *Human-centric Comput. Inf. Sci.*, vol. 9, no. 1, 2019, doi: 10.1186/s13673-019-0175-8.

[33] A. Aarabi, F. Wallois, and R. Grebe, “Automated neonatal seizure detection: A multistage classification system through feature selection based on relevance and redundancy analysis,” *Clin. Neurophysiol.*, vol. 117, no. 2, pp. 328–340, 2006, doi: 10.1016/j.clinph.2005.10.006.

[34] A. A. Aburomman and M. Bin Ibne Reaz, “A novel SVM-kNN-PSO ensemble method for intrusion detection system,” *Appl. Soft Comput. J.*, vol. 38, pp. 360–372, 2016, doi: 10.1016/j.asoc.2015.10.011.

[35] L. K. A. I. Hansen and P. Salamon, “LlMb,” vol. 12, no. October, pp. 993–1001, 1990.

[36] Z.-H. Zhou, “Ensemble Learning,” *Encycl. Biometrics*, pp. 270–273, 2009, doi: 10.1007/978-0-387-73003-5_293.

[37] R. E. Schapire, “The Strength of Weak Learnability,” *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990, doi: 10.1023/A:1022648800760.

[38] I. Syarif, E. Zaluska, A. Prugel-Bennett, and G. Wills, “Application of Bagging, Boosting and Stacking to Intrusion Detection,” in *Machine Learning and Data Mining in Pattern Recognition*, 2012, pp. 593–602.

[39] M. Amozegar and K. Khorasani, “An ensemble of dynamic neural network identifiers for fault detection and isolation of gas turbine engines,” *Neural Networks*, vol. 76, pp. 106–121, 2016, doi: 10.1016/j.neunet.2016.01.003.

[40] O. Atilla and E. Hamit, “A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015,” *PeerJ*, pp. 0–21, 2016.

[41] J. B. D. Cabrera, C. Gutiérrez, and R. K. Mehra, “Ensemble methods for anomaly detection and distributed intrusion detection in Mobile Ad-Hoc Networks,” *Inf. Fusion*, vol. 9, no. 1, pp. 96–119, 2008, doi: 10.1016/j.inffus.2007.03.001.

[42] J. Luengo, D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, *Big Data Preprocessing*. 2020.

[43] E. Latyshev, "18th International Conference on Data Analytics and Management in Data-Intensive Domains, DAMDID 2016," *Commun. Comput. Inf. Sci.*, vol. 706, pp. 1–280, 2017.

[44] S. Moritz, A. Sardá, T. Bartz-Beielstein, M. Zaefferer, and J. Stork, "Comparison of different Methods for Univariate Time Series Imputation in R," 2015, [Online]. Available: http://arxiv.org/abs/1510.03924.

[45] S. Bijlsma *et al.*, "Large-scale human metabolomics studies: A strategy for data (pre-) processing and validation," *Anal. Chem.*, vol. 78, no. 2, pp. 567–574, 2006, doi: 10.1021/ac051495j.

[46] T. Schneider and I. M. Held, "Discriminants of twentieth-century changes in earth surface temperatures," *J. Clim.*, vol. 14, no. 3, pp. 249–254, 2001, doi: 10.1175/1520-0442(2001)014<0249:LDOTCC>2.0.CO;2.

[47] 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," *Front. Energy Res.*, vol. 9, no. March, pp. 1–17, 2021, doi: 10.3389/fenrg.2021.652801.

[48] E. A. Felix and S. P. Lee, "Systematic literature review of preprocessing techniques for imbalanced data," *IET Softw.*, vol. 13, no. 6, pp. 479–496, 2019, doi: 10.1049/iet-sen.2018.5193.

[49] W. Li *et al.*, "Outlier detection and removal improves accuracy of machine learning approach to multispectral burn diagnostic imaging," *J. Biomed. Opt.*, vol. 20, no. 12, p. 121305, 2015, doi: 10.1117/1.jbo.20.12.121305.

[50] W. Liu, G. Hua, and J. R. Smith, "Unsupervised One-Class Learning for Automatic Outlier Removal Wei," 2014, doi: 10.1109/CVPR.2014.483.

[51] X. Zhu and X. Wu, "Class Noise vs. Attribute Noise: A Quantitative Study," *Artif. Intell. Rev.*, vol. 22, no. 3, pp. 177–210, 2004, doi: 10.1007/s10462-004-0751-8.

[52] Q. Kang, X. S. Chen, S. S. Li, and M. C. Zhou, "A Noise-Filtered Under-Sampling

Scheme for Imbalanced Classification," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4263–4274, 2017, doi: 10.1109/TCYB.2016.2606104.

[53] S. García, J. Luengo, and F. Herrera, "Feature selection," *Intell. Syst. Ref. Libr.*, vol. 72, no. 6, pp. 163–193, 2015, doi: 10.1007/978-3-319-10247-4_7.

[54] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014, doi: 10.1016/j.compeleceng.2013.11.024.

[55] D. Nettleton, "Selection of Variables and Factor Derivation," *Commer. Data Min.*, pp. 79–104, 2014, doi: 10.1016/b978-0-12-416602-8.00006-6.

[56] A. Jović, K. Brkić, and N. Bogunović, "A review of feature selection methods with applications," *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, no. May 2015, pp. 1200–1205, 2015, doi: 10.1109/MIPRO.2015.7160458.

[57] C. Liu and J. Li, *Feature engineering and computational intelligence in ECG monitoring*. 2020.

[58] Y. Bengio, *Learning Deep Architectures for AI*. Now Publishers Inc, 2009.

[59] A. Rubin, N. Geva, L. Sheintuch, and Y. Ziv, "Hippocampal ensemble dynamics timestamp events in long-term memory," *Elife*, vol. 4, no. DECEMBER2015, pp. 1–16, 2015, doi: 10.7554/eLife.12247.

[60] P. H. Veltink, H. B. J. Bussmann, W. De Vries, W. L. J. Martens, and R. C. Van Lummel, "Detection of static and dynamic activities using uniaxial accelerometers," *IEEE Trans. Rehabil. Eng.*, vol. 4, no. 4, pp. 375–385, 1996, doi: 10.1109/86.547939.

[61] H. Si, Y. Kawahara, H. Kurasawa, H. Morikawa, and T. Aoyama, "A Context-aware Collaborative Filtering Algorithm for Real World Oriented Content Delivery Service," *Ubicomp Metap. Urban Life Work.*, pp. 3–7, 2005.

[62] J. F. James, *A Student's Guide to Fourier Transforms: With Applications in Physics and Engineering*, 3rd ed. Cambridge University Press, 2011.

[63] E. Cheever, "Introduction to the Fourier Transform," *Swarthmore College*, 2022. https://lpsa.swarthmore.edu/Fourier/Xforms/FXformIntro.html.

[64] J. Grus, *Data Science from Scratch*, 2nd ed. O'Reilly Media, Inc., 2019.

[65] M. Ashouri, B. C. M. Fung, F. Haghighat, and H. Yoshino, "Systematic approach to provide building occupants with feedback to reduce energy consumption," *Energy*, vol. 194, p. 116813, 2020, doi: 10.1016/j.energy.2019.116813.

[66] N. M. Nawi, W. H. Atomi, and M. Z. Rehman, "The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks," *Procedia Technol.*, vol. 11, no. Iceei, pp. 32–39, 2013, doi: 10.1016/j.protcy.2013.12.159.

[67] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[68] J. C. Lam, K. K. W. Wan, K. L. Cheung, and L. Yang, "Principal component analysis of electricity use in office buildings," *Energy Build.*, vol. 40, no. 5, pp. 828–836, 2008, doi: 10.1016/j.enbuild.2007.06.001.

[69] L. J. P. Van Der Maaten, E. O. Postma, and H. J. Van Den Herik, "Dimensionality Reduction: A Comparative Review," *J. Mach. Learn. Res.*, vol. 10, pp. 1–41, 2009, doi: 10.1080/13506280444000102.

[70] "Secure Water Treatment - iTrust." https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/.

[71] iTrust Centre for Research in Cyber Security, "Secure Water Treatment (SWaT) Testbed," *iTrust lab.*, no. October, 2018, [Online]. Available: https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/#swat.

[72] M. Stamp, *Information Security: Principles and Practice*, 2nd ed. Wiley, 2011.

[73] S. Adepu and A. Mathur, "An Investigation into the Response of a Water Treatment System to Cyber Attacks," *Proc. IEEE Int. Symp. High Assur. Syst. Eng.*, vol. 2016-March, pp. 141–148, 2016, doi: 10.1109/HASE.2016.14.

[74] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," *Proc. - IEEE Int. Conf. Data Mining, ICDM*, pp. 413–422, 2008, doi: 10.1109/ICDM.2008.17.

[75] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986, doi: 10.1007/bf00116251.

[76] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997, doi: 10.1006/jcss.1997.1504.

[77] L. Rokach, “Ensemble-based classifiers,” *Artif. Intell. Rev.*, vol. 33, no. 1–2, pp. 1–39, 2010, doi: 10.1007/s10462-009-9124-7.

[78] D. E. Birba, “Study of Data Splitting Algorithms for Machine Learning,” 2020.

[79] L. P. Varoslavskiy, *Digital Image Processing.*, vol. 31–32, no. 11. 1977.

[80] A. V Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing (1999, Prentice Hall).pdf*, 2nd ed. New Jersey: Prentice Hall, 1999.

[81] MathWorks, “filter.” https://www.mathworks.com/help/matlab/ref/filter.html.

[82] I. V. Mboweni, “MScEng_Electrical Source Code,” *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical.

[83] S. B. Wankhede, “Anomaly Detection using Machine Learning Techniques,” *2019 IEEE 5th Int. Conf. Converg. Technol. I2CT 2019*, 2019, doi: 10.1109/I2CT45611.2019.9033532.

[84] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, and M. Imran, “Real-time big data processing for anomaly detection: A Survey,” *Int. J. Inf. Manage.*, vol. 45, no. February 2018, pp. 289–307, 2019, doi: 10.1016/j.ijinfomgt.2018.08.006.

[85] I. V. Mboweni, “MScEng_Electrical Source Code - DataPreprocessing/TimestampFeatures.m,” *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/TimestampFeatures.m.

[86] I. V. Mboweni, “MScEng_Electrical Source Code - DataPreprocessing/LIT101processing.m,” *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/LIT101processing.m.

[87] I. V. Mboweni, “MScEng_Electrical Source Code -

DataPreprocessing/FIT101processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/FIT101processing.m.

[88] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/AIT203processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/AIT203processing.m.

[89] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/FIT201processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/FIT201processing.m.

[90] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/FIT301processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/FIT301processing.m.

[91] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/AIT502processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/AIT402processing.m.

[92] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/FIT401processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/FIT401processing.m.

[93] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/LIT401processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/LIT401processing.m.

[94] I. V. Mboweni, "MScEng_Electrical Source Code - DataPreprocessing/AIT501processing.m," *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPrepr

ocessing/AIT501processing.m.

[95] I. V. Mboweni, “MScEng_Electrical Source Code - DataPreprocessing/AIT503processing.m,” *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/AIT503processing.m.

[96] I. V. Mboweni, “MScEng_Electrical Source Code - DataPreprocessing/AIT504processing.m,” *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/AIT504processing.m.

[97] I. V. Mboweni, “MScEng_Electrical Source Code - DataPreprocessing/PIT502processing.m,” *GitHub*, 2022. https://github.com/IgnitiousVukosiMboweni/MScEng_Electrical/blob/main/DataPreprocessing/PIT502processing.m.

Appendix A: Results

Baseline models

Training Results

Accuracy (Validation)	100.0%
Total cost (Validation)	412
Prediction speed	~1600000 obs/sec
Training time	18.714 sec

Test Results

Accuracy (Test)	95.9%
Total cost (Test)	60670

Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Surrogate decision splits: Off

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Training Results

Accuracy (Validation)	100.0%
Total cost (Validation)	175
Prediction speed	~170000 obs/sec
Training time	190.35 sec

Test Results

Accuracy (Test)	95.3%
Total cost (Test)	86811

Model Type
Preset: Boosted Trees
Ensemble method: AdaBoost
Learner type: Decision tree
Maximum number of splits: 20
Number of learners: 30
Learning rate: 0.1

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Figure A-1. Model Summary. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

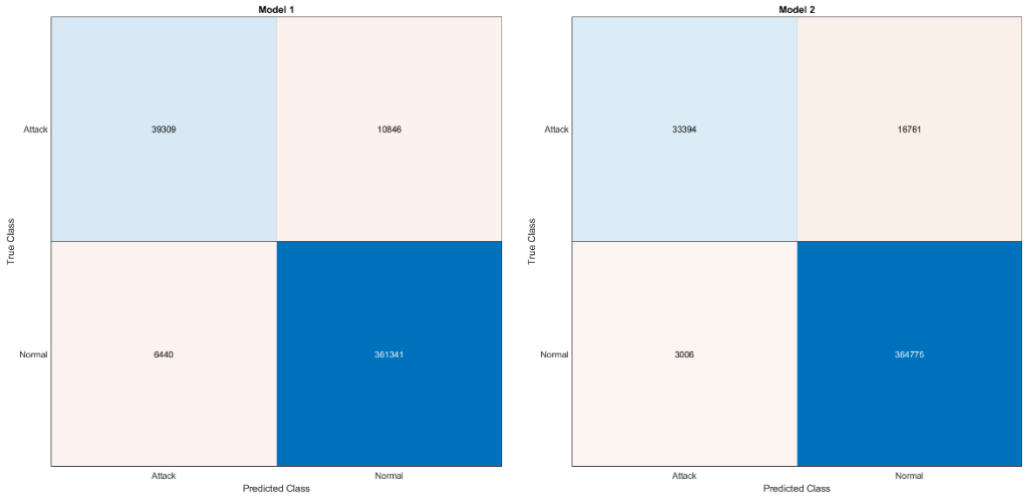

Figure A-2. Classification Matrices. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 2447
```

```
TTDmin = TTDs/60
```

```
TTDmin = 40.7833
```

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 4117
```

```
TTDmin = TTDs/60
```

```
TTDmin = 68.6167
```

Figure A-3. Time to detection. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

Experiment 1: Models trained on new dataset

Training Results
Accuracy (Validation) 100.0%
Total cost (Validation) 92
Prediction speed ~490000 obs/sec
Training time 43.878 sec

Test Results
Accuracy (Test) 96.5%
Total cost (Test) 70964

Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Surrogate decision splits: Off

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Training Results
Accuracy (Validation) 100.0%
Total cost (Validation) 96
Prediction speed ~170000 obs/sec
Training time 638.46 sec

Test Results
Accuracy (Test) 96.5%
Total cost (Test) 70964

Model Type
Preset: Boosted Trees
Ensemble method: AdaBoost
Learner type: Decision tree
Maximum number of splits: 20
Number of learners: 30
Learning rate: 0.1

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Figure A-4. Model Summary. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

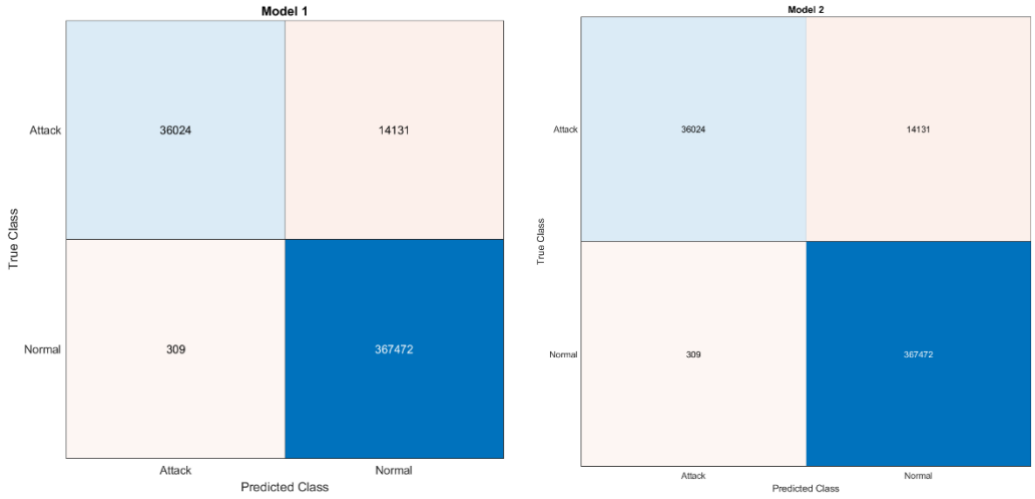

Figure A-5. Classification Matrices. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 3932
```

```
TTDmin = TTDs/60
```

```
TTDmin = 65.5333
```

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 3932
```

```
TTDmin = TTDs/60
```

```
TTDmin = 65.5333
```

Figure A-6. Time to detection. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

Experiment 2: Feature scaled vs original scale

Training Results
Accuracy (Validation) 100.0%
Total cost (Validation) 152
Prediction speed ~830000 obs/sec
Training time 19.779 sec

Test Results
Accuracy (Test) 96.5%
Total cost (Test) 70964

Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Surrogate decision splits: Off

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Training Results
Accuracy (Validation) 100.0%
Total cost (Validation) 74
Prediction speed ~160000 obs/sec
Training time 462.2 sec

Test Results
Accuracy (Test) 96.5%
Total cost (Test) 70964

Model Type
Preset: Boosted Trees
Ensemble method: AdaBoost
Learner type: Decision tree
Maximum number of splits: 20
Number of learners: 30
Learning rate: 0.1

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Figure A-7. Model Summary. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

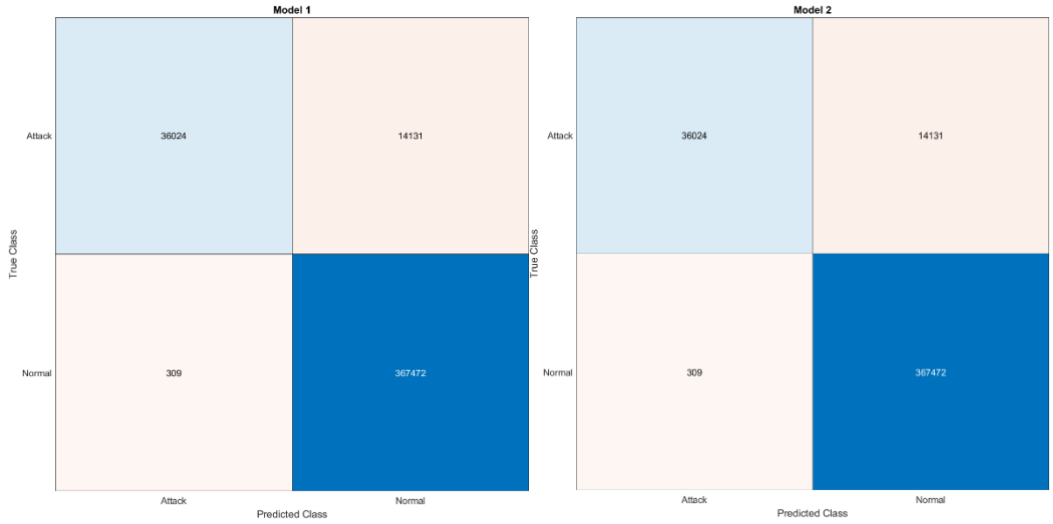

Figure A-8. Classification Matrices. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = -7
```

```
TTDmin = TTDs/60
```

```
TTDmin = -0.1167
```

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = -7
```

```
TTDmin = TTDs/60
```

```
TTDmin = -0.1167
```

Figure A-9. Time to detection. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

Experiment 3: PCA applied

Training Results
Accuracy (Validation) 100.0%
Total cost (Validation) 142 (Models have different cost matrices)
Prediction speed ~360000 obs/sec
Training time 31.303 sec

Test Results
Accuracy (Test) 86.7%
Total cost (Test) 245024 (Models have different cost matrices)

Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Surrogate decision splits: Off

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA is keeping enough components to explain 99% variance.
After training, 7 components were kept.
Explained variance per component (in order): 56.5%, 22.7%, 9.8%, 5.1%, 3.2%, 1.6%, 0.4%, 0.4%, 0.1%, 0.1% (variances of least important components hidden)

Misclassification Costs
Cost matrix: custom

Training Results
Accuracy (Validation) 100.0%
Total cost (Validation) 126 (Models have different cost matrices)
Prediction speed ~130000 obs/sec
Training time 99.798 sec

Test Results
Accuracy (Test) 95.5%
Total cost (Test) 84393 (Models have different cost matrices)

Model Type
Preset: Boosted Trees
Ensemble method: AdaBoost
Learner type: Decision tree
Maximum number of splits: 20
Number of learners: 30
Learning rate: 0.1

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA is keeping enough components to explain 99% variance.
After training, 7 components were kept.
Explained variance per component (in order): 56.5%, 22.7%, 9.8%, 5.1%, 3.2%, 1.6%, 0.4%, 0.4%, 0.1%, 0.1% (variances of least important components hidden)

Misclassification Costs
Cost matrix: custom

Figure A-10. Model Summary. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

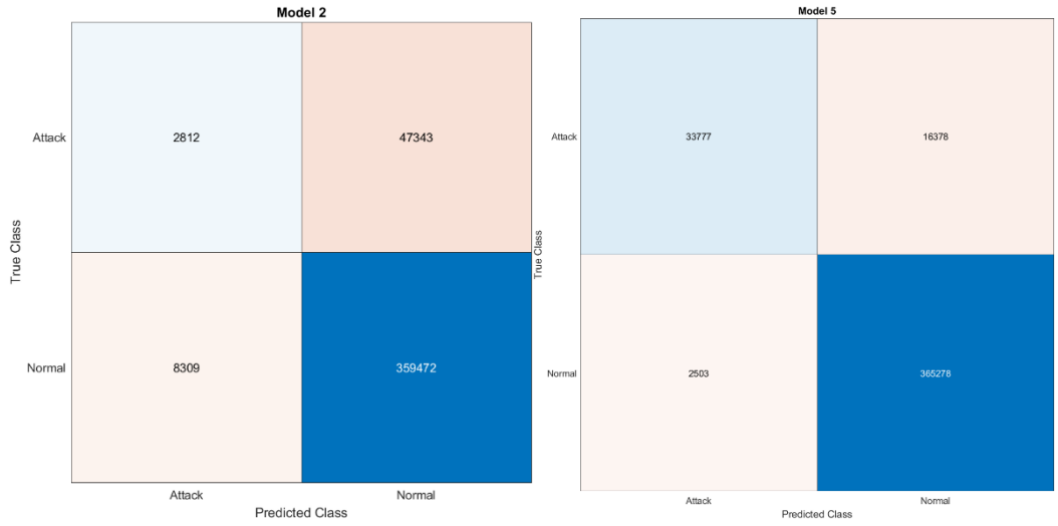

Figure A-11. Classification Matrices. From top left: Fine Tree classifier, Boosted Trees Ensemble classifier

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 33705
```

```
TTDmin = TTDs/60
```

```
TTDmin = 561.7500
```

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 2187
```

```
TTDmin = TTDs/60
```

```
TTDmin = 36.4500
```

Figure A-12. Time to detection. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

Experiment 4: Randomly partitioned data

Training Results
Accuracy (Validation) 99.9%
Total cost (Validation) 4592
Prediction speed ~730000 obs/sec
Training time 198.75 sec

Test Results
Accuracy (Test) 99.8%
Total cost (Test) 2117

Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Surrogate decision splits: Off

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Training Results
Accuracy (Validation) 99.7%
Total cost (Validation) 6163
Prediction speed ~140000 obs/sec
Training time 1602.6 sec

Test Results
Accuracy (Test) 99.7%
Total cost (Test) 2838

Model Type
Preset: Boosted Trees
Ensemble method: AdaBoost
Learner type: Decision tree
Maximum number of splits: 20
Number of learners: 30
Learning rate: 0.1

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

PCA
PCA disabled

Misclassification Costs
Cost matrix: custom

Figure A-13. Model Summary. From top left: Fine Tree classifier, Boosted Trees Ensemble classifier

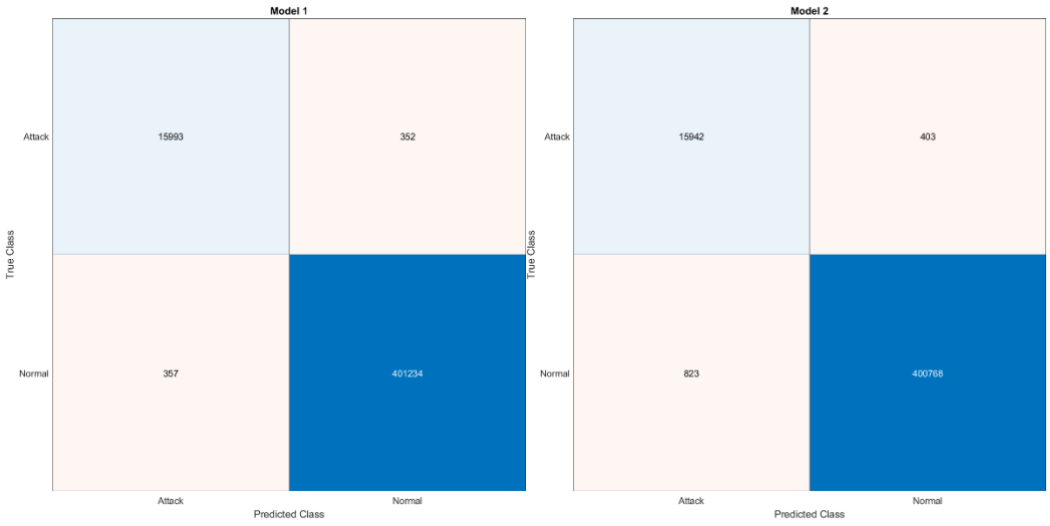

Figure A-14. Classification Matrices. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 0
```

```
TTDmins = TTDs/60
```

```
TTDmins = 0
```

```
%Time to detection
TTD = NormalAttackBinary - yfitBinary;
TTDs = sum(TTD)
```

```
TTDs = 0
```

```
TTDmins = TTDs/60
```

```
TTDmins = 0
```

Figure A-15. Time to detection. From left: Fine Tree classifier, Boosted Trees Ensemble classifier

Appendix B: Dataset reduction by correlation analysis

1. Stage P1: Raw water supply & storage

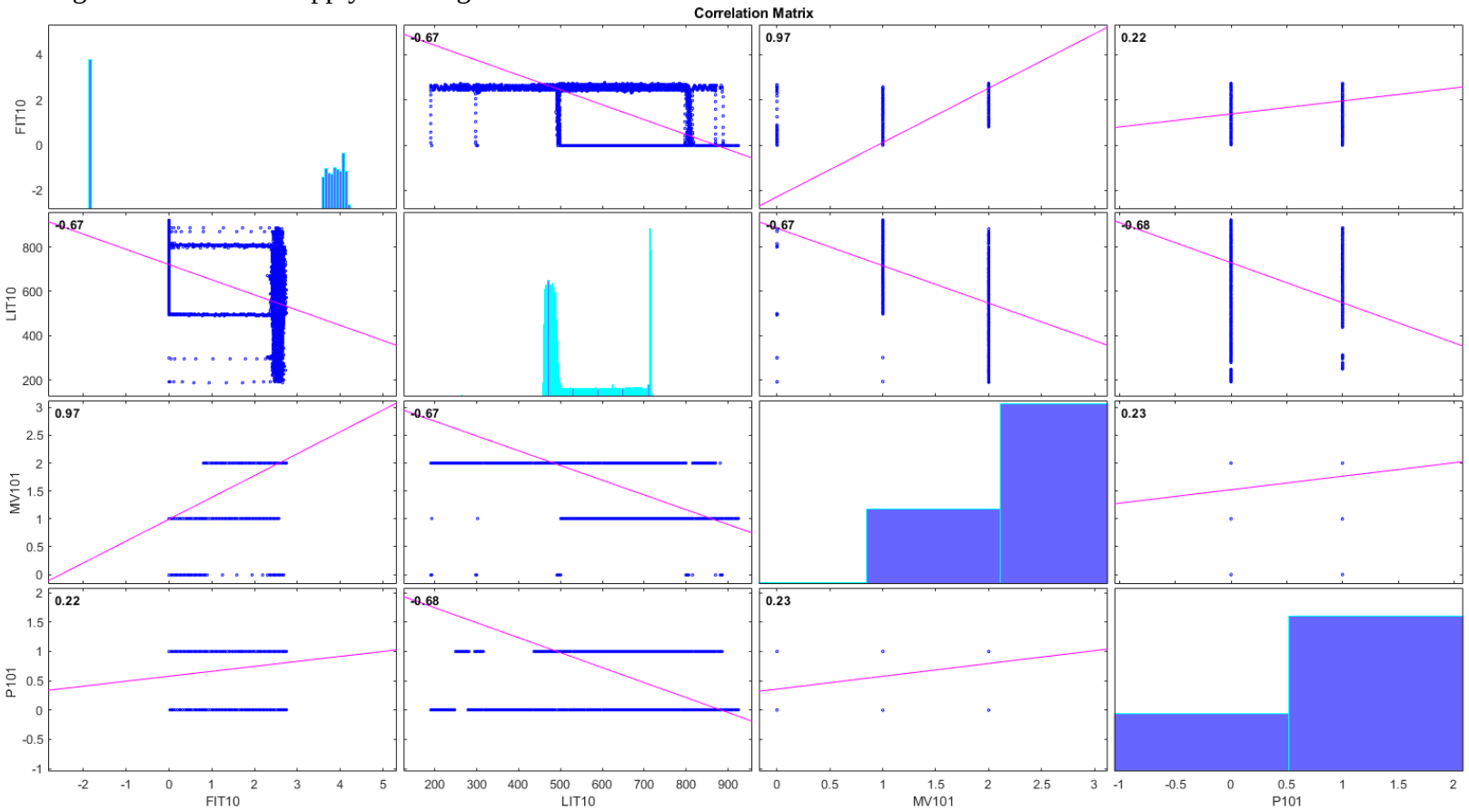

Figure B-1. Stage P1 correlation matrix (1 of 1)

2. Stage P2: Chemical dosing

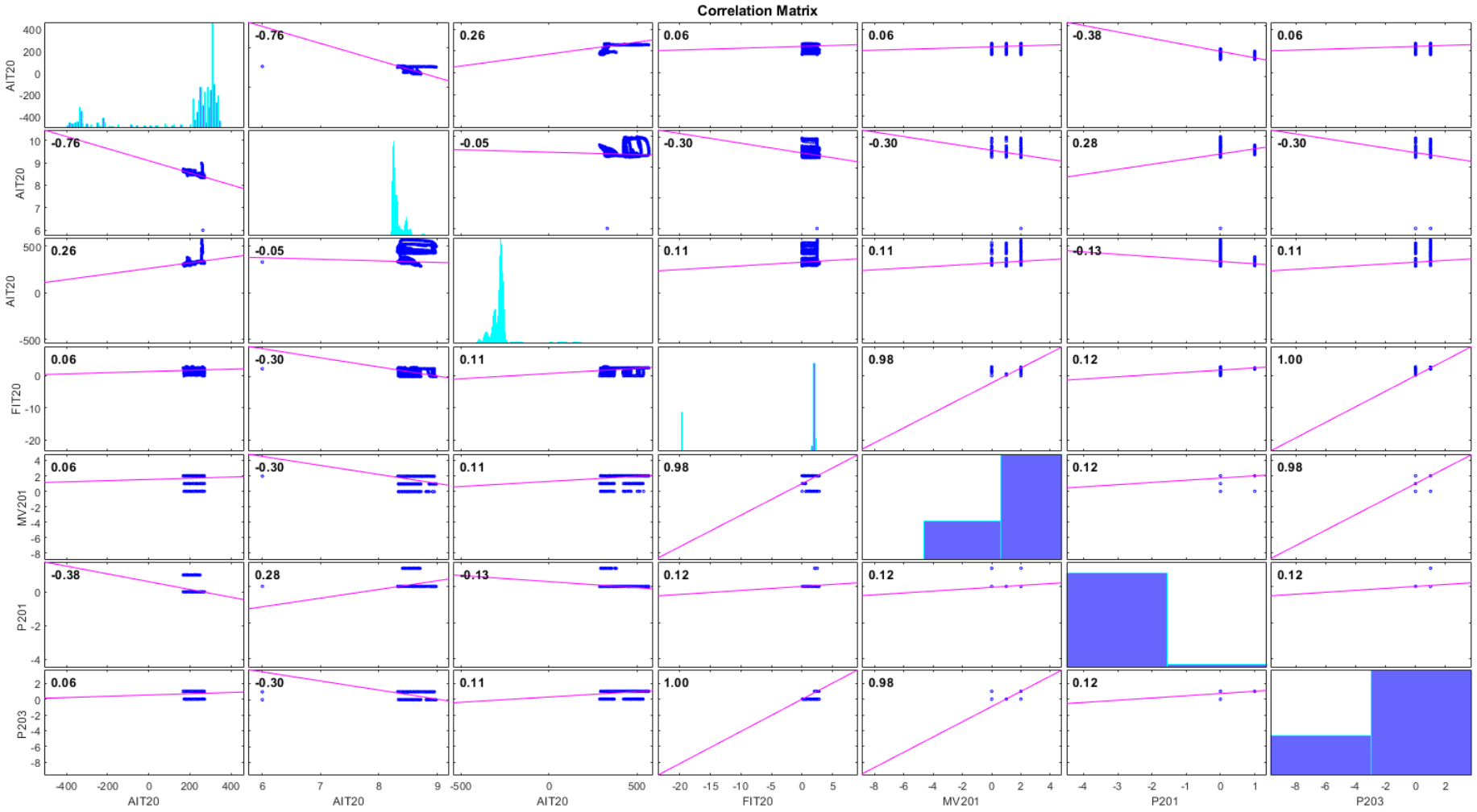

Figure B-2. Stage P2 correlation matrix (1 of 3)

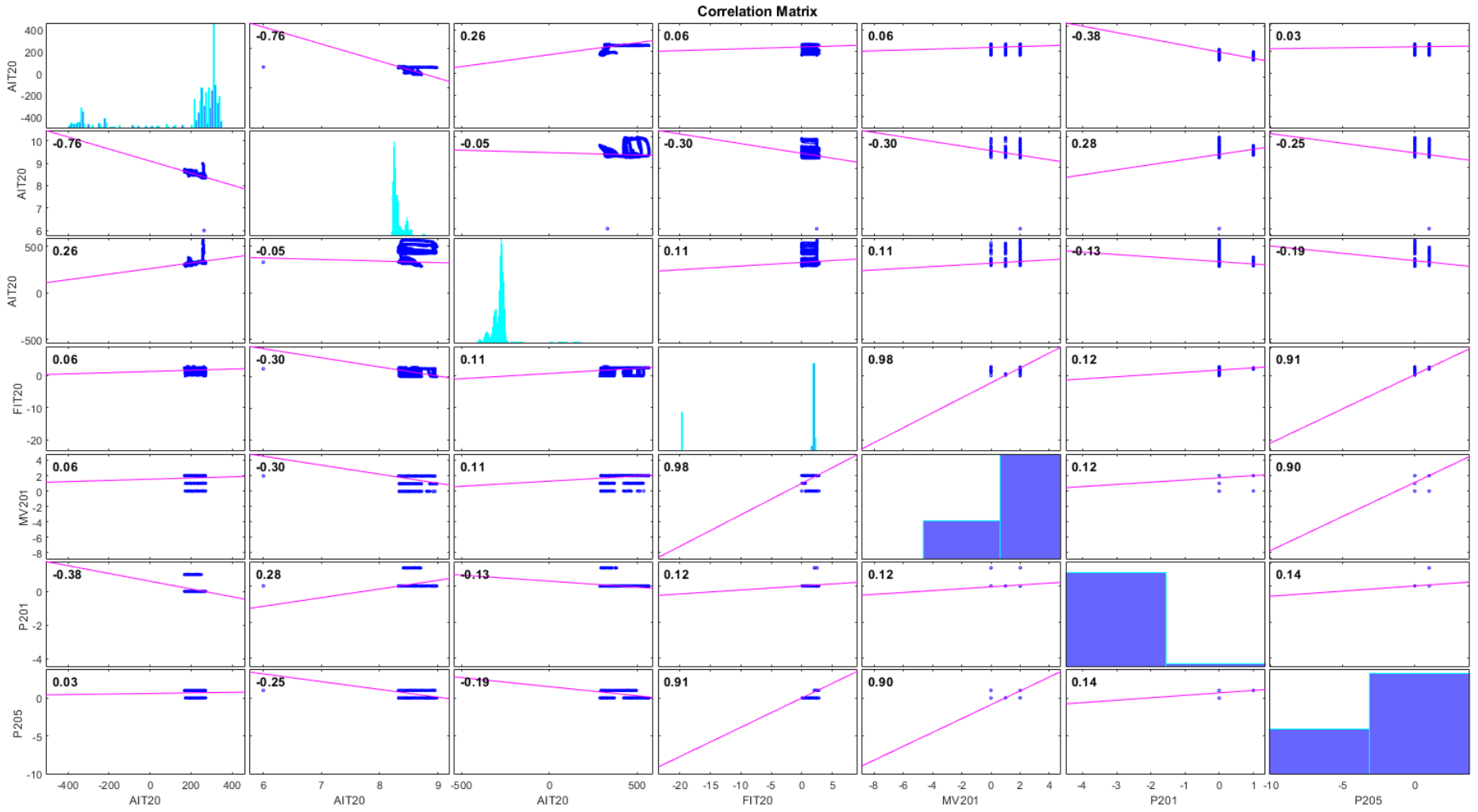

Figure B-3. Stage P2 correlation matrix (2 of 3)

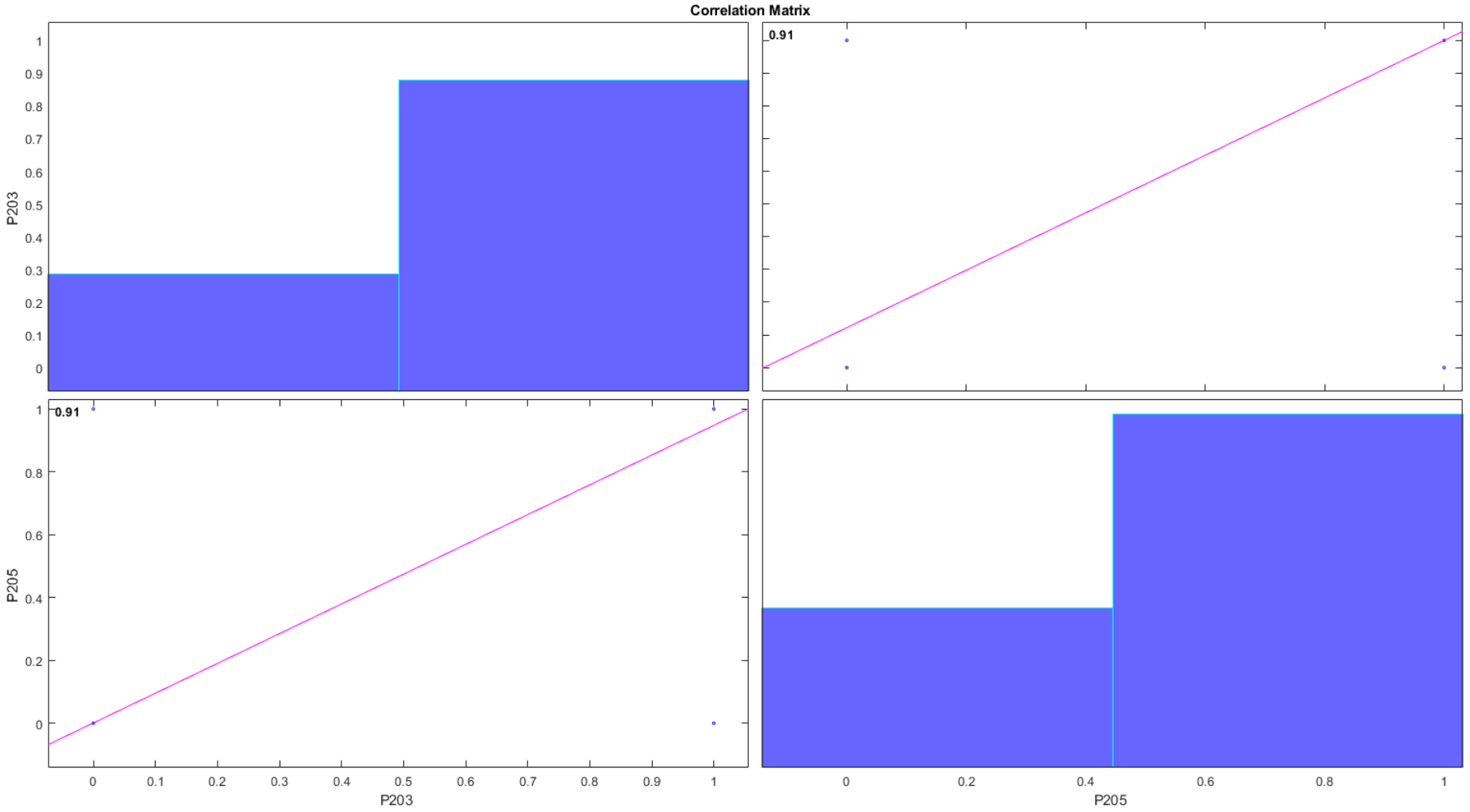

Figure B-4. Stage P2 correlation matrix (1 of 3)

3. Stage P3: Ultrafiltration

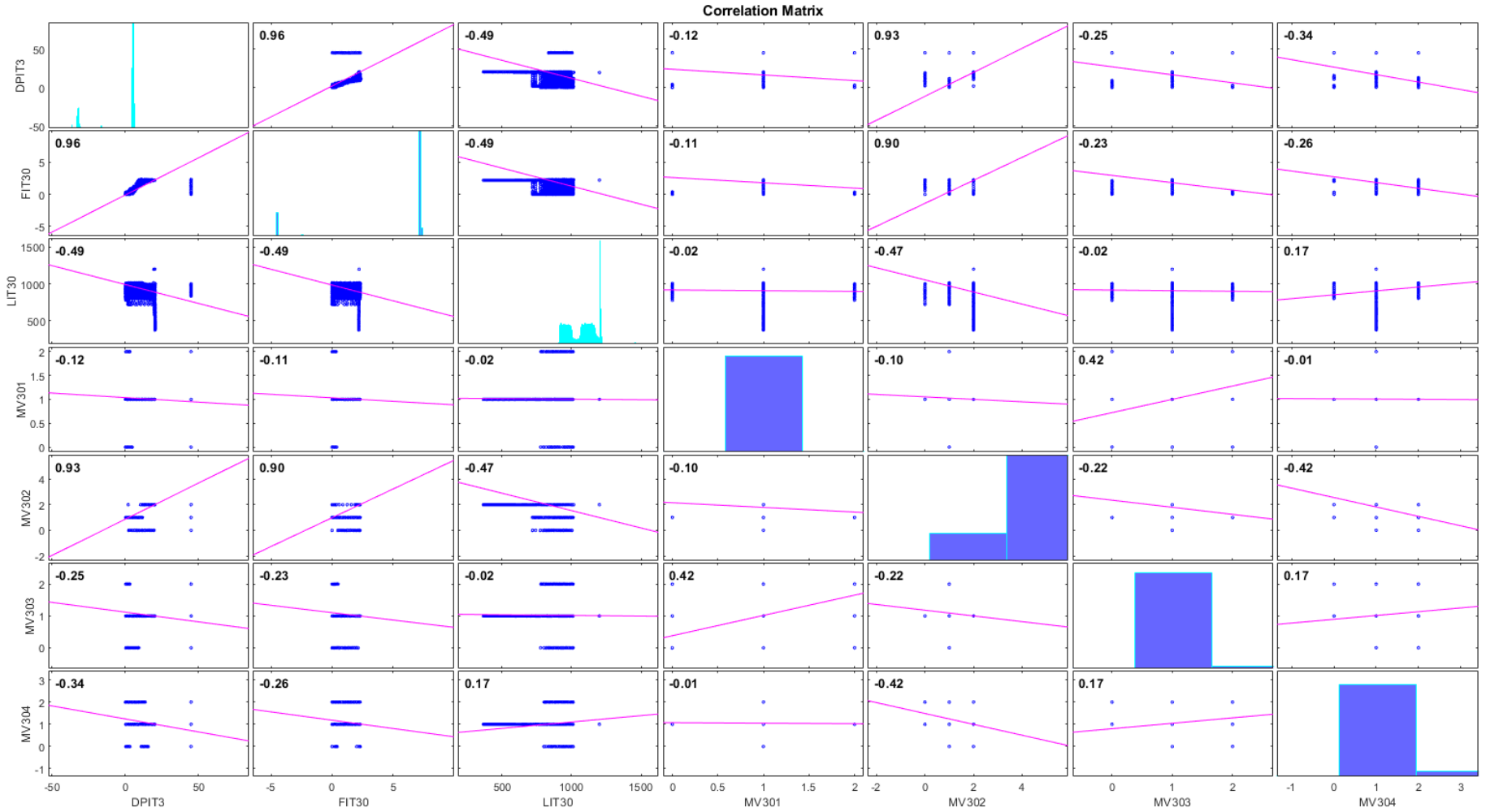

Figure B-5. Stage P3 correlation matrix (1 of 3)

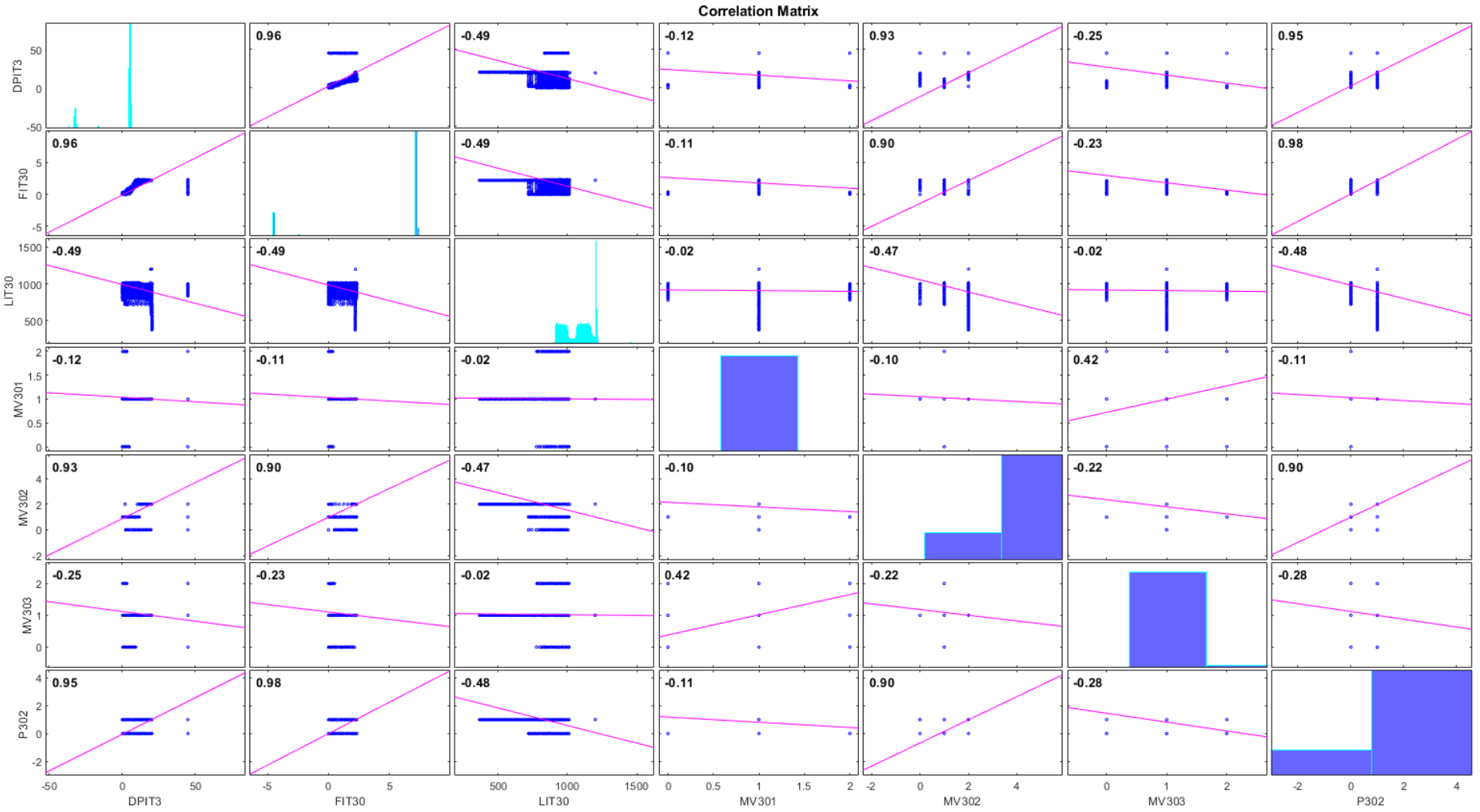

Figure B-6. Stage P3 correlation matrix (2 of 3)

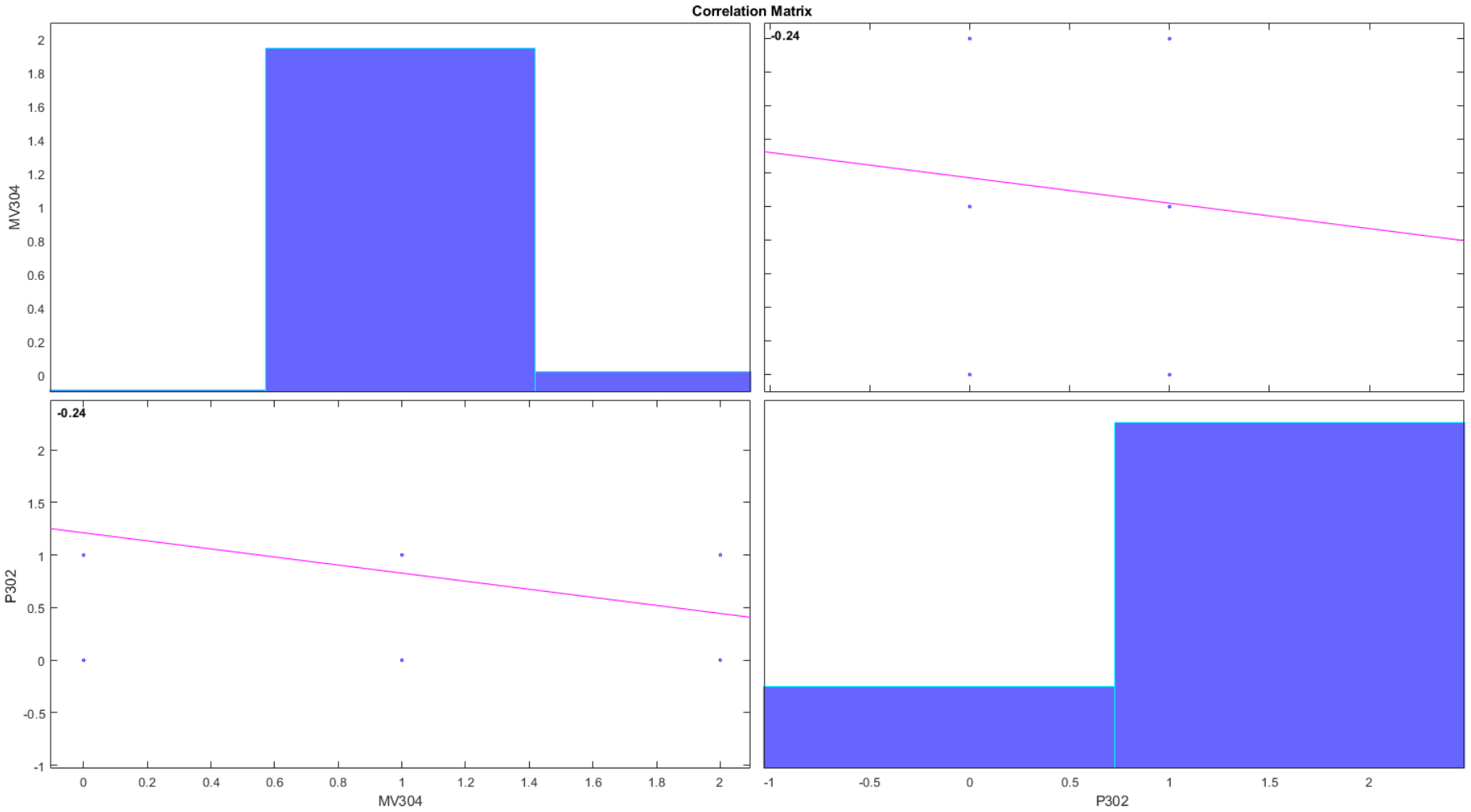

Figure B-7. Stage P3 correlation matrix (3 of 3)

4. Stage P4: Dechlorination

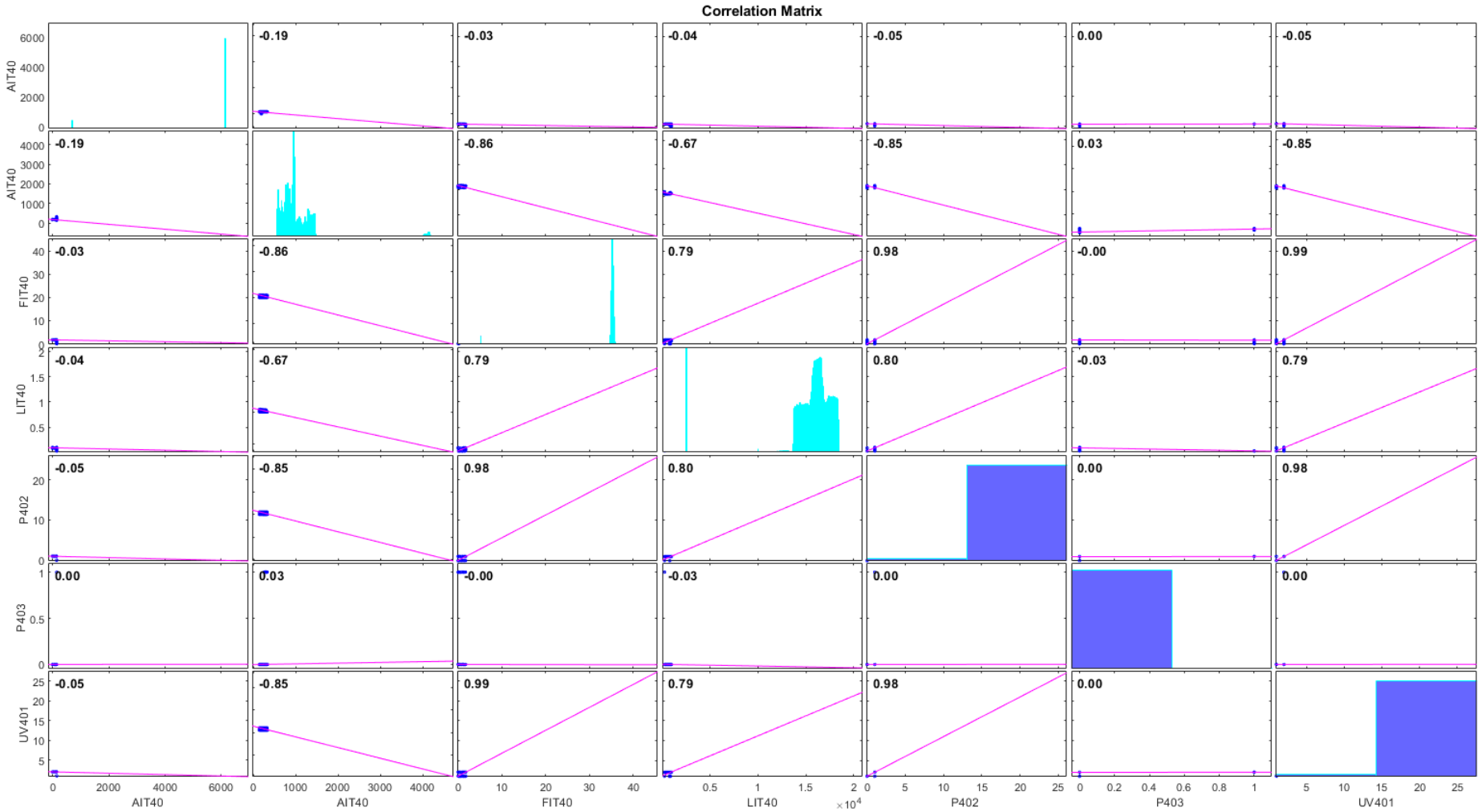

Figure B-8. Stage P4 correlation matrix (1 of 1)

5. Stage P5: Reverse osmosis

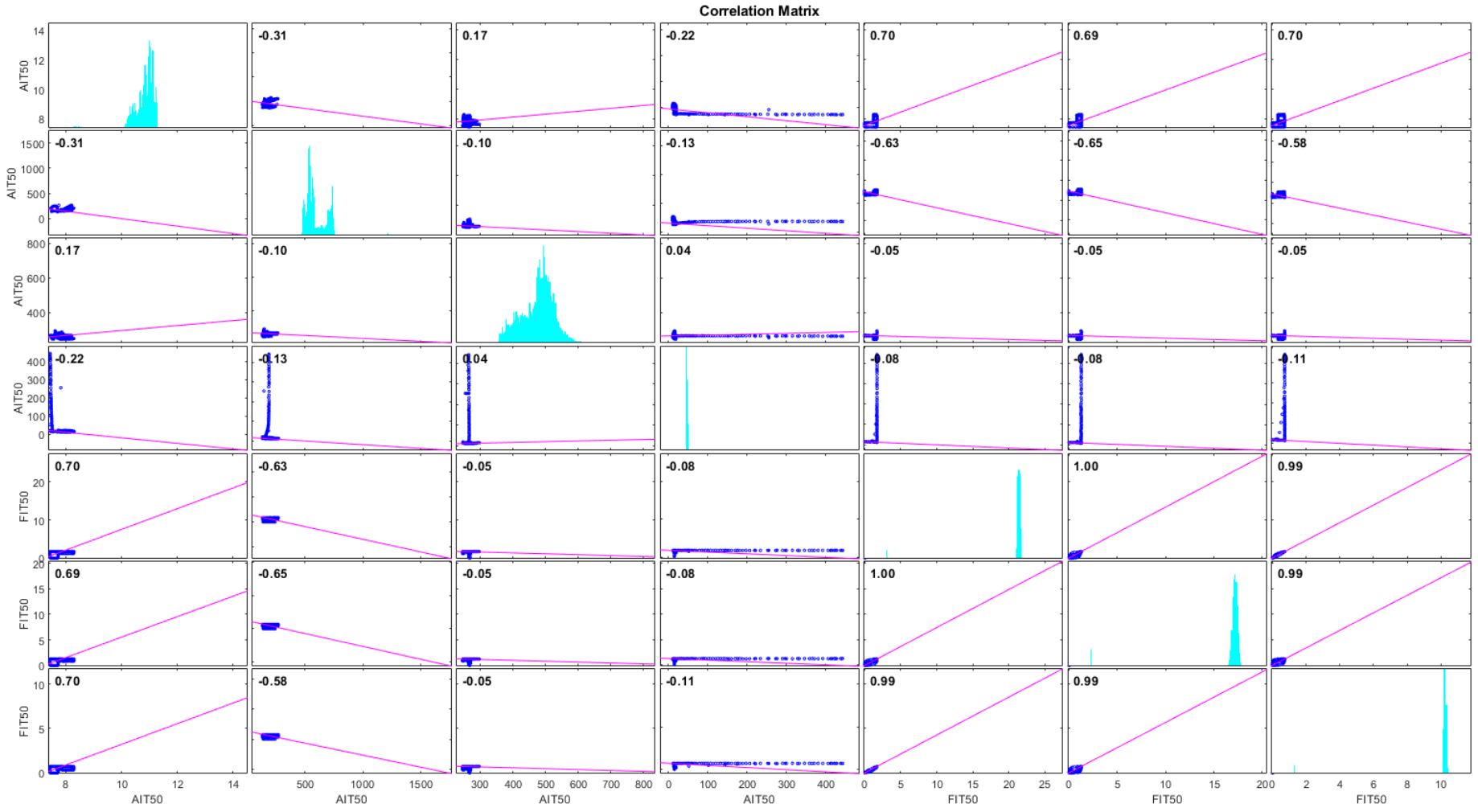

Figure B-9. Stage P5 correlation matrix (1 of 5)

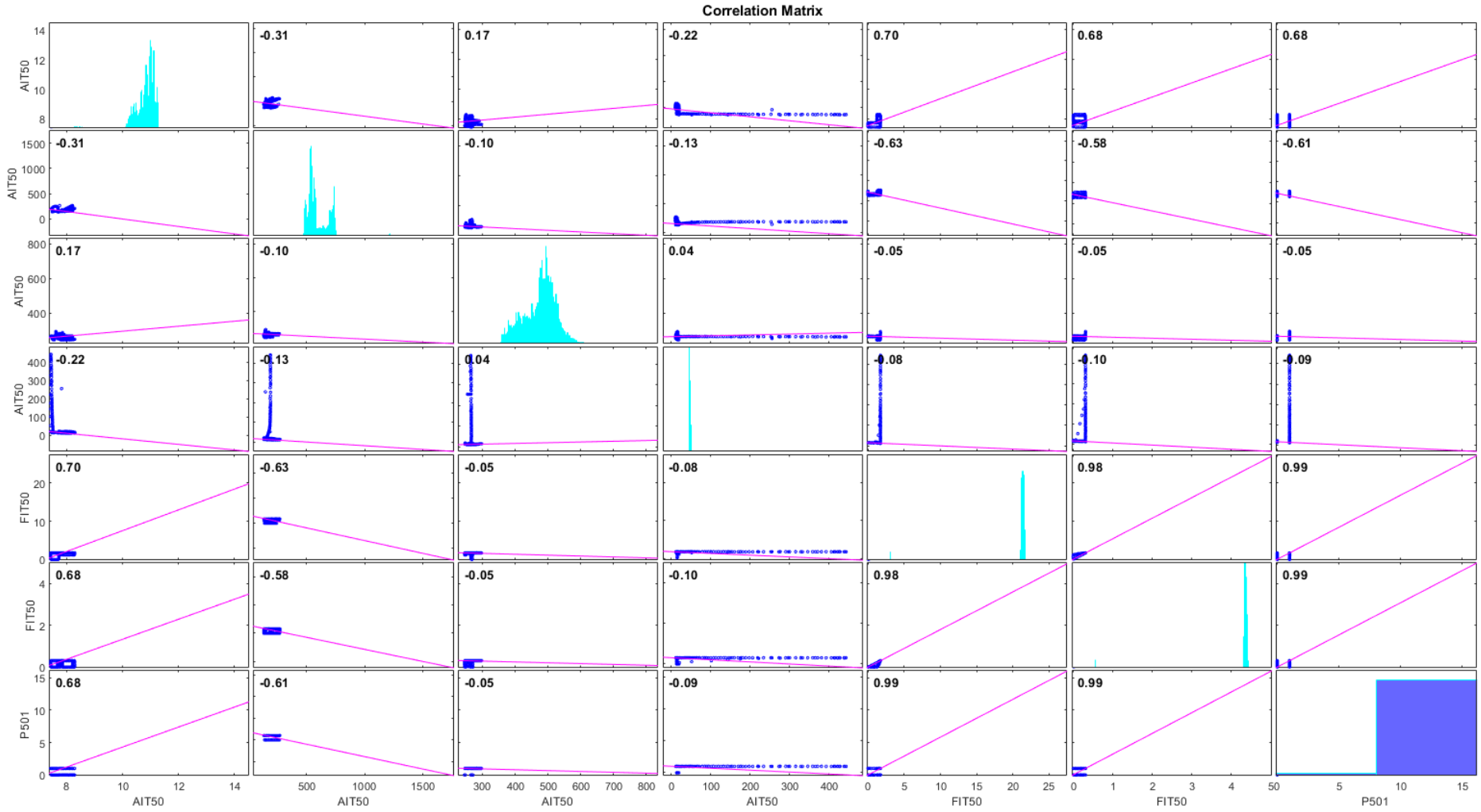

Figure B-10. Stage P5 correlation matrix (2 of 5)

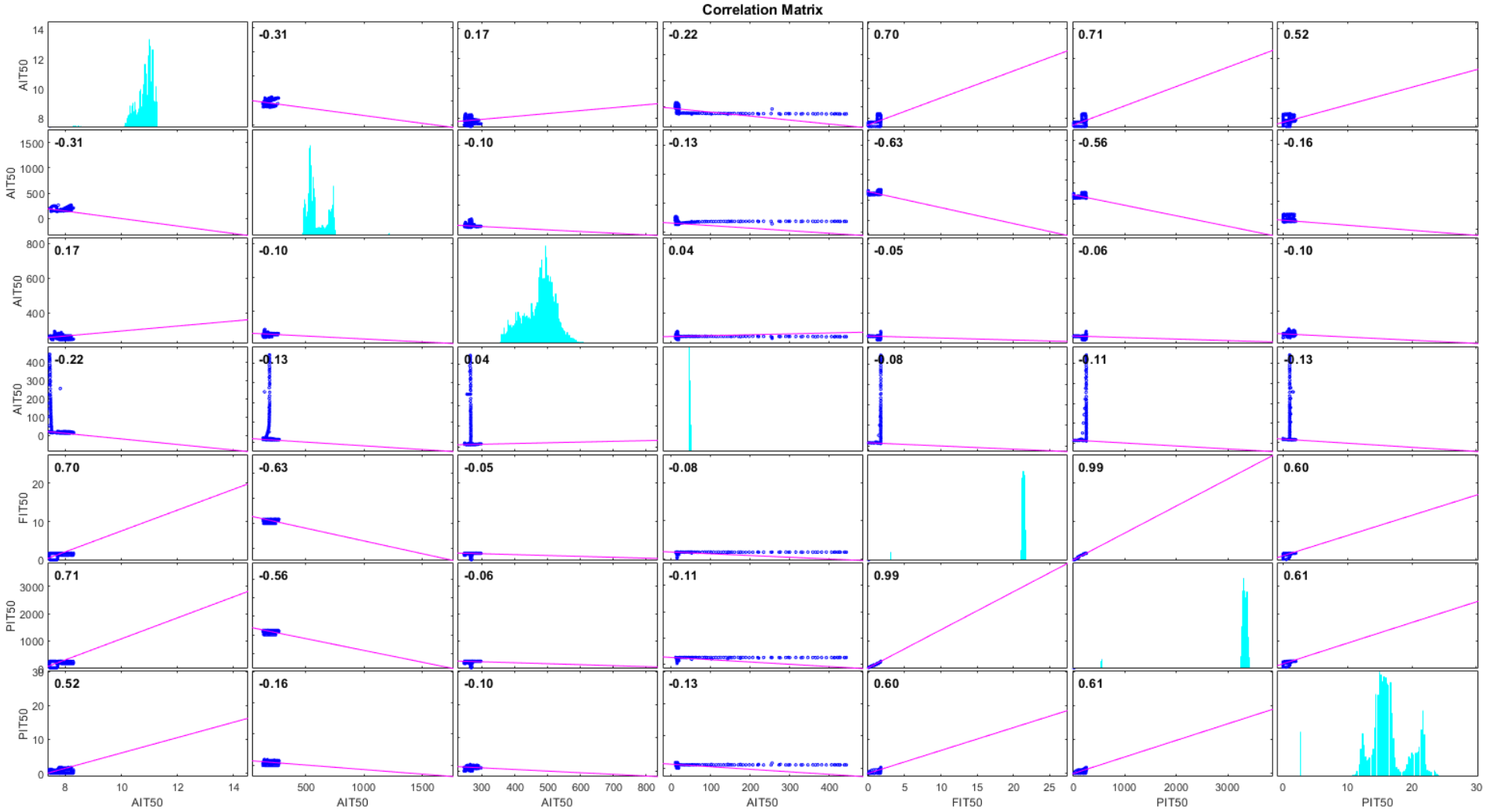

Figure B-11. Stage P5 correlation matrix (3 of 5)

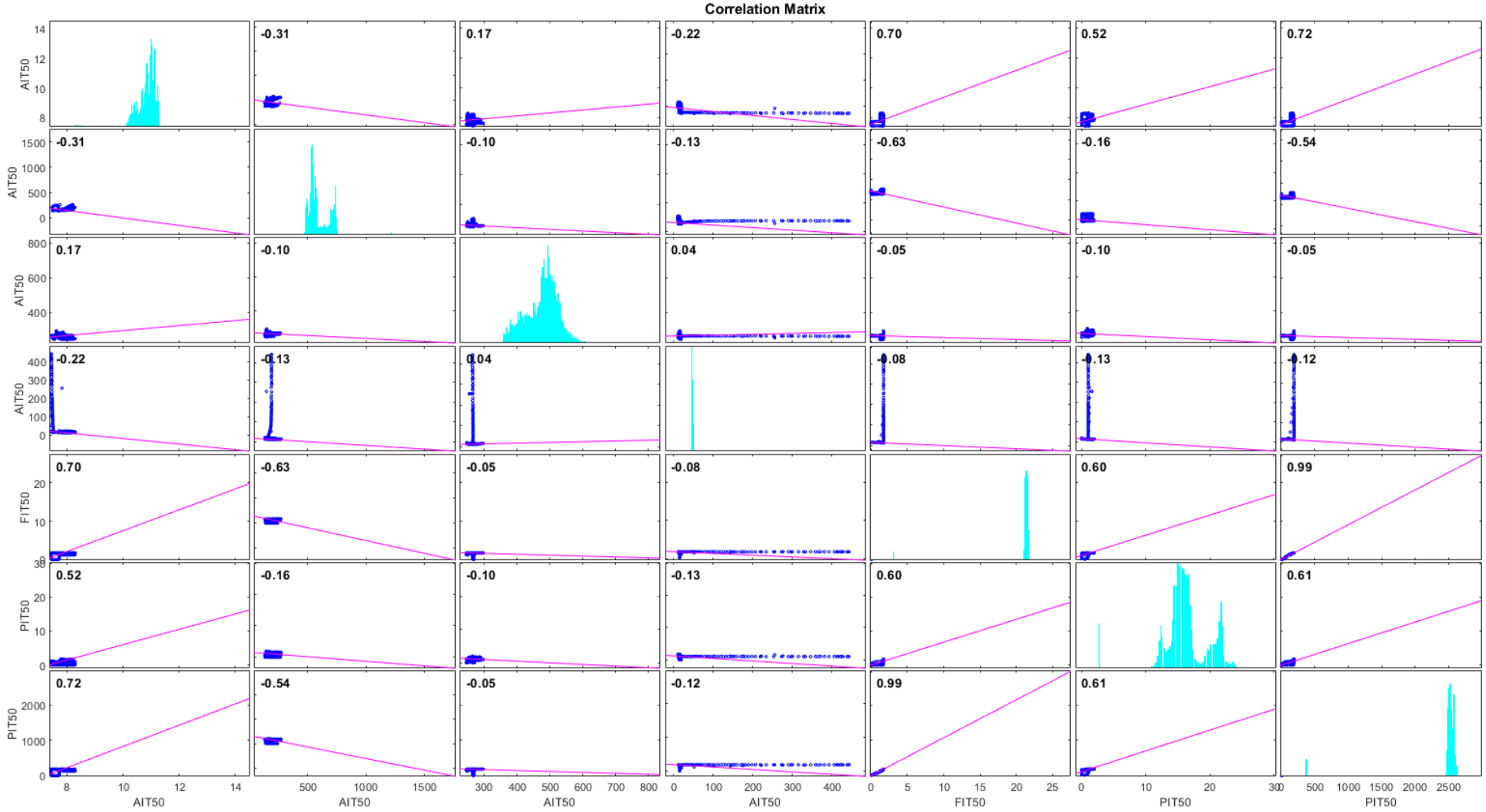

Figure B-12. Stage P5 correlation matrix (4 of 5)

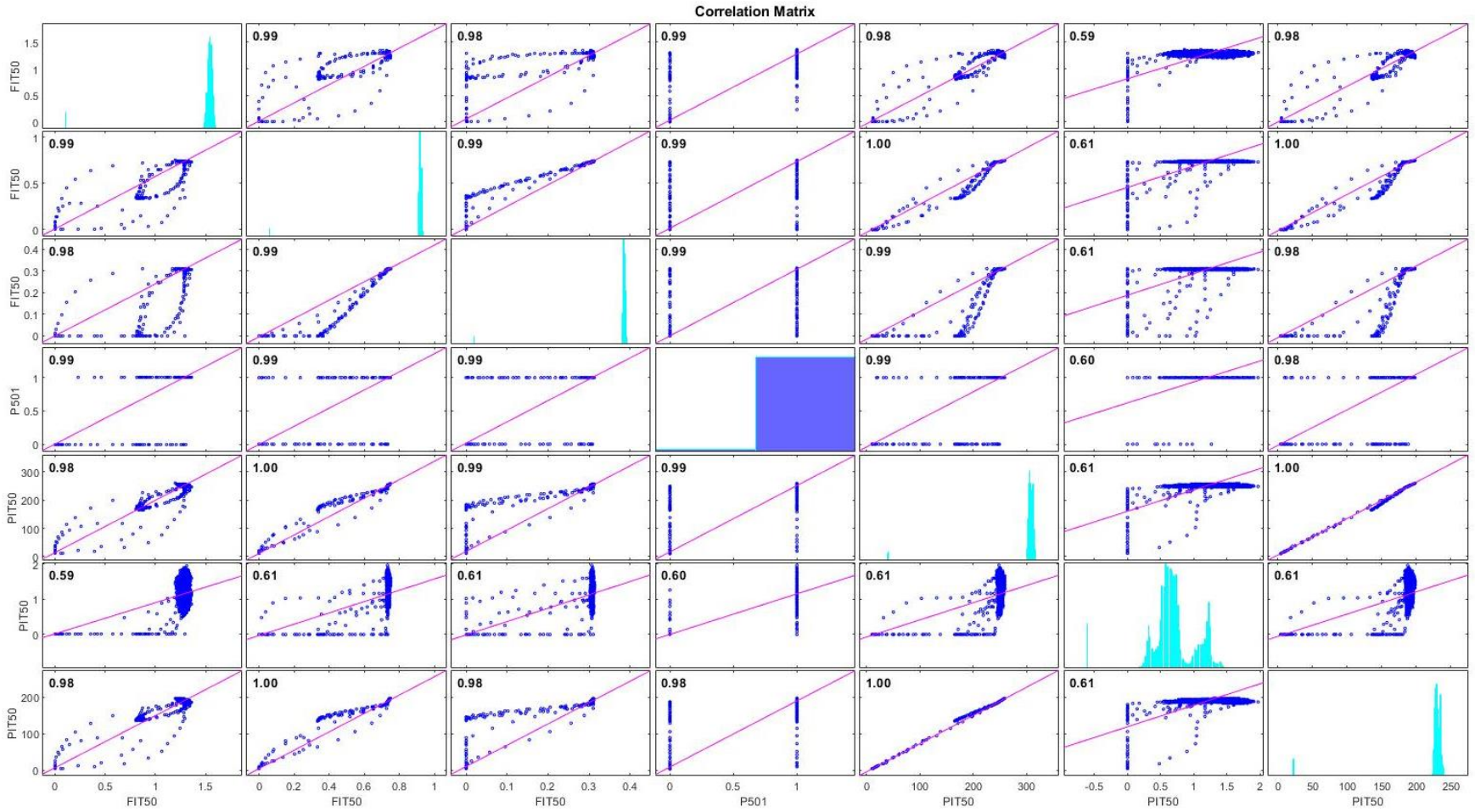

Figure B-13. Stage P5 correlation matrix (5 of 5)