

École polytechnique de Louvain

Process-aware ML Based Intrusion Detection for Industrial Control Systems

Authors: **Guillaume CORNETTE**, **David BODENGHIEN**

Supervisor: **Ramin SADRE**

Readers: **Gorby KABASELE NDONDA**, **Charles-Henry BERTRAND VAN OUYTSEL**

Academic year 2022–2023

Master [120] in Computer Science

Abstract

Industrial Control Systems (ICS) are vital components of many infrastructures and directly impact the real world. Protecting these systems from attacks is essential, especially in the current context with an increasing number of attacks on industrial systems. This thesis proposes implementing an Intrusion Detection System (IDS) based on simple but robust approaches allowing it to adapt easily and not be domain specific. Our approach is based on the Secure Water Treatment (SWaT) testbed created by iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design. Our approach required a rigorous analysis of the data collected by the SWaT simulation in order to correct inconsistencies and generalize our models. We implemented five anomaly detection models, which occur on different types of data. They result in a combination, creating a multi-model approach to deciding whether an event must be triggered. Our models give us results showing the reliability of our robust statistical approach.

Contents

1	Introduction	4
2	Background	7
2.1	System Overview	7
2.2	General Knowledge	10
2.2.1	Time Series	10
2.2.2	Quantiles	10
2.2.3	Interquartile Range	11
2.2.4	Autocorrelation	11
2.2.5	After effect	12
2.2.6	Confusion Matrix	12
3	Related work	14
4	SWaT Dataset	17
4.1	Introduction	17
4.2	Clearing dataset	18
4.2.1	Typographical errors	18
4.2.2	Duplicate Attack error	18
4.2.3	Timestamp attack error	19
4.2.4	Missed Value error	19
4.2.5	Unused Data	19
4.3	Transformation	20
4.3.1	Removing outliers	20
4.3.2	Remove noise	21
4.3.3	Slope Computation	23
4.4	Type of data	24
4.4.1	Continuous data	24
4.4.2	Discrete data	28

5	Models	29
5.1	Threat Model	29
5.2	Combination of models	31
5.3	Variance Zero	32
5.4	Min-Max Values	32
5.5	Longest Sequence of Slope Direction (Cyclic LS)	33
5.5.1	Moving Median Window	35
5.6	Longest Sequence Discrete Value (Discrete LS)	37
5.7	Markov Chain	38
5.7.1	Introduction	38
5.7.2	Illustration and definitions	38
5.7.3	Creation of states	40
5.7.4	Feature combination	43
5.7.5	Quantile Choice	45
5.7.6	Number of states	47
6	Evaluation	49
6.1	Introduction	49
6.1.1	Dataset Test	49
6.1.2	Type of attack	50
6.1.3	After effect evaluation	51
6.2	Accuracy for each Models	54
6.2.1	Min-Max	54
6.2.2	Variance Zero	57
6.2.3	Longest Sequence of Slope Direction (Cyclic LS)	59
6.2.4	Discrete Longest Sequence (Discrete LS)	61
6.2.5	Markov Chain	63
6.3	Multi-model Evaluation	65
6.3.1	Confusion Matrix on Multi-Model	67
6.3.2	Combination of models	67
6.3.3	Multi-model Detection	70
6.3.4	Failed Attack Detection	71
6.3.5	Anomaly Detection in Type of attack	71
6.3.6	Multi-model Delay	72
6.4	Limitations and discussion	75
6.4.1	Accuracy	75
6.4.2	Performance	76
6.4.3	Completeness	76
6.4.4	Fault Tolerance	77
6.4.5	Timeliness	77
6.5	Future work	78

7	Conclusion	79
8	Appendix	81

Chapter 1

Introduction

Industrial Control Systems (ICS) is the designation used to describe types of control systems and their associated instrumentation. They have recently been one of the main targets for hackers for monetary, political, or sabotage purposes that can lead to wide consequences. ICS are targets that are often attacked, especially since it is becoming more and more widespread. The rising labor costs, difficulty in hiring, and meeting deadlines are leading to the massive adoption of automated factories. It is even more important to be interested in attacks on this kind of system because it can affect critical infrastructures. They often lead to serious economic, health, and ecological consequences.

The automation of industrial systems has increased the usage of ICS, and it goes on: its market has registered a compound annual growth rate of 8.9% over the 2018 - 2028 forecasting period.[1]

The use of ICS allows to get real-time information, monitor, manage and control an industrial process.

ICSs require a high level of security with a particular focus on safety. In order to reduce the risk of these attacks having a significant impact, they must be detected as soon as possible. For this purpose, different Intrusion Detection System (IDS) implementation solutions have been sought. Skinner et al. [2] explain that IDS are used in ICS as software designed to monitor behaviors and detect anomalies. IDS are one of the most needed precautions for ICS in complement with traditional security measures like firewalls or monitoring tools to lock down private networks [3].

We can cite Stuxnet [4] as a memorable ICS attack. It has been designed to destroy the centrifuges that Iran was using to enrich uranium for its nuclear program.

Stuxnet is a malicious computer worm discovered in 2010 and aimed at ICS. The worm affected the Programmable Logic Controllers (PLCs) by exploiting vulnerabilities in the programs and hardware used by the systems. The malicious program

evaded detection by traditional security measures and caused significant damage to the industrial control systems it targeted. This highlights the need for effective IDS systems to detect and respond to such threats. It is widely regarded as one of the most significant cyber attacks in history because of the potential consequences. Although it is a large-scale attack, smaller ones can also have significant consequences. These attacks are becoming more and more frequent, it is essential to build security mechanisms to avoid the worst.

This thesis explains how we have developed an IDS that allows real-time data analysis with a high detection/complexity ratio while having a non-dataset-specific approach applicable to any industrial domain. This solution can be deployed very quickly and allows an understanding of the data's behavior and the detected anomalies. We have implemented the following models: Min-Max value, Variance Zero, Longest Sequence of Slope Direction, Discrete Longest Sequence, and a model based on a Markov Chain.

Below we made a small description for every model showing the purpose of each of them and how they work:

- The "Min-Max value" is very efficient; it detects when a value is out of its normal bounds. It is the most effective model.
- The "Variance zero" is used to detect when backups take over, which should indicate an anomaly.
- The "Longest sequence of slope direction" indicates behavior where the data from a sensor is either abnormally increasing or decreasing or staying the same for too long.
- The "Longest sequence of value" detects when an actuator (a pump, for example) keeps the same value for too long.
- The "Markov Chain" creates an abstraction of the whole system capturing normal behavior to compare them with potential anomalies.

We use all the models above all together in what we call a Multi-model which is the concatenation of our five models.

We have focused our work on the accuracy, temporal complexity, speed of detection, and easiness of deploying and adapting to any ICS.

These models are straightforward, require very little computation time, and are robust by their nature. Each model is applied according to the type of data faced. Thus, our IDS can be easily adapted to different domains. Its simplicity makes it

easy to understand where the anomalies come from, but also how and why they occurred. This is what differentiates us from the different solutions that have already been implemented for the SWaT developed by Itrust.

We based our analysis on a SWaT testbed ICS developed by iTrust [5].



Figure 1.1: Picture of the SWaT System in Singapore [6]

This testbed is composed of six sub-processes, recreating a water treatment facility. Each of the sub-process is controlled by a set of dual PLCs, and then the operation status is monitored by a Supervisory Control and Data Acquisition (SCADA) system.

The structure of this thesis is the following; we will explain in chapter 2 the process used in SWaT and background knowledge, then briefly discuss Related works. In chapter 4, we will explain the analysis and the modifications applied to the dataset provided by ITrust. Finally, we will explain in two separate chapters our models and their evaluation. The evaluation contains a section for the limitations of our models.

Chapter 2

Background

2.1 System Overview

The system is a SWaT ICS developed by iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design. The SWaT is a fully functional small version of an industrial water treatment facility; it is split into a six-stage process. The process consists of taking raw water and then adding chemicals, filtering, and de-chlorinating it. Then it passes through a reverse osmosis system, and a backwash process begins [6]. The SWaT has been a central resource since 2015 for researchers working on safer cyber-physical systems (CPS). Security of CPS is a key element in cybersecurity as they are used everywhere in our daily life, in critical environments, and in government [7].

Concerning the architecture, it consists of Human Machine Interfaces (HMI), PLCs, Supervisory Control and Data Acquisition (SCADA) workstation, and a Historian. Most of ICS are composed of each of these components, which enables the gathering, monitoring, and control of the system.

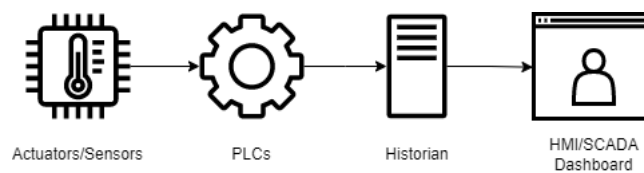


Figure 2.1: SWaT's cyber architecture

The diagram above shows the simplified process of the SWaT's cyber architecture, excluding the layered network part. Each of the 25 actuators and 26 sensors shares one of the 6 PLCs, which defines one of the six stages of the SWaT shown

in Figure 2.2.

Every PLC gets its inputs from the sensors in order to monitor the system. The programming logic controller (PLC) uses the actuators to manipulate the state of the physical process. As an example, the sensor LIT101, which is used to track the raw water tank, will send its data to the PLC dedicated to the stage P1: Raw water supply & storage. Depending on a threshold, the PLC will deactivate or not, the valve MV101 (an actuator) to stop adding water into the tank as explained by Wang et al. [8]. This process is illustrated in Figure 2.2 where each stage is controlled by a set of dual PLCs to allow a PLC to take over if one of the two fails [8]. To prevent faults, the SWaT also contains backups sensors and actuators in order to take over if needed and it contains several safety mechanisms embedded in the PLCs. The Historian is used by our IDS to collect data.

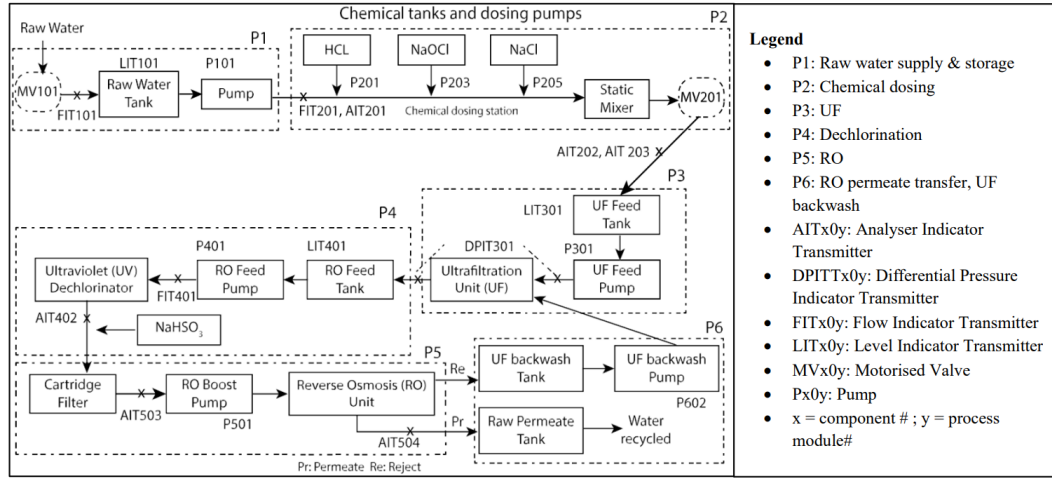


Figure 2.2: Swat sub-process from P1 to P6 [6]

As we can see in Figure 2.2, the process is split into several phases from P1 to P6, which correspond to the physical steps of the water treatment plant. SWaT consists of different sensors to ensure that the process is safe. Here are the following sensors and actuators [6] :

- LIT - Level Indication Transmitter (measured in mm)
- FIT - Flow Indication Transmitter (m3/hr)
- AIT - Analyser Indicator Transmitter
 - Conductivity ($\mu\text{S}/\text{cm}$)
 - pH

- Oxidation Reduction Potential (mV)
- DPT - Differential Pressure Indicator Transmitter (kPa)
- PIT - Pressure Indicator Transmitter (kPa)
- MV - Motorized Valve
- P - Pump
- UV - Dechlorinator

All of the actuators and sensors are described in the appendix 8.1.

The IDS has to be efficient and reliable. To evaluate this, we base our design and evaluation on the following principles introduced by Porras and Valdes [9].

- Accuracy: The IDS should trigger the least possible amount of errors. The amount of false alarms or anomaly not detected has to be low and positive alarms and normal behaviors detected to be high to have high accuracy.
- Performance: Performance deals with the speed with which events are audited. If the performance is high, it is possible to deal with data in real-time.
- Completeness: The IDS should detect all attacks. On the opposite, an incomplete IDS cannot detect all of them.
- Fault tolerance: The IDS should be resistant to attacks.
- Timeliness: IDS have to raise alarms as quickly as possible this involves the time to propagate the information and to be able to react.

Ideally, an IDS fulfills all of these properties. In addition, it is necessary to consider the specifications and needs of ICS to secure its aspects. The time aspect is essential for ICS because a slight variation can cause significant damage. The performance is crucial because physical equipment does not have much power and cannot support the use of heavy security programs [10].

We assume that the attacks are made by modifying information about the sensors and actuators. By doing so, PLCs miss interpret the actual state of the system. This is called "false data injection attacks" (FDIA), which happens when an adversary manipulates sensors measurement used by the PLCs [11].

2.2 General Knowledge

This part is an overview of the relevant background information and context for our analysis. The purpose of this section is to provide a general understanding of the topic being studied and to establish the needs for the research presented in the thesis. This information will be used to set the stage for the research presented in the rest of the thesis.

2.2.1 Time Series

A time series is a series of data points indexed in time order. Time series data are used to understand how a metric or a variable changes over time. It is a sequence of data points recorded at regular intervals of time in seconds for our case. Time series data can be continuous (e.g., water flow, pH, pressure) or discrete (e.g., pump and valve states) [12].

2.2.2 Quantiles

The notion of quantile is the general term to designate the values that divide a data set into intervals containing the same number of data. A quantile of 25% gives us the value which will split that data into two parts, the bottom part, which is 25% of the data, and the upper part, which is also 75% of the data set. The quantiles are essential to create thresholds on the dataset in order to create what we will later call Status, which will determine our states in the Markov Chain. The decision of the choice of our quantiles is crucial in our research project. It is significant to remember that the quantiles selection will depend on the specific features of the dataset and the analysis's objectives. Before choosing which quantiles to employ, it is usually a good idea to study and visualize the data.

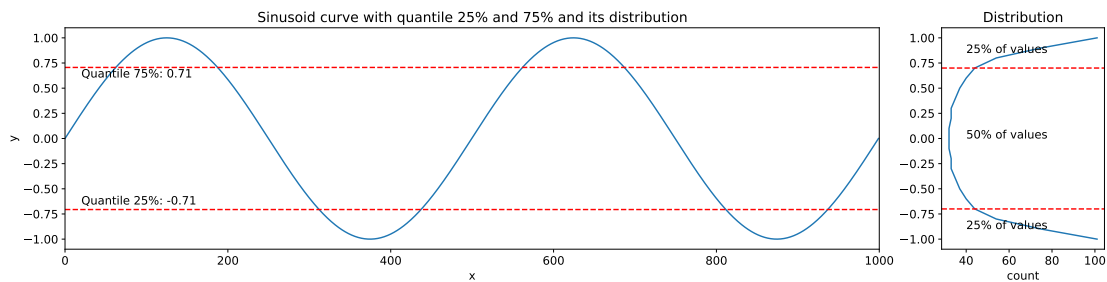


Figure 2.3: Sinusoid curve with quantiles 25% and 75%

Here is an example of the use of quantiles on a sinusoid curve present in the Figure 2.3 above as well as its distribution curve. Using the same quantiles as at the beginning of the explanation with quantiles 25% and 75% represented in red on the Figure 2.3. From now on, when we talk about quantiles in the thesis, we will only mention a percentage x to which we will create quantiles Q1 and Q2 such that $Q1 = x$ and $Q2 = 100 - x$ to divide the data in three parts. We will refer to this convention in the rest of the thesis although it is not the classical one. The bottom part will be equivalent to the data point in the range below the Q1 quantile of the data distribution, starting below threshold equal to -0.71 (see 2.3). The part above the Q2 quantile of 75 % of the data distribution will be defined above the threshold equal to 0.71. This part is also equivalent to 25% of the data. The third part will take all points between Q1 and Q2. As we can see, we have two quantiles separating the data perfectly by giving $x = 25\%$ on each side of the distribution curve we mentioned above.

2.2.3 Interquartile Range

The Interquartile Range (IQR) Frost [13] method can be used to locate outliers in a dataset and possibly remove them. Outliers are exceptional or extreme values that might not be typical of the data as a whole. The Interquartile Range (IQR) of the dataset, which is the difference between the 75th percentile (Q3) and the 25th percentile (Q1), must first be determined in order to apply the IQR method. Then, using this value, we may spot any values that deviate significantly from the norm as outliers. For instance, we might classify as an outlier any result that is either below $Q1 - 1.5 * IQR$ or above $Q3 + 1.5 * IQR$. We can choose to ignore the outliers or remove them from the dataset after we've found them.

2.2.4 Autocorrelation

We will need a technique to detect the cyclicity of our variables called autocorrelation, a mathematical technique used frequently in signal processing. It is the cross-correlation between a time series and a lagged version of itself. It measures the degree to which the values in a time series are correlated with values that come before or after them. The following formula explains the autocorrelation:

$$ACF(k) = \frac{Cov(X(t), X(t+k))}{Var(X(t))} \quad (2.1)$$

where $X(t)$ represents the time series and k represents the LAG value in seconds [14].

2.2.5 After effect

When the system is attacked on a sensor or an actuator, the other features may suffer the indirect effects of this attack. An attacked feature may create a cascading effect and will change some of the values of the sensors and actuators that have an indirect or direct link with the initial target. Another possible consequence of an attack is a change in the value of a sensor outside its limits. This change may take time to resolve and to make system returns to a stable and normal state. During this period, the system may no longer be under attack, but the feature is still experiencing the consequences of the attack. This phenomenon can have consequences on the evaluation of the result by creating false alarms. This is called the after effect, which results in unpredictable states that are not labeled as attacks but can be detected as such by our models.

2.2.6 Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model, such as an anomaly detection model. It is a useful tool for understanding the strengths and weaknesses of a model, and for identifying areas for improvement.

A confusion matrix consists of four counts: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). These counts represent the number of times the model made the following predictions:

- True positive (TP): The model correctly predicted that a data point was an anomaly.
- True negative (TN): The model correctly predicted that a data point was not an anomaly.
- False positive (FP): The model incorrectly predicted that a data point was an anomaly.
- False negative (FN): The model incorrectly predicted that a data point was not an anomaly.

Depending on the precise objectives of the anomaly detection model, these metrics may or may not be significant. For instance, a high proportion of TP is preferred if the objective is to find as many anomalies as possible. On the other hand, a low level of FP is preferred if the objective is to reduce the number of false alarms.

In some cases, it is essential to detect anomalies, even if it may imply having some FP. On the other hand, sometimes detecting wrong anomalies (FP) can be more harmful than not diagnosing an anomaly.

Useful Metrics

With the help of the confusion matrix, we can recover some metrics that will be very useful in measuring the efficiency of our models.

- The **Precision** is a measure of the accuracy of the model's positive predictions. It is calculated by dividing the number of True Positive by the total number of positive predictions made by the model:

$$\frac{TP}{TP + FP} \quad (2.2)$$

A model with high precision is one that is very accurate at predicting positive cases but may not be as good at predicting negative cases [15].

- The **Recall** is a measure of the completeness of the model's positive predictions. It is calculated by dividing the number of TP predictions by the total number of actual positive cases:

$$\frac{TP}{TP + FN} \quad (2.3)$$

A model with high recall is one that is very good at identifying all of the positive cases but may also have a higher number of FP [15].

- The **F1 score** is a metric that combines precision and recall into a single measure. It is calculated by taking the harmonic mean of precision and recall:

$$\frac{2 * (precision * recall)}{precision + recall} \quad (2.4)$$

The F1 score is useful because it provides a balance between precision and recall, and will allow us to see the overall performance of a model. A high F1-score indicates that the model performs well in terms of precision and recall. More specifically, a high F1-score indicates that the model has a high proportion of True Positives and a low proportion of False Positives. A low F1-score indicates that the model has poor precision and/or recall performance. This may be due to a high percentage of False Positives or a low percentage of True Positives [15].

Chapter 3

Related work

Security in ICS is a crucial topic, a lot of research has already been done to find solutions for IDS. A wide range of specialized algorithms has been developed for anomaly detection and analysis in time series, from basic statistical analysis and signal processing to data mining and deep learning, and it contains a much larger list.

As mentioned by Burgetová et al. [16] the most used approaches are traditional statistics and machine learning models or a combination of both of them. But traditional statistics, although very robust, are often too specific and only work under certain assumptions.

Choi et al. [17] provide a comparison between several time-series anomaly detection algorithms. The general result that emerges from the experiments done on three real-world datasets, including the SWaT, shows that each algorithm's results depend a lot on the dataset used. However, they highlighted some points : The Deep Autoencoding Gaussian Mixture Model implementation by Zong et al. [18], which is an Autoencoders (AE) multivariate algorithm that does not take into account temporal data, is inferior to Recurrent Neural Network (RNN) based models. It is explained by the fact that they can take long sequences as input and capture temporal dependencies. They conclude that the approaches must be able to take into account temporal dependencies in order to be applicable to times-series anomaly detection. In spite of its capacity for sequence modeling, one of the problems of RNN is its restriction to parallelization because it calculates its output sequentially.

Moreover, RNN is outperformed in several applications developed by the following searchers Choi et al. [19], Wen and Keyes [20], Zhou et al. [21] that use short-term data by Convolutional Neural Network (CNN). CNNs do not explicitly capture the temporal context but learn patterns in segmented time series, which creates one of its disadvantages, that is to say, to understand behaviors appearing

over a long period. This introduces Temporal Convolutional Networks, a variant that was proposed by Bai et al. [22].

RNN has been extended by variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) to solve the vanishing or exploding gradient problem. However, LSTM needs a lot of training memory in case of a long sequence input. LSTMs are strongly present in the detection of anomalies for times series. In the work of Gjorgiev and Gievska [23], the authors propose a solution that combines LSTM with AE. Like LSTM, AE also has many research papers showing their efficiency, as in work done by Wang et al. [24] putting forward good results although using a static threshold to detect attacks.

Schmidl et al. [25] re-implemented 71 anomaly detection algorithms from different domains and evaluated them on 976 time-series datasets in order to present the different families of possible detection techniques and to evaluate them. Two elements that stood out were the cost/benefit ratio and the memory limit. Indeed, the algorithm with the best cost/benefit ratio according to Schmidl et al. [25] is DWT-MLEAD which stands for Discrete Wavelet Transform and Maximum Likelihood Estimation for Anomaly Detection. It has an average score (using AUC-ROC) of 83% and a runtime of 2.2 ms per data point, which is very low. It is able to analyze time series on different time scales and detect short-term and longer-term anomalies. Next, concerning the memory limit, LSTM-AD and Encoder-Decoder (EncDec-AD), two deep-learning algorithms, had a lot of trouble with the 3GB memory limit. EncDec-AD is capable of employing a reconstruction error mechanism to find anomalies after learning to rebuild "normal" time-series behavior. This memory problem has already been mentioned above and is common in deep learning.

In the paper written by Feng et al. [26], the authors have implemented an invariant monitoring system that defines the physical rules that must be respected in order for the ICS to operate normally and detect anomalies quickly. Their implementation avoids the high cost of generating these rules manually by systematically generating them from the logs using a combination of machine learning and data mining techniques. This approach raises a good point; it allows to easily detect the afters effects of attacks by taking them into account based on the transitions of the system and not on its fixed state. This allows them to have a very low False Positive rate. With this positive point also comes a negative point: there is a way to make attacks that block the system while respecting the invariants. For their part, our models take into account the time in which a state remains the same in order to detect when the system is blocked.

Feng et al. [26] are not the only authors of a work proposing the specification to create a system to detect anomalies. In Nai Fovino et al. [27] work, they implement a firewall using an internal representation of the controlled SCADA system.

Compared to these implementations, our solution raises some important features: our training time and memory training are extra low. Our models do not require hyperparameters like most of the algorithms mentioned above, consisting of neural networks (LSTM, RNN, CNN, AE, etc...).

Flexibility, reliability, and simplicity are three axes that have been mentioned by Schmidl et al. [25]. In their work, the authors concluded that their algorithm should be improved based on these axes due to the lack of satisfaction with the overall results.

We have implemented our solution with these three axes in mind, and these three features are present in our IDS.

Flexibility comes from the hybrid nature of our anomaly detection and the different strengths of each of our models.

When reliability is discussed in the paper, the authors mention problems related to robustness and scalability, which are caused by time and memory limitations, which are not problems in our implementation due to the simplicity of our models. This simplicity is addressed by Hand [28], where the researchers show that simplistic methods provide results almost as good as extremely sophisticated methods while avoiding the problems related to complexity.

This brings us to the last axis, simplicity. Most of the algorithms studied in [25] require a lot of parameterization, which we avoid by having models that are self-configuring by their generalistic nature.

The use of a Markov-based model allows us to capture the transition relationship between time-dependent data and infer corresponding states. This Markov model-like is combined with a robust statistical approach while being very general. The simplicity of the combination of our different models allows us to avoid the black-box formation created by neural networks and to understand better the events generated by our IDS.

Chapter 4

SWaT Dataset

4.1 Introduction

In order to perform our tests and create our models, we used the dataset provided by iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design. A six-stage Secure Water Treatment (SWaT) testbed was used to implement the data collection method. This plant permitted data gathering for two behavior: normal and attacked. During 11 days, the scaled-down water treatment facility collected data from its initialization state, thus empty, to its final functional form. For seven days, the data collected were labeled as "normal" since there were no attacks. However, in practice, we observed something different regarding time distribution. Each of the two sections dividing the dataset between the "attacked" and "normal" behavior scenarios lasts a little bit more than five days.

The dataset provides 11 days of uninterrupted operation, including exactly 496800 seconds of normal operation and 449919 seconds of attack scenarios. The data comes from the 51 different features (each of them representing a sensor or actuator) present in the processing plant. The data is labeled according to whether the behavior is normal or abnormal (under attack). There are 41 possible attacks on the water treatment, but only 36 have physical impacts [6]. As our analysis aims to detect physical attacks, we are not interested in other types of attacks.

Logically, for any dataset analysis and model creation, it is necessary to make a feature selection. In our case, we will not make this selection, it is essential to keep all the features because any of them could be attacked. Removing them would increase the risk of missing an anomaly.

The data collected from iTrust is available to researchers worldwide, free of

charge. However, the dataset is not public; you have to apply for access using a registration form to access their drive, which you can find here [5]. Their drive is composed of thousands of log files on network traffic, but we are interested in the files associated with physical attacks. For this, three files interest us; SWaT_Dataset_Normal_v0.xlsx, SWaT_Dataset_Attack_v0.xlsx, and List_of_attacks_Finals.pdf. The three .xlsx files are the data on which we will do all of our analysis. The pdf lists all the attacks labeled in the dataset.

4.2 Clearing dataset

Cleaning the dataset is extremely important, we need a good quality dataset so the evaluation can be as correct as possible. The first thing done on the dataset is verifying its integrity. We have verified if the data was relevant and reliable. The next sub-sections each report an error that has been detected and corrected.

4.2.1 Typographical errors

One of the errors found is due to a typo in the dataset. The column 'Normal/Attack' is miss labeled, the word 'Attack' has been 37 times badly written as 'A ttack'. We also have rewritten the 'Timestamp' label as 'Timestamp'.

4.2.2 Duplicate Attack error

The second type of error comes from the list of attacks that the Itrust team has referenced. In the dataset SWaT_Dataset_Attack_v0.xlsx [5] provided, we observe an attack between the following period: 29/12/2015 6:30:00 PM and 29/12/2015 6:42:00 PM. However, the file reporting the list of attacks (List_of_attacks_Finals.pdf [5]) and the day and time they occur tells us of no attack for these values. We assume this error comes from a distraction concerning the different times of the day: PM & AM. During the period mentioned above, there are no attacks. On the other hand, on the same day at the same time, but in the morning (AM), attack number 21 is listed in table below 4.1:

	Attack 21	Attack not referenced
Start time	29/12/2015, 18:30:00	29/12/2015, 06:30:00
End time	29/12/2015, 18:42:00	29/12/2015, 06:42:00
Attack Point	MV-101, LIT-101	X

Table 4.1: Attack 21 and attack unknown from the dataset

4.2.3 Timestamp attack error

The next problem we encountered also concerns the referenced timestamp. The data for the 41st attack in the document `List_of_attacks_Finals.pdf` [5] shows a different timestamp. Attack 41 starts at the same time and day in the dataset and in the document (2/01/2015, 13:13:02) but ends 15 seconds later in the dataset (at 13:41:11 versus 13:40:56). We corrected the problem by adapting the dataset data according to the document because it is assumed that the document delivered by the team is the most reliable source of information.

4.2.4 Missed Value error

Finally, in the dataset `SWaT_Dataset_Attack_v0.xlsx` [5] at time 2015-12-31 21:01:58, there is a break of 82 seconds or no data has been recorded for all features. Here is the plot of the sensor FIT101 between 20:56 and 21:06.

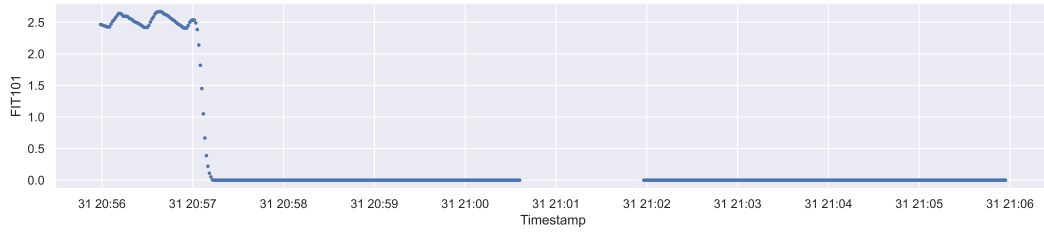


Figure 4.1: Plot of the FIT101's value showing the missing data points

4.2.5 Unused Data

We have removed the first four hours of the dataset to remove the process where the SWaT starts from an empty state to a stabilized one. This data is useless since the IDS is supposed to detect problems in the long run and, thus in a non-empty state.

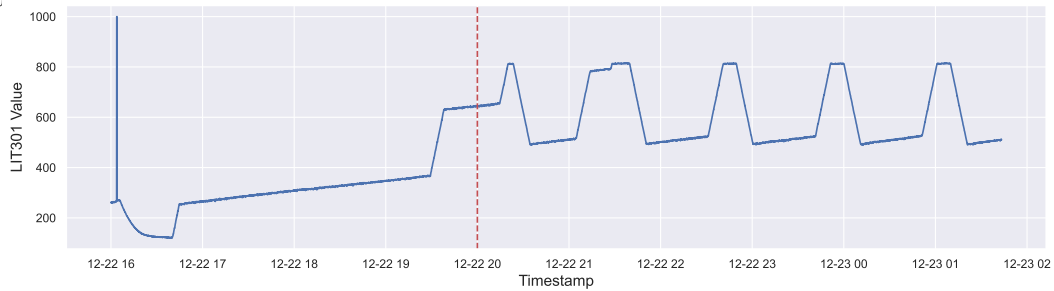


Figure 4.2: Plot of the LIT301's value showing the split of unused data

Figure 4.2 shows the LIT301 during the first few hours of SWaT operation, as can be seen, the tank fills during the first four hours. The dashed red line represents when the dataset was cut off exactly at 19:59:59 on 2015-12-22 when the system stabilized.

4.3 Transformation

This section explains the transformation applied to some features in order to have more readable data and extract the most relevant information.

4.3.1 Removing outliers

Some features contain outliers, reducing the readability and understanding of their data. It even decreases heavily the capacity of one of our models to detect anomalies: the Min-Max model. This also makes the task of identifying cyclic and acyclic features much harder. We will discuss the importance and need for such identification in section 4.4.1.

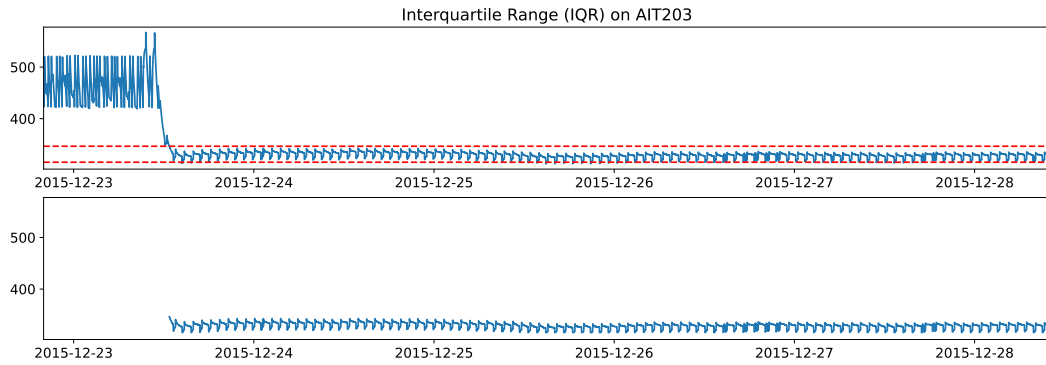


Figure 4.3: Removal of outliers on the feature AIT203

The two graphs above represent the AIT203 feature. Both charts have an abscissa in seconds and an ordinate showing the data value.

The first chart above shows the raw data. It has two horizontal red lines splitting the data between two quantiles: the upper and lower part represents 25% each, where the middle is 50% of the data. Indeed, the data is split between two quantiles: $Q1 = 25\%$ and $Q2 = 75\%$.

The second one, at the bottom of the Figure 4.3, shows the transformation on the AIT203. This transformation removed the outliers situated outside the range defined by the two quantiles of the IQR method. This method has been defined in the General Knowledge section 2.2.3 Interquartile Range.

4.3.2 Remove noise

One of the biggest problems when dealing with the dataset was reading the data efficiently while getting the highest amount of information. To have more readable data, we had to smooth it. The sliding median, moving median, and moving average are statistical methods that can be applied to a time series of data to smooth out short-term oscillations and detect longer-term patterns.

- **Sliding median** : This is a rolling window statistic that determines the median of the data over a predetermined period. The window advances one time period at a time, and the median is updated at each time point, hence the term "sliding".
- **Moving median**: This is in contrast to the sliding median, which shifts its window with each unit of time and where the median is recalculated over its entire window at each second.
What we call a "moving median" is a method where the window moves by its size, and the data points in the window are updated with a median computation on its data point. At a time t , the next calculation of the median will thus be done at $t + n$ where n is the size of the window.
- **Moving average**: This method is basically the same as the Sliding Median but with the use of the mean instead of the median.

Here is the result of our tests with a window (labeled as n) of 100 seconds on the AIT201 feature over a short period with the moving average, sliding window, and moving median.

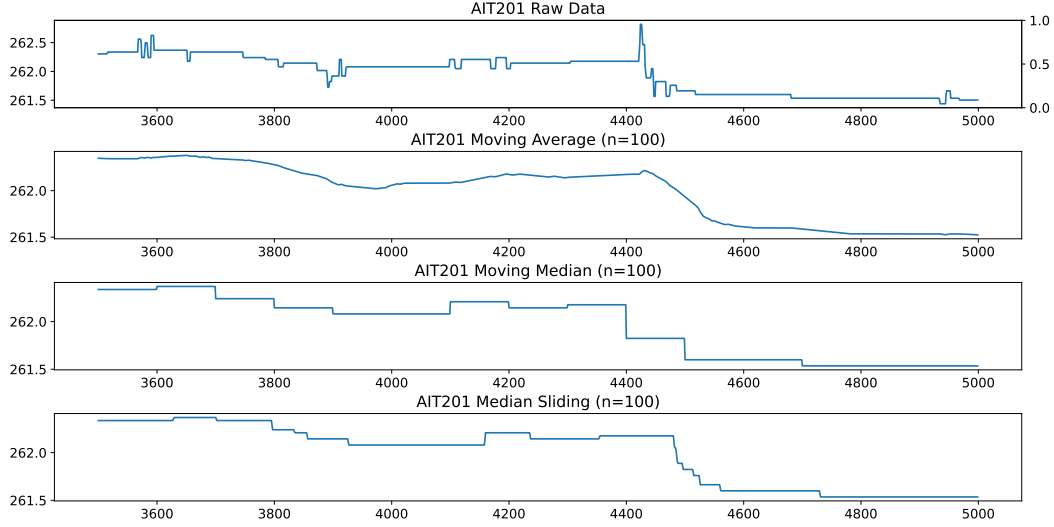


Figure 4.4: Moving Average, Sliding Median, and Moving Median

On the plots 4.4, the abscissa value expresses the time in seconds, the left ordinate is the value of the feature.

On each plot, the transformation of the feature AIT201 is represented in blue. Except for the first plot on top, where the feature AIT201 did not undergo a transformation.

The moving median and the moving average are often helpful for removing short-term swings from the data and spotting longer-term patterns. The quality of the data and the analysis's objectives will determine which option is best. If the data contains extreme values that could skew the results, the median may be a better option because it is more resistant to outliers [29]. If the data is mainly homogeneous and there are no extreme values, the mean may be preferable because it is more responsive to the data's values.

As we can see in Figure 4.4, the sliding median does not provide a result with sufficient noise removal for the data to be usable by our models, which we will see later in the 5.2 Models section.

To remove noise in our situation, we use the moving median to create a new feature that is much simpler and easy to understand than a continuous feature without this transformation, which may be very chaotic.

4.3.3 Slope Computation

Some continuous features need to give us more information to be usable, so we had to find something exploitable for this type of feature.

The previous section 4.3.2 Remove Noise is in direct link with the slope of the features. Indeed the transformation of the feature into a slope will use the moving median with a size window (labeled as n) of 100.

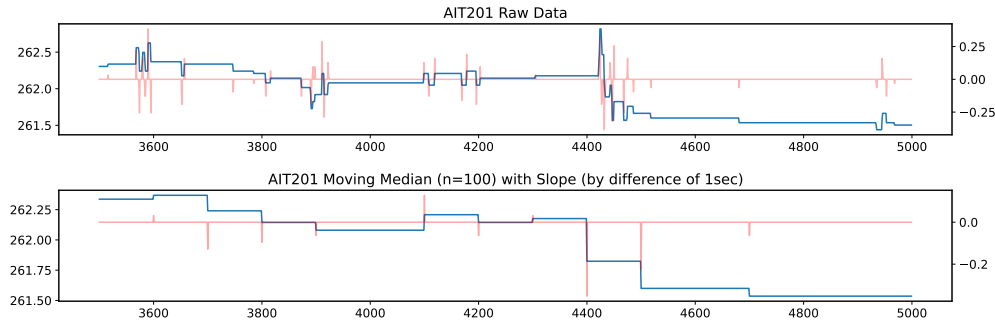


Figure 4.5: Moving Median with a window size of 100 on the feature AIT201

The Figure above 4.5 is a small part of the AIT201 feature. At the top, we have the base feature, and below the transformation of this feature using the moving median that we saw in the previous section 4.3.2 Remove Noise with a size window of 100. The red line on each plot represents the slope of the feature whose values are represented on the vertical axis on the right. This slope represents the difference between two data of a time $t - 1$ and t . The slope on the graph of the transformation using the moving median is not usable. As we can see, there are just some peaks every 100 seconds.

In order to be usable, we have to define a new type of slope. Here is the pseudo-code used to create this new type of slope :

```
# apply the transformation moving median with a window size 100 on the feature
med = moving_median(feature, n=100)

# initialization of array slope
slope = np.zeros(len(med))

# run from 100 seconds to the end of length of med
for t in [100: len(med)]:

    # run from 1 to 100
    for k in [1: 100]:

        # calculate the difference between med[t] and med[t-100]
        diff = med[t] - med[t-100]

        # assign difference from S[t] to S[t-99]
        S[t-k] = diff
```

Listing 4.1: Calculation of Slope

The interest of doing this manipulation with the moving median is to have a transformation including a value for the slope, avoiding being equal to zero most of the time. This process suppresses sudden slope peaks like on the bottom graph of Figure 4.5. The solution is to spread these changes over 100 seconds, as shown in Figure 4.6, forming a staircase slope.

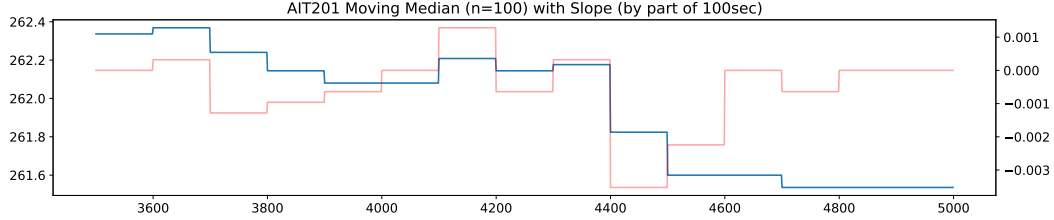


Figure 4.6: Moving Median with a window size of 100 on the feature AIT201

Indeed as we can notice in Figure 4.6, the difference by part of 100 seconds gives us a feature usable for the creation of State in the Markov Chain model 5.7.

4.4 Type of data

The dataset used for training is composed only of the normal phase without any anomalies. The SWaT is composed of different components with different behavior. Depending on the nature of the data, we apply different models. The data's nature depends on which sensors it comes from or if it comes from an actuator.

This will lead to the creation of 2 main categories in data: discrete from actuators and continuous from sensors. Each of these categories has its subdivision too.

4.4.1 Continuous data

The following sensors are composing the continuous feature: FIT101, LIT101, AIT201, AIT202, AIT203, FIT201, DPIT301, FIT301, LIT301, AIT401, AIT402, FIT401, LIT401, AIT501, AIT502, AIT503, AIT504, FIT501, FIT502, FIT503, FIT504, PIT501, PIT502, PIT503, FIT601.

Acyclic and Cyclic

The continuous data is split in either cyclic or acyclic. Cyclic values have a pattern repeated the whole time, contrary to the acyclic values that do not have cycles. We have to be able to differentiate them because we cannot deal with both of them the same way. Indeed, acyclic features are chaotic. Moreover, we cannot extract

the same amount of information as for the cyclic feature. This plays an important role for our models in Chapter 5 Models.

Acyclic features are easily recognizable visually, but it is more complicated to determine them automatically. To compute it, we use an auto-correlation system. The key component of this process is the LAG value which is the number of seconds from which the feature is delayed with itself.

The correlation is computed between the actual feature and its delayed value. It can be summarized with y as the value of a continuous feature, n the duration of the dataset, k the LAG value, and y_t the y in a time range $t[i; j]$ with i and j being the start and end of the range. The autocorrelation function is given by :

$$\text{Corr}(y_t[0; n - k], y_t[k; n]) \text{ with } k = 1, 2, \dots, n/2 \quad (4.1)$$

The computation takes :

1. The value of the feature in a range starting from the beginning ($= 0$) of the dataset to the value of the duration of the dataset minus the LAG ($= n - k$).
2. The value of the feature in a range starting from the LAG ($= k$) to the end of the dataset ($= n$).

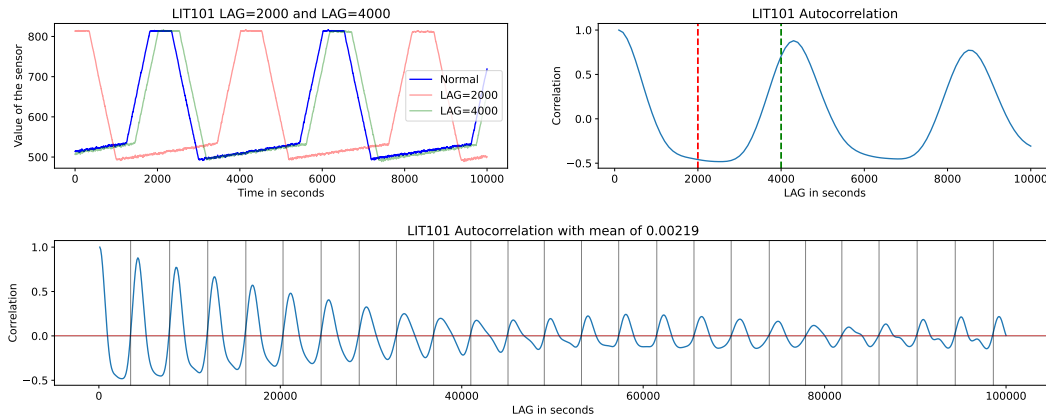


Figure 4.7: Auto-correlation of the LIT101 feature with two different LAG values and the result.

The process is illustrated above in Figure 4.7 in the upper-left corner with a chart having an abscissa labeled with the time in seconds and, on the ordinate, the value of the sensor. The three colors represent the value of the feature LIT101 depending of the LAG value; in blue LAG value=0, in red LAG value=2000 and in green LAG value=4000.

The correlation is computed thanks to the equation (4.1).

The result is shown in the upper right corner on a chart labeled on the y-axis with the correlation score and the lagging time in seconds on the x-axis. It shows the blue curve of autocorrelation of the LIT101. The green dotted vertical line represents the autocorrelation score of the feature LIT101 with a LAG=4000 and in red a LAG=2000. As we can see, the score for the LAG represented in green is far higher than for the LAG in red.

It is explained by looking in the upper left, the proximity and the similarity with the blue (no LAG) and green line. On the other hand, the blue and red lines are not similar at the same time. This chart only shows a small part of the dataset, but this behavior is repeated forever.

On the bottom of the Figure 4.7, we have a full view of the autocorrelation process for the LIT101 feature, with on the ordinate the correlation score and on the abscissa the value of the LAG. This chart is easily interpreted, we can see that with the first LAG value (=1), we have a very high correlation score because the feature is more or less correlated with itself. Then the more the LAG increases, the less there is autocorrelation until the correlation score goes up again, representing one period.

The chart shows this phenomenon with a vertical black line representing the separation between periods. While the horizontal red line shows the mean of the correlation value, and the horizontal black line is $y = 0$. In this case, we cannot see the horizontal black line because it is overlapped with the red one. This overlapping is how we detect cyclic value, but we need a threshold to split cyclic features from acyclic ones. We will discuss this threshold after the visualization of the summary of all autocorrelation plots in the next paragraph 4.8.

During the process of finding cyclicity, we take a LAG ranging from 0 to half the length of the training dataset. This increase of LAG value will change the mean of the autocorrelation curve and give a more relevant metric. This range (from 0 to half of the length of the training dataset) is essential to be sure to compare the first half of the time series with the second half of itself. This method gives us the autocorrelation of the two different time series belonging to the same time series.

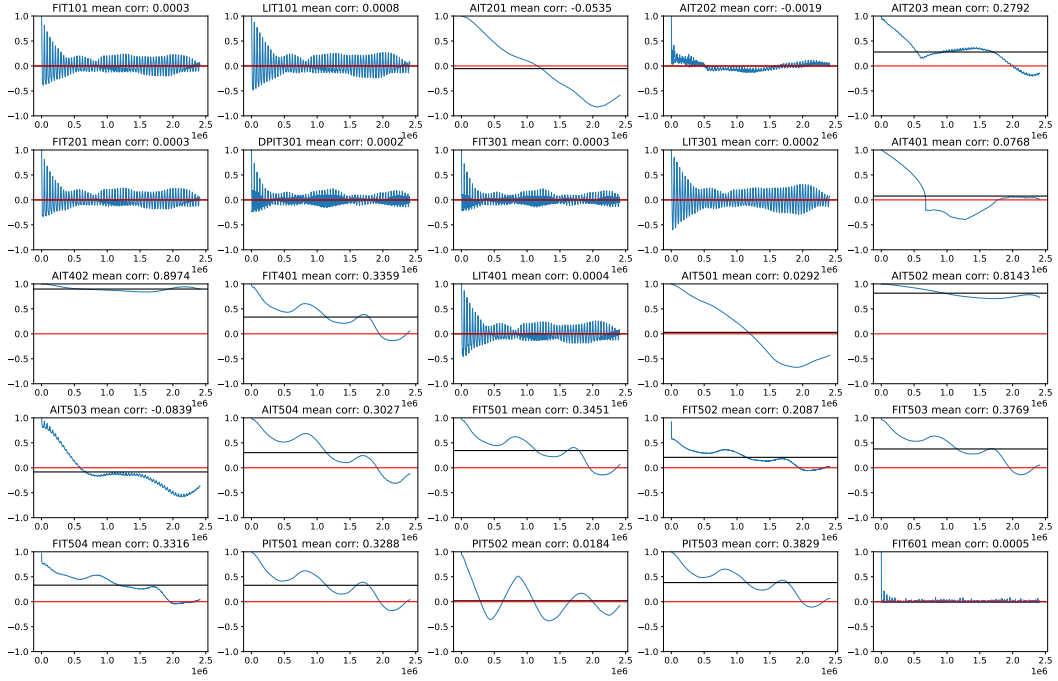


Figure 4.8: Autocorrelation summary for all continuous feature

In Figure 4.8 there is the summary of all continuous features with the same axes as described in Figure 4.7. We can even quickly identify cyclic and acyclic features by visually checking the curves on the autocorrelation plots.

We realized that below a threshold of 0.001 on the autocorrelation mean in absolute value, we can split the cyclic features from the acyclic ones. This is explained by the periods staying along the value zero of the autocorrelation mean, which implies a cyclicity. However, we could not set the threshold to 0 because of the noise embedded in the cyclic features, even if the noise is not strong. It is by using this threshold that we determine which feature is either acyclic or cyclic.

We can now say that our cyclic features are: FIT101, LIT101, FIT201, DPIT301, FIT301, LIT301, LIT401, FIT601. Some of these features have much more periods than others like FIT601. However, this does not imply a better correlation or a better mean correlation.

As explained before, it is essential to dissociate cyclic and acyclic features. By making the wrong association between the model and the data type, there is a risk of creating a large number of false positives or false negatives in the evaluation.

4.4.2 Discrete data

Discrete features also have their separation: constant and not constant. The constant values keep the same value during the whole train dataset. Unlike the previous case, not constant features vary during the whole process.

Variance and Mean on Discrete Features															
Mean	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	2.00	2.00	2.00	
Variance	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
	P206	P301	P204	P401	P202	P403	P404	P201	P102	P502	P601	P603	P501	UV401	P402
	Features														

Figure 4.9: Variance and Mean on discrete feature with a variance of zero

Figure 4.9 above shows the variance and mean of features with a variance equal to zero during the training phase. These features correspond to : P102, P201, P202, P204, P206, P301, P403, UV401, P401, P402, P404, P501, P502, P601, and P603, which we will discuss in the Variance Zero Model 5.3.

The averages of our features having a variance of zeros will be very useful for the continuation of our research because, as one can guess, these features are supposed not to change their value. It means that we must keep in memory the averages of these features to be able to detect if this case happens.

Variance and Mean on Discrete Features											
Mean	1.01	1.01	1.02	1.02	1.68	1.71	1.74	1.74	1.75	1.80	1.83
	Variance	0.01	0.01	0.03	0.04	0.22	0.21	0.20	0.19	0.19	0.17
MV301		P602	MV303	MV304	P205	MV101 Features	MV201	P203	P101	MV302	P302

Figure 4.10: Variance and Mean on discrete feature with a variance not constant

Figure 4.10 shows the features with a variance different from zero. These features are MV101, P101, MV201, P203, P205, MV301, P302, MV302, MV303, MV304 and P602.

Chapter 5

Models

5.1 Threat Model

Nowadays, attacks can have different goals using various techniques, which complicates the task of detection. When it comes to the SWaT infrastructure, we assume that hackers will inject false data 2.1 FDIA. It can interfere with performance, safety and halt production [30]. There are two types of possible modifications by hackers in SWaT :

- **Sensor Spoofing:** Hackers change the sensor readings in the ICS. This can be accomplished by physically changing the sensors or by using an injection in the ICS system to receive false readings.
- **Actuators Manipulation:** Hackers physically tamper with or change the settings of actuators in an ICS system. As a result, there may be problems with safety and outputs if actuators like pumps are not working properly.

Our IDS is based on the fact that a hacker can only modify the value of the sensors and actuators in the system. Furthermore, the hacker may not know the system and see it as a black box. We have other assumptions preventing hackers from doing anything they want.

We also assume that the system has a minimum of physical protection, that the software are frequently updated, and that we can trust the PLCs. So that the actuators react correctly to the sensor data and do not change the system rules. We assume there is a use of secure protocols such as HTTPS to ensure secure connections between the sensors/actuators, PLCs, Historian and the SCADA & HMI system in use. This avoids a change of data, we can think of a case where an attack is in progress on the sensors, but the data that is reported is modified so we cannot detect the attack. To avoid this kind of attack we also need the assurance that the Historian guarantees data integrity.

Another important aspect is the implementation of network segmentation to separate the control part from the rest of the company so that it is not accessible from outside. This allows us to be protected from the outside world and avoid, for example a DDOS. A DDOS may freeze the system or a part of the system, which would make it impossible to detect anomalies.

Therefore we can make the hypothesis of possible attacks on the SWaT. The goal of our models is to prevent as soon as possible any change that would appear abnormal.

The possibilities of attacks are multiple but the hacker is still limited by the hypothesis mentioned above. We can create a list of plausible attacks:

- Hackers can make slow or fast attacks. More precisely, they can increase the value of a sensor little by little until a certain threshold is reached to avoid being detected. However, this kind of behavior will be detected by one of our models, the Longest Sequence of Slope Direction model.
- When these attacks appear, it could be that a pump in the system breaks, if there is a backup pump for the faulty pump, it could take over. In this case of attacks, we can detect it using the Variance Zero model.
- As for the pumps, if a pump stays open or closed too long, it would probably be an attack, in these cases we have the LS Discrete model to deal with it.
- Another attack would be to increase the value of a sensor little by little or to set the value of a sensor to a certain amount beyond the normal limits that would be detected by the Min-Max model.
- The Markov Chain model can detect more attacks since its main function is to see abnormal events in the functioning of the SWaT system. As the model is also based on the slope of the features, it will be able to detect abrupt changes of the features and thus detect fast attacks.

Despite the detection of these attacks, there are still some anomalies that could also come from a cascade effect due to the malfunction of a sensor or a pump under attack. In these cases, we often have to deal with after effect and we have not found a solution to not detect this after effect as attacks (reminder of 2.2.5 After effect section).

5.2 Combination of models

This section explains the goals and the usage of the models used to predict if a certain behavior is an anomaly or not. Having multiple models allows to leverage behaviour from different types of data and analysis by combining them. Also, it fulfills the potential of the dataset by checking specific aspect of the data while remaining general and generic.

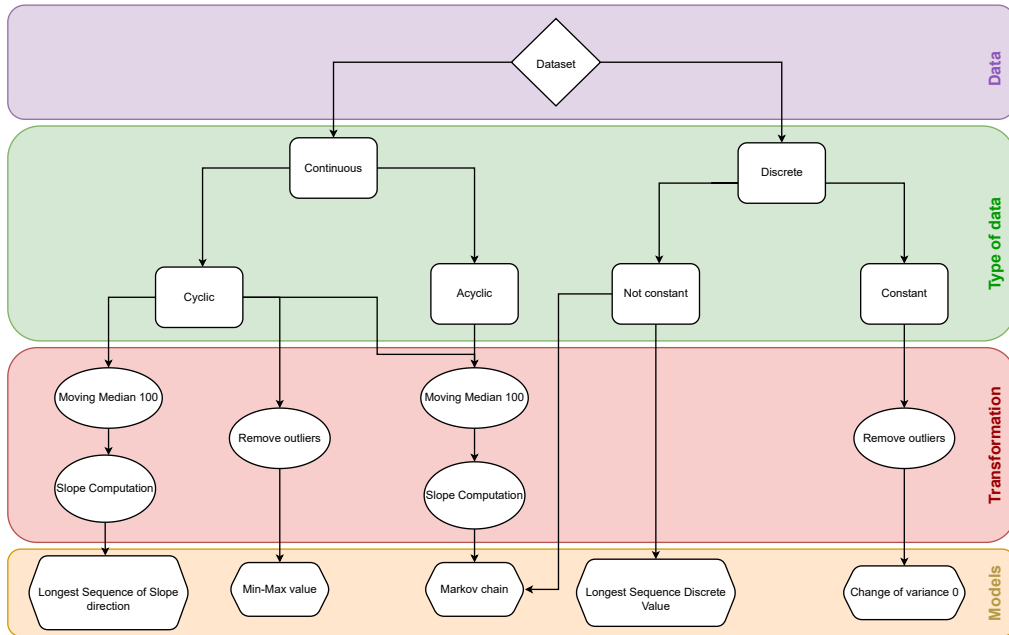


Figure 5.1: Summary of models

As we can see on the diagram, the data in the dataset is categorized according to the specifications of types of data explained in 4.4 Type of Data. Depending on those categories, different transformations are applied on the data for each model. Indeed, in some cases to apply the models, we have to modify the data in order to be able to extract understandable and readable information, all the transformations are explained in their respective sections in 4.3 Transformation.

Not all models are used on every type of data. Not only because the models are used to extract the most useful information based on the type of data they deal with but also to avoid the use of inappropriate data that would create a lot of FP or NP rather than TP or TN.

By capturing normal behavior, our models can detect which behavior is expected and which one isn't.

We have five models :

1. Variance Zero
2. Min-Max Values
3. Longest Sequence of Slope direction (Cyclic LS)
4. Longest Sequence Discrete Value (Discrete LS)
5. Markov Chain

Each of these models is used for a different purpose and not only to analyze data but the behavior of the SWaT components themselves.

Four of these five models are univariate, they analyze each feature independently from the others. The Markov chain is the one that is multivariate, it compounds data from each feature to analyze it as a whole.

Each of the following sections is dedicated to explaining one of the models.

5.3 Variance Zero

For this model, we focus on the potential fixed values in a system. When the system is in normal behavior its value does not change. It is thus easy to detect an anomaly when suddenly its value changes.

In the case of the SWaT it is straightforward to use this principle because some sensors are only used as a backup. This means their value will only change if the main sensor has a problem. It is therefore easy to determine when a problem occurs: we calculate all variables with a variance equal to zero, so we can define their average to be equal to their unique value. If this value changes, an anomaly is raised.

5.4 Min-Max Values

In the case of the Min-Max model, we can only use continuous variables.

The minimum and maximum values in the train set will be defined as the two extreme values between which the variables vary. However, for an acyclic feature that may increase or decrease over a long period of time, the values collected are not sufficient to determine if the future behavior is normal or abnormal. This statement is based on the hypothesis that acyclic features, due to their chaotic behavior, can't be predictable. This implies that the threshold values would be almost constantly exceeded, resulting in a large number of False Positives. Indeed,

exceeding the boundaries set by the Min-Max creates an anomaly that can last for a long period. This would not only create one anomaly but could lead to countless others.

Contrarily, cyclic variables vary from one value to another constantly in the short term as well as in the long run.

5.5 Longest Sequence of Slope Direction (Cyclic LS)

The computation of the longest slope sequences consists of calculating the longest sequence having the same slope direction (positive, negative, or neutral) in a row for each feature in the training set. When in the test set, a sequence is longer for one of the directions than what we observed before in the training set for the same slope direction, an anomaly is raised. This computation is done on the result of the moving median transformation applied on a feature.

Here is an example of longest sequence on the feature LIT401:

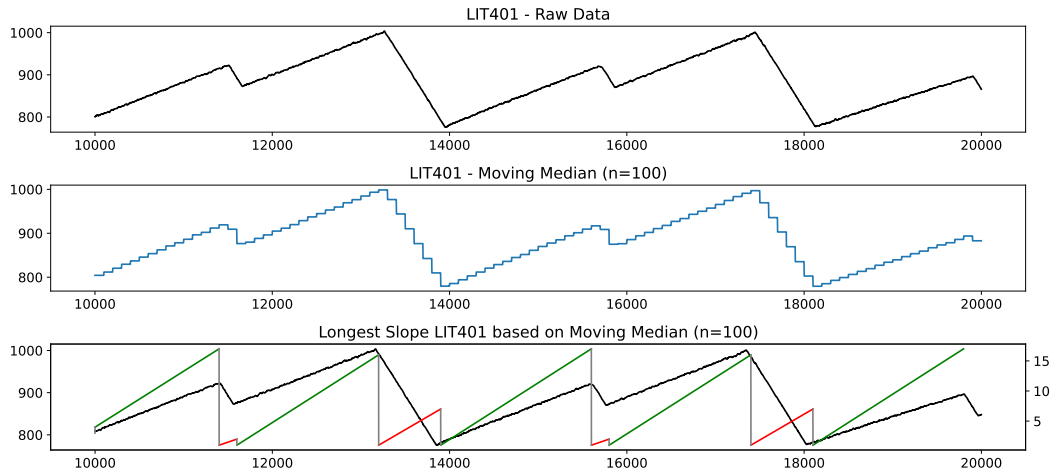


Figure 5.2: Longest slope sequences computation for the LIT401

These graphs above show respectively, the raw data of LIT401, its transformation using a moving median with a window of size 100, and the largest sequences of slope direction that the curve has without interruption. On the right ordinate axis of the last plot, there is an accumulator representing the maximum length of this sequence and on the abscissa, we have the time in seconds.

There are three types of curve representing the direction of the slope. In the figure

5.2 we can see only two of them, which are differentiated by their color, the green for the positive and the red for the negative slope.

As we said there are three curves, the missing one is for the neutral slope which is absent in this case, and when it happens it is usually very short.

With this model, we do not look at the intensity of the slope but the duration of the trends.

We can have a knowledge of the behavior of each feature and later, analyze it with the test set. This makes it possible to detect slow attacks but that occur over time by modifying the behavior of one of the features.

This model is only used on continuous features with cyclic behavior. The reason why we do not use it on acyclic features is that their chaotic nature does not allow to use correctly the specification raised by the computation of the length of a trend. Indeed, the acyclic features can have very different behavior from one moment to another and, therefore possibly from the training set to the test set. Calculating the Cyclic LS in a more or less stable train set makes no sense if then, the data only goes down. In this case, the model would permanently detect an anomaly.

In order to be able to use this model on cyclic features we still need to apply a transformation on the raw data: the moving median transformation 4.3.2. The use of the moving median is essential. Indeed, we are in the presence of continuous features which means that there is noise. Because of this noise, there is a high probability that we compute a slope between two points (between a time t and $t - 1$) that breaks the trends because the slope calculated would be in the opposite direction. This would lead to an unfair break in the trends. For example, if there is an unwanted value smaller than the previous one when we are in the middle of an ascending slope.

This is why we use the moving median in the calculation of the longest slope, it allows us to recover the information without noise and therefore guarantees that we calculate the total length of each slope until the end of the trend.

The usage of a sliding median was an option but as expected, the sliding median does not remove enough noise which leads to poor results.

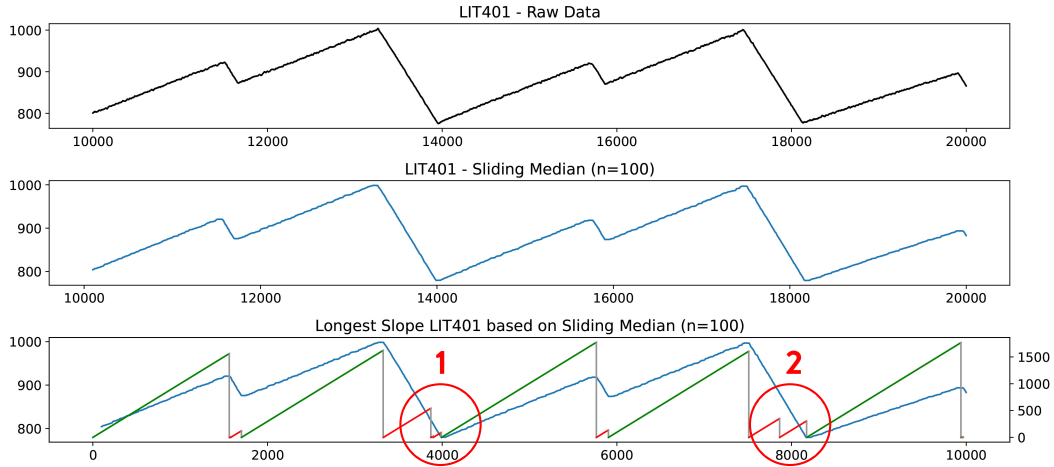


Figure 5.3: Longest Sequence of Slope Direction computation with a sliding median for LIT401

As we can see on the Figure 5.3 we have an example of what the sliding median does. In this example, we used a window with a size equal to 100. The first plot shows the raw data of the feature. The second plot shows the feature LIT401 after the transformation of the sliding median applied. A closer look at this second plot shows that the curve is not smooth and undergoes many small variations. This is the behavior we want to avoid in order to use our model in the best way. On the last plot we see the effect of the noise which persists after the transformation. We see the accumulator of the size of the sequences according to their ascending, descending or neutral trend in their respective color. The green part corresponds to the rising sequences, in red the falling sequences. In this case, we can observe two places where the descending sequences have been split in several parts instead of forming one long sequence (indicated by a circle and labeled as "1" and "2"). This has a big impact on the model as it does not reflect the real behavior of the feature.

We tried to apply this sliding window technique by changing the size of the window. When we increase its size, the curves are much smoother and softer, which distorts the trends and makes the result no longer consistent with reality. When we decrease its size there are even more small variations making the result less accurate.

5.5.1 Moving Median Window

As we have just seen, the moving median is the best transformation we can apply, it avoids the persistent noise we get with a sliding median. However, we need to define the best window size that will be used. As a reminder, the maximum length encountered on the moving median for the up, down and neutral slopes are kept in

memory for comparison. These maximum values for each type of slope direction vary according to the window size used by the moving median. This is due, as with the sliding median, to the inefficiency of removing noise from the original feature when the window is too small or, conversely, when the window is too large, and thus the coherence of the original feature is lost.

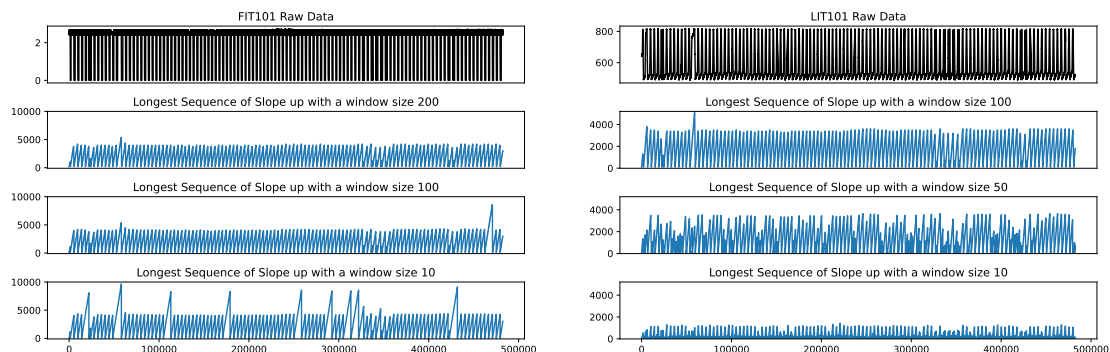


Figure 5.4: Longest Sequence of Slope ascending for the feature FIT101 and LIT101

This figure 5.4 shows us for the features FIT101 and LIT101 all the length of the upward slope sequence for the train set. The first plot above shows us the basic features without transformation with their value as ordinate. The abscissa is the same for all the plots of the figure, it represents the timestamp of the training set. The three plots below each of the basic features represent the upward slope calculated on the moving median with different window sizes. The plots of the feature LIT101 show that the difference between the length of the slopes loses coherence when the window becomes small. Indeed, we expect the slope lengths for the feature LIT101 to be the same all along the train set because of its cyclic property. We notice that when the window shrinks, the homogeneity of the length is not respected anymore if we refer to the ordinate, which means a loss of coherence.

The FIT101 feature does not lose its coherence but it loses precision. Indeed, the smaller the window becomes the more errors there are in the slope length. If we take a closer look, we can see that the length of the ascending slopes increases abnormally at some time when the window decreases.

This analysis applies to all cyclic features and is not specific to FIT101 and LIT101.

In summary, the smaller the window, the less noise will be suppressed, and the more we will lose either coherence or precision depending on the shape of the

cyclic feature. Nevertheless, we have to choose the size window showing the most stability on all the features but also having the smallest size possible. The bigger the window size, the more it will affect the anomaly detection time. Indeed we will have to wait for the slope to be computed for a window of size n and therefore wait n seconds to be able to announce an anomaly. The moving median of size 100 is therefore a very good choice to keep stability, consistency, accuracy and detection speed.

5.6 Longest Sequence Discrete Value (Discrete LS)

The idea of Longest Sequence for Discrete values is the same as explained in the 5.5 section. The goal is to understand the behavior and evolution of features' data. In this case we only deal with discrete value, we still have accumulators but they increase when there is a sequence of the same value without interruption.

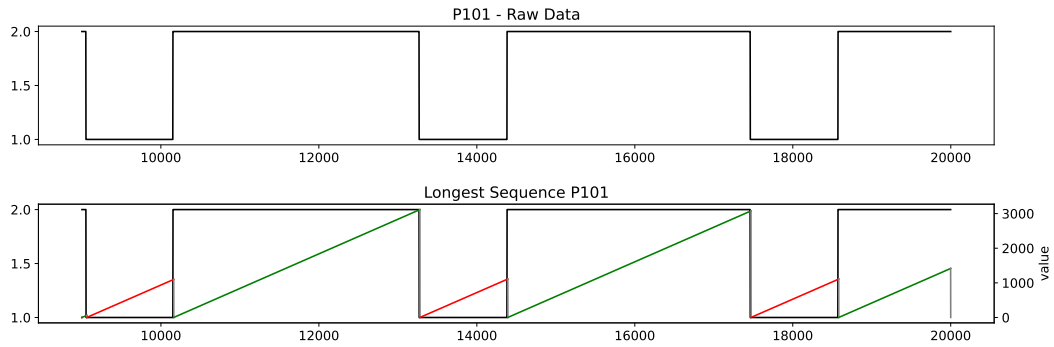


Figure 5.5: Longest Sequence Discrete Value on feature P101

The figure 5.5 shows us what happens graphically when we compute the longest sequence for discrete features. The example taken above is the pump P101. As we can see, the pump has only two values: 1 or 2. For each of these two values, there is an accumulator for the number of seconds where the values do not change in the training set. The maximum length for each of these values is kept in memory. These values are defined as limits that must not be exceeded in the test set, otherwise an anomaly is reported.

5.7 Markov Chain

5.7.1 Introduction

Markov chains are used to calculate the probability of an event occurring by considering it has a state transitioning to another state or to itself [31]. It is a stochastic model describing a sequence of states (possible events) where all the useful information for prediction is contained in the present state [32].

5.7.2 Illustration and definitions

In our case, we want to calculate the probability of an anomaly occurring based on states from our features data.

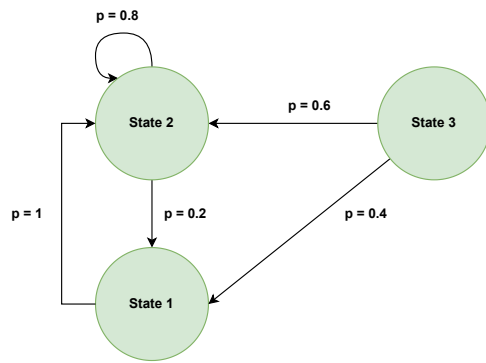


Figure 5.6: Markov Chain

$$T = \begin{pmatrix} 0 & 1 & 0 \\ 0.2 & 0.8 & 0 \\ 0.4 & 0.6 & 0 \end{pmatrix}$$

$$\pi = (0.33 \quad 0.33 \quad 0.33)$$

Figure 5.7: Transition matrix and stationary vector

In the figure 5.6 above, we have an example of a Markov Chain with three states. These three states are linked by incoming and outgoing links, labeled by a probability p . This percentage p corresponds to the probability of going from state A to state B or back to state A.

In figure 5.7, the probabilities of transitioning between two states are represented in the form of a square matrix T , called the transition matrix, with a size defined by the number of states. The coefficient T_{ij} is the probability of transition from state i to state j [32]. We can build the Markov Chain from this transition matrix T and vice versa.

The transition matrix has the following properties:

- All coefficients are between 0 and 1.

- For each row, the sum of the coefficients is 1.

The stationary vector π is also present, it represents the probability distribution that remains unchanged as time progresses in the Markov Chain. It is an invariant that satisfies the following property [33] :

$$\pi = \pi * T \quad (5.1)$$

Computation of probabilities

To calculate the transition matrix, we compute the chance from a certain state A to go to another state B: we compute each time A visits B and divide it by the number of outgoing links from A.

The stationary matrix is computed by dividing the sum of each appearance of each state by the total number of states.

The transition matrix allows knowing the possible links between the various states and the probabilities that they are triggered. The stationary vector is used to know the probability that a state appears.

Sequence of states transition

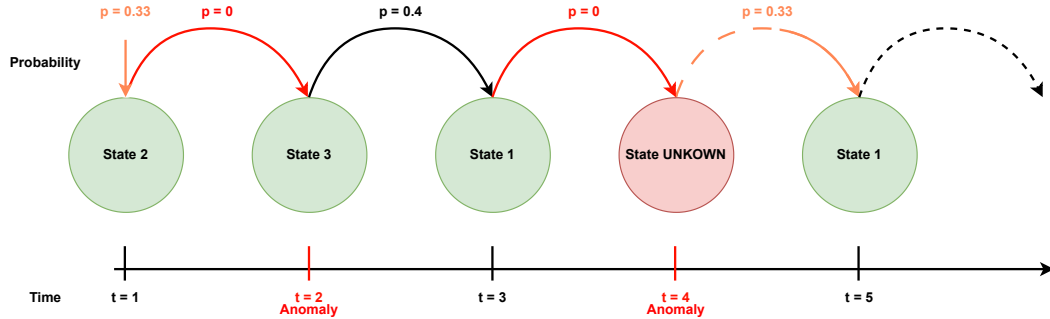


Figure 5.8: Diagram of states transition during the test set in Markov chain model

In figure 5.8, states are represented with circles. The probabilities for the next states to happen are at the top of the figure labeled with the letter p and below, the time t in seconds. It shows the process of finding anomalies in the test set where attacks may occur.

At each second, a new state is created from data of all features (this process is explained in the section 5.7.3 Creation of states). Each state represents the general

state of the SWaT process at a time t with a transition probability p from time $t - 1$.

During the anomaly detection phase of the test set, our model will calculate the probability of each consecutive state. Some transitions are not possible and will trigger an anomaly detection, there are two types:

- Irrelevant state : The probability p of transition between two known states at time $t - 1$ and t must be greater than 0 in the transition matrix (see 5.7) to not be identified as an anomaly. Otherwise, it means this state has no link to the next state. We can see this phenomenon on the figure above, from state 2 to state 3 by going from time $t = 1$ to $t = 2$, triggering an anomaly by our model at $t = 2$.
- Unknown state : If a state is unknown to our states present in the memory from the training set, it implies an anomaly detection as it is the case for state 1 towards the unknown state at time $t = 4$ on the figure 5.8.

The state following an unknown state will be assigned to its probability present in the stationary vector (see 5.7) as in the case for state 1 at time $t = 5$. The same thing happens for the first state of the process as illustrated in figure 5.8 at $t = 1$.

5.7.3 Creation of states

During the creation of the states we use several techniques to capture efficiently data and behavior from every feature in order to create an abstraction of the system behavior at a time t .

The states will allow us to differentiate the different possible states of a feature in order to understand its behavior. However, we need only discrete data to create our states. Indeed, there will be too many possibilities if we concatenate all the continuous features. So we need a method to render continuous features into discrete ones.

To do so, we split the process of making states depending on the type of data: cyclic, acyclic and not constant discrete data.

There is a difference between the state of the system and the state of a feature. From now on, the state of the feature will be called a **status**. A **state** is thus the concatenation of all features **status** at a time t .

Discrete data

Discrete data do not need an elaborate system to create status since each discrete feature contains only a few values. Thus we use their original value to create a status.

However, it is not so simple, as we have seen in the Variance Zero model, we have a model that checks if the features having a variance of 0 do not change their value in the test set. As our states are created partially from these discrete features, the slightest variation in the value of a feature with a variance of zero in the train will irrevocably create a new state (which will be unknown) when it has a value different from its average. This means that the state containing this value will necessarily be different from anything else we have seen in the train set, resulting in an anomaly detection by the Variance Zero Model. Therefore, it is not interesting to include features with a variance of zero in the discrete features contained in the Markov model.

If we keep only the not constant discrete variables and we create our states only on these status variables, we will find ourselves with only 284 possible states, it means that during the training phase of 482,401 seconds, on the 11 discrete features not having a variance of zero, there are 284 possible combinations exist to classify the discrete features.

This principle of combination will be important for the continuation of the creation of the Markov Chain model because, as we will see for the continuous features, the number of combinations can increase very quickly.

Continuous data

To create our status for the continuous features, we must use the same method for the cyclic features and acyclic features to have consistent status.

Indeed, we need the same method to have the minimum number of states. It implies the smallest possible number of combinations for each type of continuous feature. To do so, we need a low number of states with a probability equal to 0 in the transition matrix. This necessity comes from the fact that if we create many states from the training dataset, it is very likely that we will have too many states in the testing phase, which would be really problematic.

This leads to two problems:

1. The states would not be specific enough, resulting in a lack of common states with the test phase. This means that too many states would be discovered in the test phase. These unknown states would cause anomalies which in most cases would probably be a False Positive.
2. A high number of states involves too many zeros in the transition matrix. Indeed, when there are many states, it means that there are many possible states in which the system can be. In this situation, it is more likely that a particular transition has a probability equal to zero, as there are more

possible options for the system to transit. This situation would involve a lot of anomaly detection and FP.

It is thus essential to have a method of creation of status on a solid and reliable base to be able to claim a minimal number of state and thus minimize the number of False Positives in the phase of test. To do this we will use the slope defined in the section 4.3.3 Slope Computation.

The risk of creating too many states was our biggest problem in this phase of the implementation, but we had to keep in mind that too few states is also bad. Indeed, if we create a too small number of states, they will be too general and will not capture the real behavior of the ICS. By being too general, we will miss behaviours that are out of the ordinary and should be detected and classified as anomalies. To summarize, in order to avoid creating False Negatives, we had to avoid being too general but to avoid False Positives, we should not be strict and restrictive.

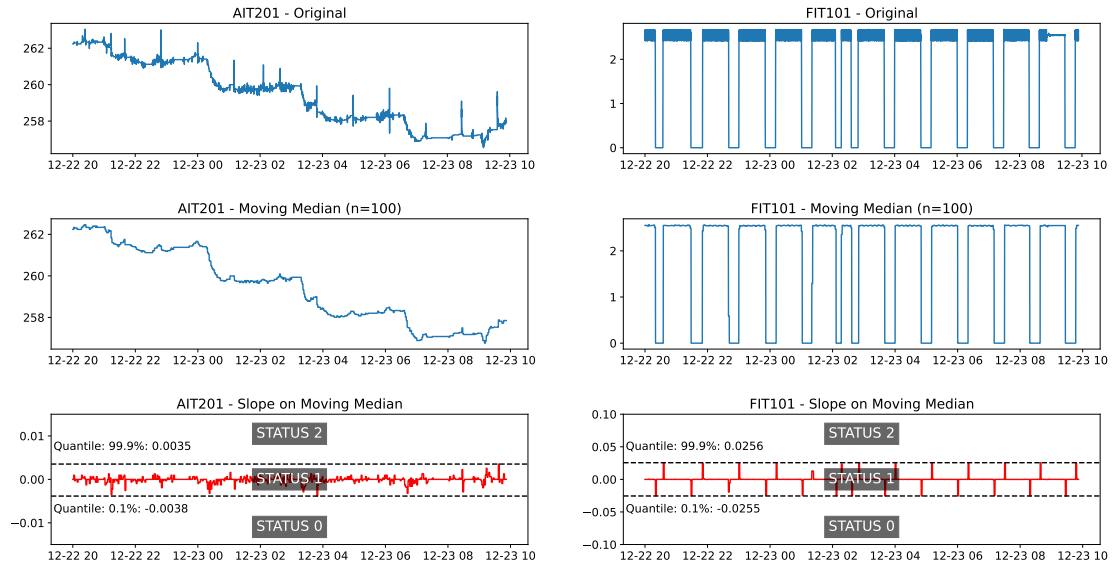


Figure 5.9: Creation of status on continuous features : AIT201 and FIT101

Figure 5.9 contains three plots for each of the features AIT201 and FIT101, the aim is to study the characteristics of a cyclic (FIT101) and an acyclic (AIT201) feature. The first plot at the top shows the raw feature without modification. The plots in the middle contain the features that have been transformed using the moving median method with a window size of 100 seconds. The window size is chosen according to the relevance of the analysis in the section 5.5.1. At the

bottom, the plots have undergone the transformation of the moving median into a slope as described in the section 4.3.3 Slope. This slope has, as we can see, some spikes showing us the steepest parts of the feature in order to detect big changes. Our new features status for each continuous feature will be created in relation to these spikes in the slope. We want our states to be able to detect these extreme slopes' spikes so we need to choose a certain threshold above and below which the feature changes its status. These status are represented in Figure 5.9 between and outside the two black dotted lines representing the thresholds equivalent to our quantile 0.1% as explained in the section 2.2.2 Quantiles. We will see in the next section the reason for the choice of these quantiles' values.

5.7.4 Feature combination

The choice of quantiles values in the creation of status is very important. Indeed, the creation of states depends on the creation of status and thus the choice of the quantiles values.

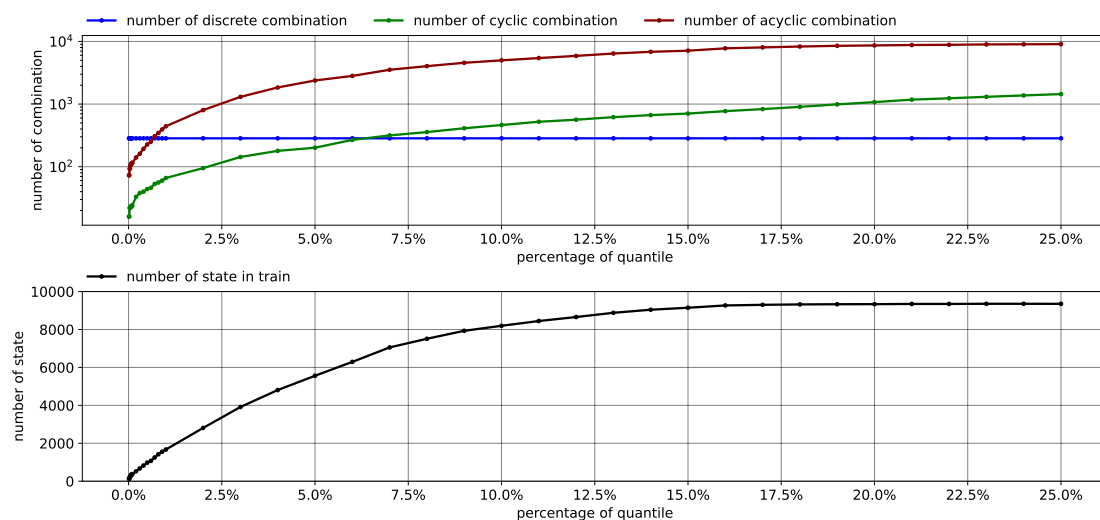


Figure 5.10: Number of combinations and states depending of the quantiles values

In figure 5.10 above, the effect of quantiles is shown on the number of states and the count of combination of states for each type of data: discrete (not constant), cyclic and acyclic in the training set. The first chart has as ordinate the number of state combinations in a log scale with a different color for each type of data. The second chart has as ordinate a number of states. The abscissa is the percentage to

which the quantiles are associated for the creation of the status on the slope of the continuous features, as explain in the section 2.2.2 Quantiles.

The purpose of the two plots is to see the exponential number of combinations linked to the number of states created in the train set. In the first chart above, we have a blue line that represents the number of combinations of states only for the discrete variables. The number of state combinations for the discrete features, which is 284 combinations, will remain constant since it is not affected by the quantiles.

This is obviously more complex for the continuous features, if we look at the green curve which corresponds to the number of combinations on the states for only cyclic features' status during the training phase. We see that the number of combinations increases with the rise of the quantile value. This is due to the fact that high percentage of quantiles (less than 50% obviously) will create status that are less strict for the extreme values, which leads to the creation of a larger number of possible combinations.

The red line is the count of the combination of states for the acyclic features. The slope is similar as the one for the combination of cyclic features but the difference is in the number of combination. Indeed, we have no more than one thousand possible combinations for cyclic features but more than ten thousand when the percentage on the quantile is equal to 25% according to our definition 2.2.2. It is important to note this growing number of combinations because each group of discrete, cyclic and acyclic features put together will necessarily create even more combinations.

The black curve shows us in the second plot precisely the number of states created according to the quantiles chosen in the creation of status for continuous features. As we can see, we easily have more than 8000 states created in the train set. Here is the importance of quantiles in the creation of the status, the smaller the quantiles will be and strict, the fewer there will be states.

As we want a strict model to limit the number of False Positives we will accept a very small percentage of the data points of the steepest slopes to create our status. As the percentage will be small we assume that the number of False Positives will be small. Indeed, if we create our status on a percentage of 0.1%, this would be equivalent to taking 0.2% of the points that are in the steepest ascending/descending slopes to create our status. In this case, 99.8% of the statuses created will be equivalent to 1 and therefore 0.1% of the train set will have status 0 and 0.1% for status 2. This labeling (status 0, 1 and 2) corresponds to the identification of the different possible status of a same feature as shown at the

bottom of the figure 5.9.

We can already tell ourselves that there won't be many status equivalent to 0 and 2 because they are reserved for extreme values. There are therefore not many different states because, as a reminder, states are the concatenation of the status of all features (except constant ones) at a time t .

5.7.5 Quantile Choice

The number of combinations must be minimized as much as possible, as we have seen in detail in section 5.7.4 and which therefore leads to a minimization of the number of states.

The method of creation of our states is quite general while being precise enough to capture the system's behavior. This implies that we meet most of the state's possibilities at the beginning of the training set. We therefore often come across these same states during the train phase, the more time passes and the less it will be possible to discover new states never encountered. This is graphically translated by a logarithmic curve representing the new states discovered as a function of time.

On the contrary, if the discovery of new states were done in a linear way in the train phase, it would mean that we don't have a good method of state creation. With such a linear curve, the evolution of time does not lead to a decrease in the number of discovered states, which is what we want to avoid.

We hypothesize that the consequence of this logarithmic curve will have an effect on the performance in the test set. The logarithmic curve implies that the vast majority of states are discovered in the training set. Therefore, the attack phases in the test set would contain only undiscovered states, which would result in the detection of abnormal states.

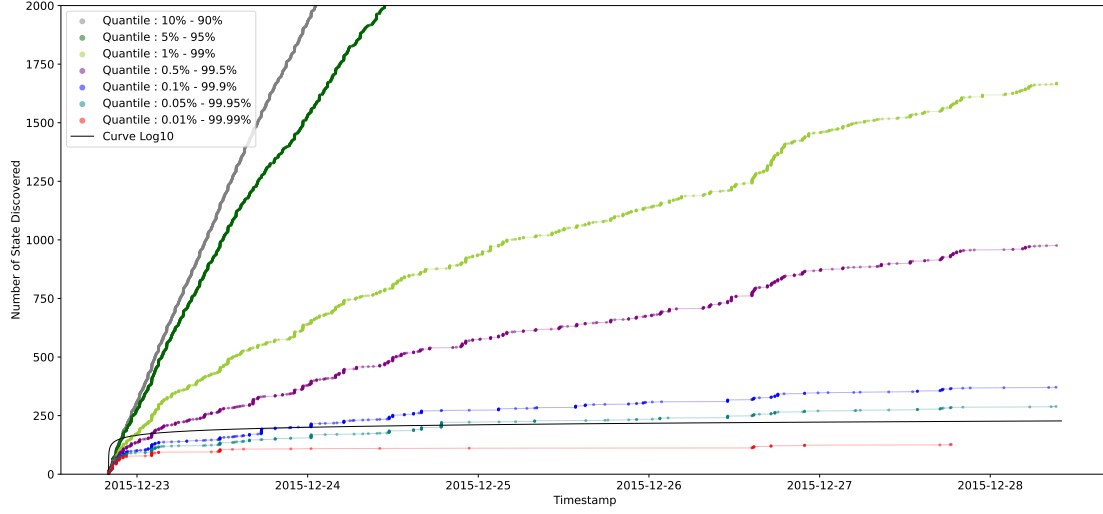


Figure 5.11: Number of discovered states according to the time per quantile

The figure 5.11 represents for each of the colours the number of states discovered over time with different quantiles percentages for the creation of status in line with our quantile definition as explained in the section 2.2.2 Quantiles. The legend gives the percentage corresponding to each line of points. On the y-axis, we have the number of states discovered up to a time t given by the x-axis during the whole training phase.

As can be seen, the lower the percentage, the more the point curve has a logarithmic shape that can be compared with a real logarithmic curve (indicated in black on the graph).

What is important to note in figure 5.11 is that the variation of the curve is extremely changeable as we test extremely low quantiles. Thanks to this graph we know that we must choose a curve with a percentage between 1) 0.01% - 99.99% (red dots) and 2) 1.00% - 99.00% (green dots) because it is between these two logarithmic curve that we are looking to discover all the possible states found in the training set. The first proposition 0.01% and 99.99% quantile in red is not a good solution because of the lack of discovered states even if its curve is very logarithmic. This problem was mentioned in 5.7.3 and indicates that too few states result in a lack of precision in the abstraction of the system by the status.

The same goes for the second choice with the green curve of the 1% - 99% quantile, which calculates too many states and never stabilizes, which is a problem because it will probably have the same behavior in the test phase and thus have no

states in common with the training phase. This will lead to many False Positives. The best value to have neither too much nor too little state is to take the quantiles 0.1% - 99.9%.

5.7.6 Number of states

As explained above, it is important to check if our states are discovered in a logarithmic curve. With our quantiles set to 0.1% - 99.9% we can now analyze our actual states found in the training set.

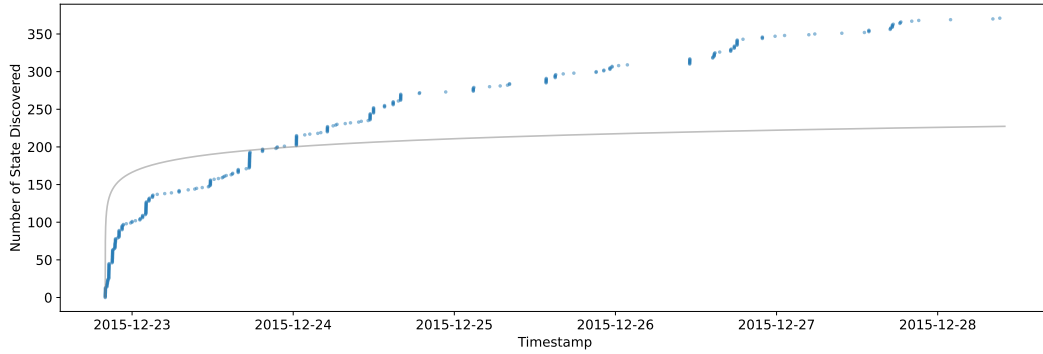


Figure 5.12: Count of state in time

The blue points show the number of states discovered as a function of time, as we can see, this curve is logarithmic at the beginning which means that we are in the presence of a state creation method consistent with the system. The fact that the number of states does not remain logarithmic after a certain time does not indicate a wrong method of creation. This is simply due to the discovery of new states because, as a reminder, states are the concatenation of 42 variables and it is obviously impossible to foresee the slightest chaotic variation of one feature or another, especially those concerning acyclic features.

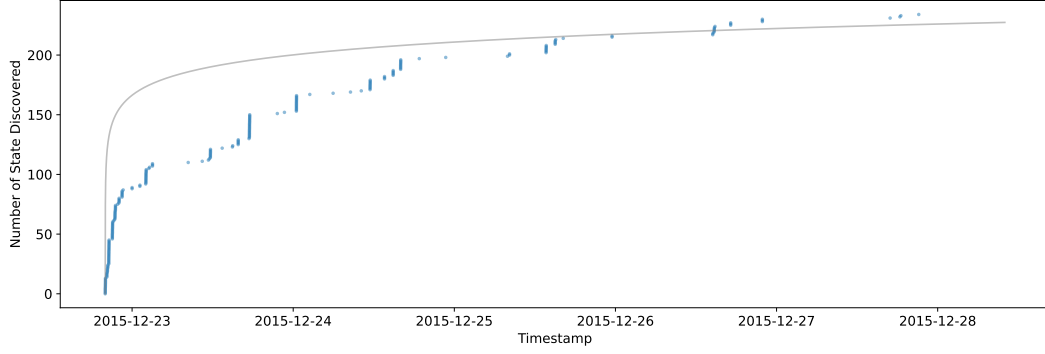


Figure 5.13: Count of state in time

As we can see on the Figure 5.13, this is the number of states discovered as a function of time when only the discrete not constant, and cyclic features are included in the states. This gives us exactly what we would have wanted in terms of a logarithmic curve which allows us to confirm that some states are inadvertently created based on the chaotic aspect of the acyclic features that even the slope could not make coherent.

Chapter 6

Evaluation

6.1 Introduction

In this section, we will present the performance of our models from several angles. Our work aims to implement several models based on simple and robust mathematical principles applicable to any IDS with any type of data.

We begin our analysis with an individual evaluation of each model. All models have strengths and weaknesses that are highlighted and explained. We support our statements with confusion matrices and the impact of each feature on those models. Then, we evaluate the performance of a multimodel based on all the models but also with a selection of models. This allows us to highlight more robust models than others and to find the best combination models.

We also base our analysis on a central point of our IDS, the detection time. In addition to this important point, as we want a versatile and resilient IDS, we evaluate our IDS against the criteria listed in the section 2.1 and introduced by Porras and Valdes [9]. These criteria are: accuracy, completeness, fault tolerance, timeliness, and performance.

We also discuss the types of attacks found in the test set and end this section with the limitations of our models.

6.1.1 Dataset Test

We have already mentioned the testing dataset in section 4.2 but in this specific section, we will take a deeper look at the details. The testing dataset contains almost half of the entire dataset, exactly 449919 seconds which starts on December 28, 2015, at 10:00:00 until the end of the dataset, January 2, 2016, at 14:59:59. Over

this time interval, there are 36 attacks which cover 54621 seconds which represents 12% of the test dataset. The attacks are of different lengths ranging from 101 seconds to 34209 seconds. Of these 36 attacks, the Itrust team indicates that there are some attacks that have failed. This is particularly the case of attack number 4 where we can read in the document `List_of_attacks_Finals.pdf` (mentioned in 4.1) the sentence "No impact". Attack number 4 targets MV504, but the MV504 pump is not present in the dataset provided by the Itrust team, so it is impossible to predict any anomaly with this attack.

This is not the only attack that has failed, here is the list of attacks that had no physical impact on the features with the `unexpected_outcome` comment of the iTrust team:

- Attack 4: No Impact
- Attack 13: UF Startup did not stop because MV-304 was closed late
- Attack 14: Attack Failed because Startup sequence did not start because Tank 301 was already full
- Attack 19: RO did not shutdown; Water did not go to drain
- Attack 20: RO did not shutdown; Water did not go to drain
- Attack 24: Not much impact made due to closure of P-101 because tank LIT-101 became full
- Attack 29: The three dosing pump did not start because of some mechanical interlock

There are therefore 7 attacks that have failed, which will have an impact on our confusion matrices being considered as real attacks, which will affect the accuracy of our models. Nevertheless, we will make the evaluation considering that these attacks exist because they are labeled as such in the dataset by the iTrust team. On the other hand, we will highlight some numbers which will reflect the power of our model in the non-detection of failed attacks.

6.1.2 Type of attack

Our dataset includes attacks labeled by iTrust team as four types in their .pdf document 4.1. We define them below :

- Single Stage Single Point Attack (SSSP): These attacks only require a single point of entry and use a single step or phase. A single sensor reading manipulated data could illustrate an SSSP attack.

- Single Stage Multi Point Attack (SSMP): These are single-step attacks that make use of multiple points of entry. An example of attack would be a coordinated assault on several sensors or actuators inside the SWaT, in which the attacker modifies the values being supplied by the sensors.
- Multi Stage Single Point Attack (MSSP): These are attacks that use a single point of entry but have multiple steps or stages. This type of attack has as input a pump/valve or a sensor that the hacker will target several times over time.
- Multi Stage Multi Point Attack (MSMP): These are attacks that use multiple points of entry and involve multiple steps or stages. A MSMP attack might take the form of a coordinated attack on the systems of the water treatment facility in which the attacker changes multiple times the value of multiple sensors or pumps/valves.

Out of the 36 attacks, here is the distribution of the number of attacks according to the type of attacks we have just described :

- 23 attacks of type SSSP (5 failed)
- 6 attacks of type SSMP (2 failed)
- 4 attacks of type MSSP
- 3 attacks of type MSMP

Each type of attack has a different probability of being detected by our models. Moreover, the attacked features themselves have a different impact in the detection of anomalies, either by their discrete or continuous aspect in addition to their cyclicity.

6.1.3 After effect evaluation

Our models detect a lot of after effects, which we have already mentioned in the section 2.2.5 After Effect. Here is a concrete example based on the Min-Max model:

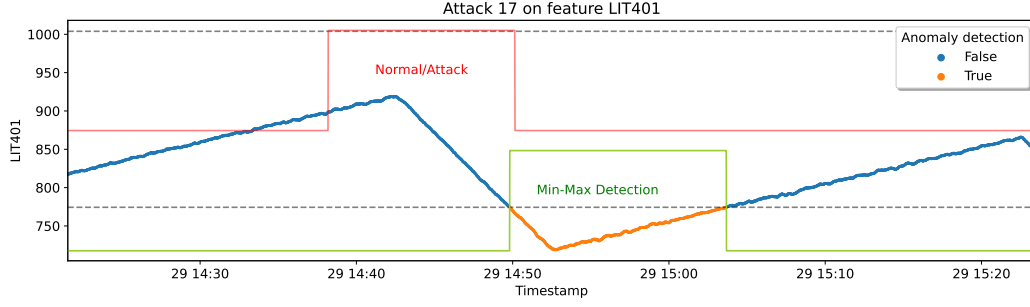


Figure 6.1: Attack detected on feature LIT401

In this graph, we have plotted the LIT401 feature (a tank of water) on the y-axis with its value and the x-axis is the timestamp.

The red line represents the attack number 17 labeled in a binary way, when the line goes up it corresponds to an attack. The green line represents our Min-Max model also in a binary way, when the line is up it corresponds to an anomaly detection by our model.

As we can see, the Min-Max model detects an anomaly on the LIT401 feature because it undergoes the black dotted lines, which are its minimum-maximum thresholds calculated from the training dataset. This exceeding of the threshold is shown by the change of color from blue to orange. This anomaly detection is actually an after effect. Indeed, the attack 17 did not target the feature LIT401 but the feature FIT301.

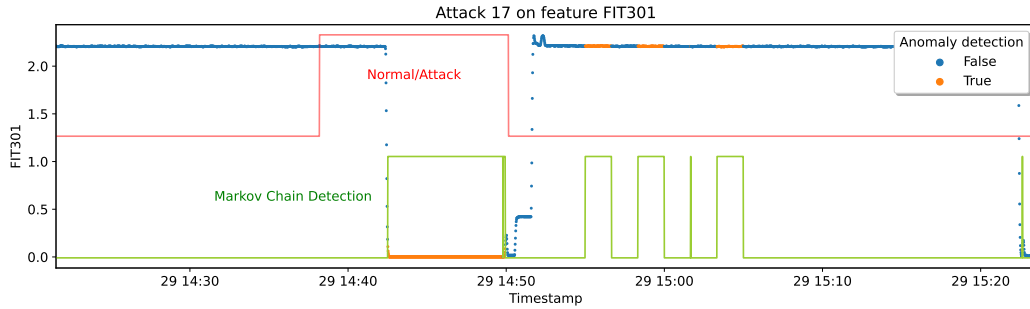


Figure 6.2: Attack 17 detected on feature FIT301

The graph above represents the attack 17 on the feature FIT301. The colors used have the same meaning as on the graph 6.1. An attack targets FIT301, and the anomaly is detected by our Markov Chain from the moment the attack has an effect on the feature. We can see that once the attack is over, other anomalies are picked up by our model because of the after effects.

The LIT401 feature 6.1 suffers from the indirect effects of the attack on the feature

FIT301 as we can see on the Figure 6.2 which creates this gap between the labeling of the dataset and the real anomaly which is in reality a system anomaly. It is important to understand this kind of after effect because it will have a consequent impact in the continuation of our evaluation and especially a negative effect on the accuracy of our models. As we can see, after the attack 17 is finished, we have some False Positives detected by the Markov Chain Model.

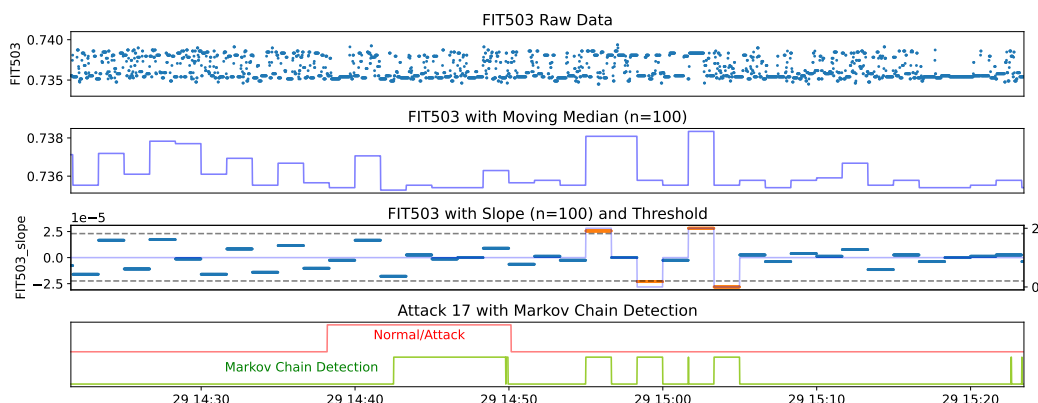


Figure 6.3: Attack 17 detected on feature FIT301

The Figure 6.3 shows us in detail the transformation of the feature FIT503. The first plot shows us the basic FIT503 feature and then in the second plot it undergoes a transformation with the moving median with a window of 100. The third plot shows the transformation of the moving median to the slope transformation as well as the two thresholds in the black dotted line calculated from the slope of the training set on the quantiles 0.1% and 99.9%. The slope is colored in orange when the status on the slope is below or above these quantiles, creating status based on extreme values being out of the ordinary. We can also notice that a blue line represents those statuses that will create states with all the other features having undergone the same transformation.

It is interesting in the fourth plot to see that the status on the feature FIT503 match with the Markov detection model. This is due to the fact that we are in the presence of unknown states directly linked to the status of the feature FIT503. Unfortunately, they are False Positives. This shows the difficulty to detect after effects and to differentiate it from an error related to the sensitivity of the Markov Chain model. The main drawback of Markov Model is its sensibility of the after effects and thus the detection of False Positives due to the creation of states never seen before.

6.2 Accuracy for each Models

For all models we carried out a two-stage evaluation. Firstly by evaluating confusion matrices and then by looking at the metrics for each feature.

The part with the confusion matrices is composed of a first matrix with on its left the "Attack" and "Normal" labels. The "Attack" label represents the data points labelled as an attack in the dataset and the "Normal" label, the data points without attacks.

At the base of the matrix we find "True" and "False" values representing the outcome of our anomaly detection. The "True" label indicates that the data points have been labelled by our model as an anomaly and the opposite for the "False" label. The first two matrices are performed on the same data with the difference that the second matrix is expressed with percentages.

Then the third matrix represents the labelling of our models only with respect to the data labelled as an "Attack" in the dataset.

The forth matrix represents our labelling with respect to the "Normal" data in the dataset.

The second part with the metrics per feature is a concatenation of all confusion matrices of the features used by the model. It also shows interesting metrics about the "Normal" and "Attack" data points.

The data used to perform our confusion matrices only includes the testing set.

6.2.1 Min-Max

In this model, we only use cyclic features as explained in section 5.4 Min-Max model. The non-use of acyclic features is based on the assumption that these features do not have predictable behavior. However, the min-max model is based on the repetition of values, which corresponds perfectly to cyclic variables that go continuously from one value to another. This assumption has proven to be true.

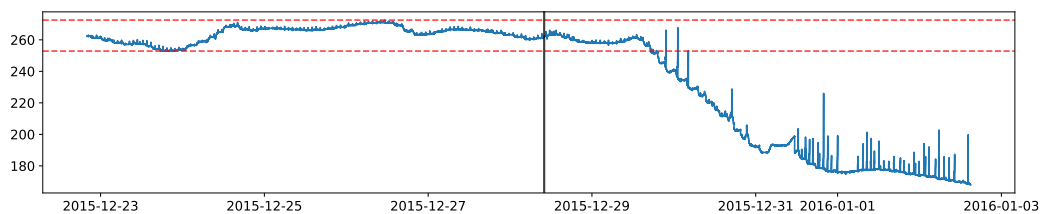


Figure 6.4: Chart showing the minimum and maximum value from the Min-Max model on an acyclic feature (AIT201)

The chart above corresponds to the feature AIT201 from the beginning of the dataset to the end on January 2, 2016, at 14:59:59. The vertical axis corresponds to its value and the horizontal axis represents the timestamp. The separation between the train set and the test set is marked with the vertical black line which is on December 28, 2015, at 10:00:00. The upper horizontal red line shows the maximum value from the train set and the lower one is for the minimum value. These two lines are used as thresholds to determine if a value corresponds to abnormal behavior or not.

As we can see quite blatantly on this particular feature, shortly into the test set, the value of the feature exceeds the minimum value observed in the training set. The vast majority of the test set would therefore be considered as subject to an attack which is false. This is due to the behavior of acyclic features. This behavior is not unique, therefore we can't use acyclic features.

Confusion Matrix

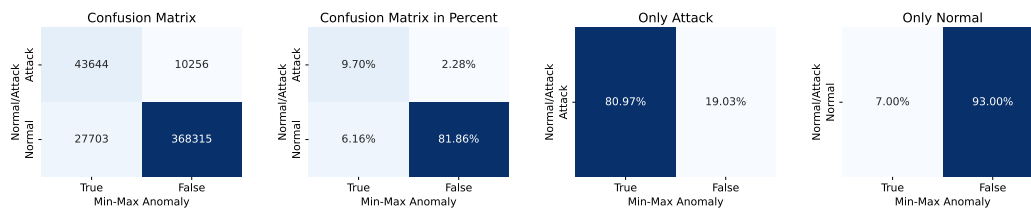


Figure 6.5: Confusion Matrix with the Min-Max model

The first confusion matrix shows us that on the 54621 seconds of attack that the testing set contains, 43644 seconds are correctly detected as an anomaly using the Min-Max model. This detection represents 80.97% of the attacks as we see in the third matrix "Only attack". Unfortunately, as we see in fourth matrix, on "Normal" data there is 7% of the detected anomalies that are actually after effects or inaccuracies from sensors. Indeed, the Min-Max model being the model with the biggest amount of True Positives, there is only a very small chance that it could be wrong by detecting a false anomaly, the cases for which the model is wrong are as already seen in the Figure 6.1 on the feature LIT401.

Metrics per features

Our evaluation of the Min-Max model can be done on each cyclic feature, so we have a better visual on the possible over detected or under detected anomalies for each cyclic feature.

	TP	FP	TN	FN	Count Anomaly	Precision Attack	Recall Attack	F1-Score Attack	Count Normal	Precision Normal	Recall Normal	F1-Score Normal
LIT401 -	35625	20222	375796	18275	55847	0.64	0.66	0.65	394071	0.95	0.95	0.95
LIT301 -	30704	3190	392828	23196	33894	0.91	0.57	0.7	416024	0.94	0.99	0.97
LIT101 -	2610	4347	391671	51290	6957	0.38	0.05	0.09	442961	0.88	0.99	0.93
FIT201 -	2987	43	395975	50913	3030	0.99	0.06	0.1	446888	0.89	1	0.94
DPIT301 -	1634	81	395937	52266	1715	0.95	0.03	0.06	448203	0.88	1	0.94
FIT601 -	0	44	395974	53900	44	0	0	0	449874	0.88	1	0.94
FIT101 -	0	4	396014	53900	4	0	0	0	449914	0.88	1	0.94
FIT301 -	0	3	396015	53900	3	0	0	0	449915	0.88	1	0.94

Figure 6.6: Metrics per feature with the Min-Max model

Figure 6.6 gives us the information for all the cyclic features in the Min-Max model. We have already seen the confusion matrix for all the features combined in Figure 6.5. The difference in Figure 6.6 is that each feature has its own confusion matrix which gives us the TP, FP, TN and FN in seconds linked to a single feature in the detection of anomaly. The column named "Count Anomaly" gives us the number of seconds of anomalies detected by the model. The next three columns give us the precision (2.2), the recall (2.3) and the f1-score (2.4) only for the data labeled as an attack by the iTrust team. The next column gives us the number of seconds that have not been detected as an anomaly by the model. For the last three columns, we find the precision, the recall and the f1-score for the data that have been labeled as normal.

As can be seen in this Figure 6.6, the anomalies detected mainly concern the tanks, namely the LIT401, LIT301 and LIT101 features. Indeed, as can be seen in the pdf of the attacks 4.1, most of the attacks made by the iTrust team attempt to overflow or underflow the tanks (15 of the 36 attacks) through the pumps linked to these tanks. The Min-Max model is very powerful in detecting this cases because if an attack is linked to the overflow or underflow of a tank, it will inevitably be detected by the thresholds set used for this model.

This reasoning explains the number of False Positives in the Figure 6.6 because underflow and overflow create numerous after effects. Having to raise or lower the level of a tank's contents is a time-consuming operation. This situation has already been seen in attack 17 on the LIT401 feature in Figure 6.1. The duration of these aftershocks therefore creates many False Positives and this leads to poorer results

than what is actually happening.

Unfortunately, only the three tanks give a lot of information. Indeed, the FIT feature and the DPIT do not raise a lot of anomalies. We think that this is simply due to the way the attacks are carried out. Indeed, the attacks on overflow and underflow only concern pumps and tanks, either the pump is kept open/closed, or the tank value is set to a value outside the min-max values. The disadvantage of this model is that in case of long attacks that increase/decrease the volume every second, the attacks take time to be detected. On the opposite, a rapid change of volume outside the limits creates an instant anomaly detection.

6.2.2 Variance Zero

It is necessary to know that some features have a variance of zero on the whole dataset, it concerns the features P202, P401, P404, P502, P601 and P603. We are not going to study these features in-depth, they are in fact the backup of pumps that never had to take over from the main pumps to which they are associated. These features are obviously taken into account in the models but having a variance of zero they are useless to the understanding of the model that is why we will not talk about them.

Confusion Matrix

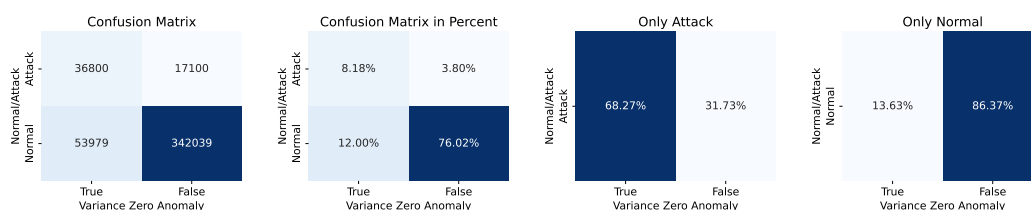


Figure 6.7: Confusion Matrix with the Variance Zero model

The confusion matrix of the variance zero model is bad. Indeed there are too many False Positives compared to the number of True Positives detected. This may indicate a diagnostic error in the discrete features having a variance of zero in the dataset train. The second hypothesis, just as likely, is that the attacks have completely disrupted the system, and several backups would have taken over from the original pumps. As a reminder, as soon as a discrete feature with a variance of zero changes its value in the test, it implies the detection of an anomaly.

Metrics per features

Our evaluation of the variance zero model can be done on each discrete feature with a variance of zero in the training dataset, so we have a better visual on the metrics on the anomaly detection for each discrete feature of variance zero. The next graph has the same specification as described in the Min-Max model 6.2.1.

	TP	FP	TN	FN	Count Anomaly	Precision Attack	Recall Attack	F1-Score Attack	Count Normal	Precision Normal	Recall Normal	F1-Score Normal
P201	739	53732	342286	53161	54471	0.01	0.01	0.01	395447	0.87	0.86	0.86
UV401	33234	128	395890	20666	33362	1	0.62	0.76	416556	0.95	1	0.97
P501	33230	126	395892	20670	33356	1	0.62	0.76	416562	0.95	1	0.97
P402	31973	65	395953	21927	32038	1	0.59	0.74	417880	0.95	1	0.97
P102	3029	96	395922	50871	3125	0.97	0.06	0.11	446793	0.89	1	0.94
P403	0	60	395958	53900	60	0	0	0	449858	0.88	1	0.94
P204	56	0	396018	53844	56	1	0	0	449862	0.88	1	0.94
P206	55	0	396018	53845	55	1	0	0	449863	0.88	1	0.94

Figure 6.8: Metrics per feature table with the Variance Zero model

As we can see on the Figure 6.8, only three features give the majority of the True Positive detection nevertheless, the False Positive rate for each feature is extremely low, which would have made it an excellent anomaly detection model. Unfortunately, we can also see that the feature P201 has a very bad precision on attacks but also on normal behaviors. The hypothesis of an error on the diagnosis of a discrete feature with a variance of zero is very likely for the feature P201. Indeed, as we can see on the Figure 6.8, feature P201 has a total of 54471 seconds of anomaly detected in which only 739 seconds of True Positive. This feature alone made our Variance Zero model unusable.

The case of P201

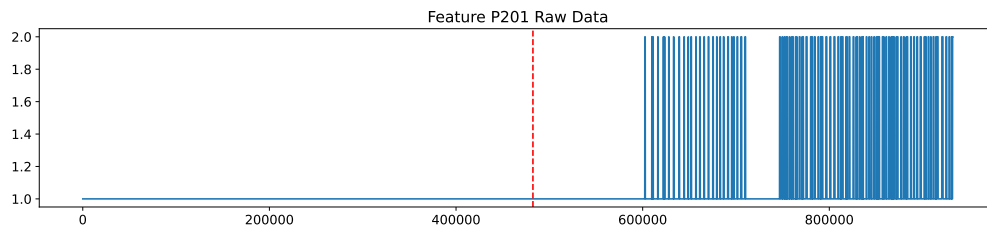


Figure 6.9: Feature P201 Raw Data

The Figure 6.9 shows the feature P201, which has as abscissa the time in seconds during all the dataset and in ordinate the value of the feature P201 varying between

1 and 2. There is a red dotted line marking the limit between the train set where we had a variance of zero for the feature P201 and the test set where the pump P201 starts to activate. Our basic hypothesis is therefore wrong, we thought that the training dataset was supposed to represent all possible values for each feature unfortunately, the P201 feature is not a backup and yet is never activated during the train set, which means that it is mistaken for a backup and gives a lot of False Positives as we have seen in the Variance Zero model metrics table 6.8.

6.2.3 Longest Sequence of Slope Direction (Cyclic LS)

Confusion Matrix

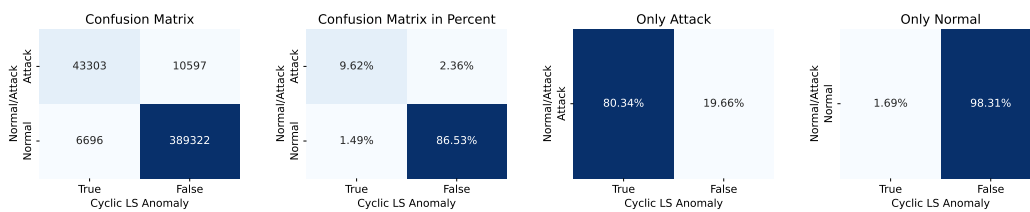


Figure 6.10: Confusion Matrix with the Cyclic LS model

The confusion matrices above present the results of the Cyclic LS model, as we can see, the True Positives are in large numbers. Indeed, the model detects 17 out of 29 attacks (without the failed ones), these 17 attacks detected represents 43303 seconds of the detection of anomaly which represents 80% of the total duration of attacks.

Metrics per features

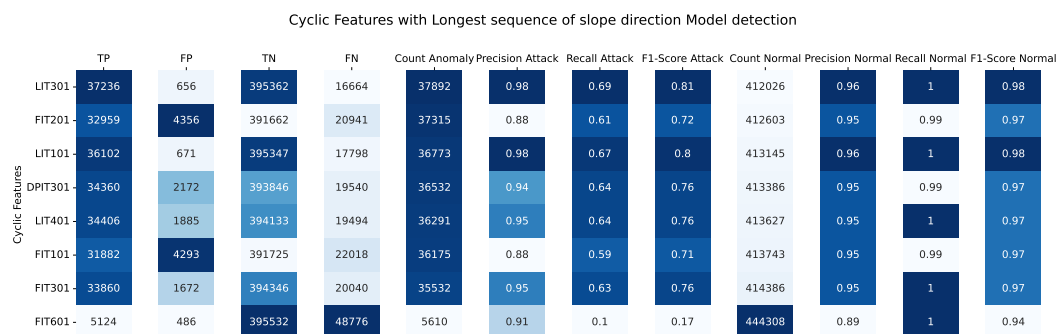


Figure 6.11: Cyclic features in Cyclic LS

As we can see in the Figure 6.12, almost all the cyclic features except the FIT601 feature are involved in the majority of the anomaly detections, which makes this model one of the most reliable. Indeed we have more certainty than for the Min-Max model 6.2.1 which had only two features giving the majority of the detections. The interest of this model lies in its robustness, because the anomalies are detected in the majority on all the cyclic features, except the FIT601 which we will analyze in more detail in the next case : 6.12.

The Cyclic LS model will be used in a general way for any cyclic features. It is therefore a generalized model for other datasets with cyclic features. Nevertheless, there is a disadvantage to the Cyclic LS model with cyclic features, it is the delay of anomaly detection. In fact, the model is based on the time spent for a feature in the same trend, it is therefore easy to understand that we have to wait for the maximum time calculated in the train set for each trend of the feature to be able to detect an anomaly.

The case of the FIT601

The FIT601 feature being the one with the least anomalies detected, it is interesting to study it more closely.

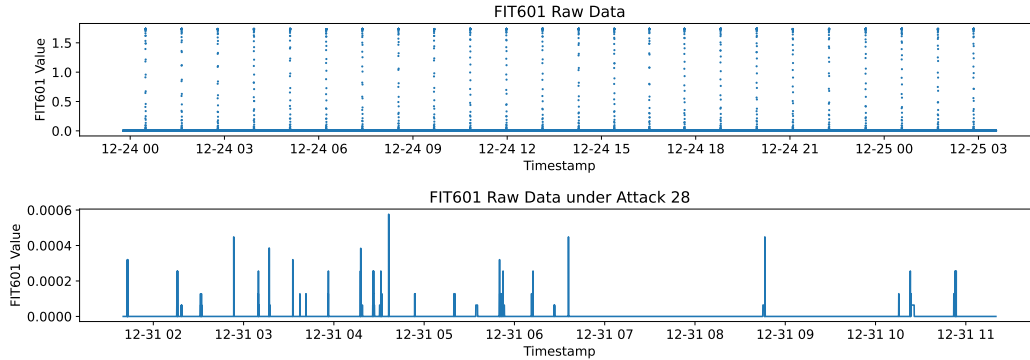


Figure 6.12: FIT601 raw data and raw data under attack 28

The Figure above 6.12 has as first plot the basic feature during a period of one day. The second plot is a zoom on attack 28 which lasts more than ten hours. Both plots have the feature value on the ordinate and the timestamp on the abscissa. We can see on the basis of these graphs that the feature has a very special shape, indeed it has a value close to zero throughout the train set interspersed with spikes every hour. This model based on the ascending, descending and neutral slope has little effect here considering the behavior of this feature FIT601. Only the

neutral slope can have an impact which explains why it has little anomaly detection.

The values on attack 28 do not allow to detect a lot of anomalies either, indeed we notice that the sensor contains constant noise which prevents the good calculation of the neutral slope and therefore the non-detection of the attack. Features with the same unusual characteristics would not be very useful in the Cyclic LS model.

6.2.4 Discrete Longest Sequence (Discrete LS)

Confusion Matrix

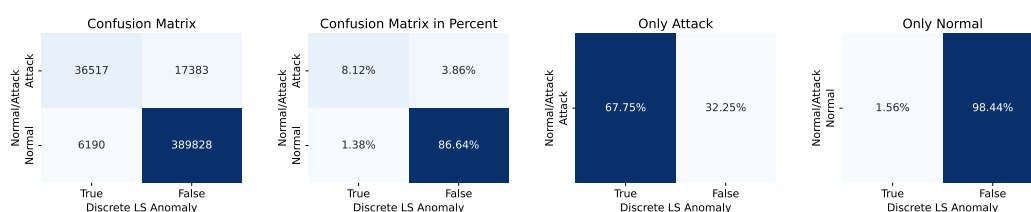


Figure 6.13: Confusion Matrix with the Longest Sequence Discrete model

The Figure 6.13 on the confusion matrices of the Discrete Longest Sequence model shows the assurance that it is a good model. Indeed if we refer to the percentage of False Positive which is 1.38%, and the rate of False Negative which is 3.86% we have an excellent model. These great results may be explained by the characteristics of the discrete features. Indeed, in this type of data, there is no chaos or noise. This implies that we don't have to apply any transformation on this type of feature. Not using transformation leverage the authenticity of the data and avoid to make approximations. To summarize, the purity and reliable data provided by the discrete variables make this model the one who is detecting the less FP and the most TN.

Metrics per features

Discrete Features with Discrete Longest Sequence Model detection

	TP	FP	TN	FN	Count	Anomaly Precision	Attack Recall	F1-Score	Attack Count	Normal Precision	Normal Recall	Normal F1-Score
MV301	35680	3997	392021	18220	39677	0.9	0.66	0.76	410241	0.96	0.99	0.97
P602	35516	3985	392033	18384	39501	0.9	0.66	0.76	410417	0.96	0.99	0.97
MV303	35524	3976	392042	18376	39500	0.9	0.66	0.76	410418	0.96	0.99	0.97
P302	35272	75	395943	18628	35347	1	0.65	0.79	414571	0.96	1	0.98
MV302	35142	54	395964	18758	35196	1	0.65	0.79	414722	0.95	1	0.98
MV201	32959	2183	393835	20941	35142	0.94	0.61	0.74	414776	0.95	0.99	0.97
MV304	35027	65	395953	18873	35092	1	0.65	0.79	414826	0.95	1	0.98
P203	33023	1077	394941	20877	34100	0.97	0.61	0.75	415818	0.95	1	0.97
P101	32944	1059	394959	20956	34003	0.97	0.61	0.75	415915	0.95	1	0.97
MV101	31883	1534	394484	22017	33417	0.95	0.59	0.73	416501	0.95	1	0.97
P205	31339	577	395441	22561	31916	0.98	0.58	0.73	418002	0.95	1	0.97

Figure 6.14: Discrete feature in Discrete LS model detection

The Figure 6.14 shows us that the model is good and that no feature really gives more information than another. Even if the model detects 80% of the total attack time, unfortunately it only detects 8 of the 29 attacks that can be detected (without failed attacks). The next table shows us all the attacks detected by the model.

Attack	11	17	23	25	27	28	30	38
Anomaly Detected	87	415	80	541	1076	34208	63	46
Time Duration	560	716	696	611	1690	34208	1170	280
Delay Detected	441	302	617	71	615	0	1065	155

The table informs for each attack the anomaly detected which corresponds to the number of seconds detected by the Discrete LS model in seconds, as well as for the time duration, which corresponds to the total duration of the attack in seconds. We can see that the attack being detected in its entirety takes a lot of weight in the accuracy of the model. Indeed, attack 28 lasts for 34208 seconds, it represents 94% of the True Positive of the confusion matrix 6.13. A big disadvantage of this model is the attack detection time. As explained in the Cyclic LS model for the cyclic variables 6.2.3, we have to wait until the attack exceeds the maximum allowed time in the same value for a feature before being able to detect an anomaly.

We also notice that the three best features in terms of TP have approximately the same number of FP.

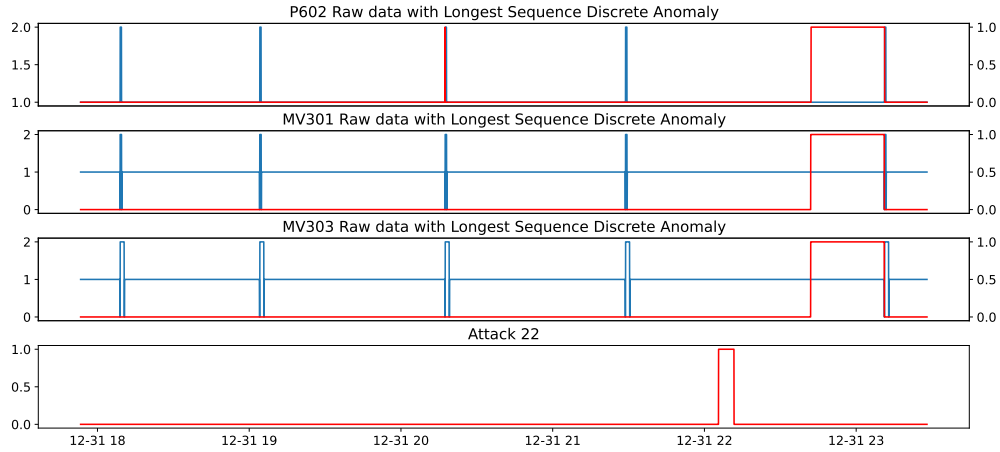


Figure 6.15: Confusion Matrix on the Markov Chain model

The Figure 6.15 above shows the three best features in blue with their value on the ordinate and the anomaly detection in red in binary form. The four plots have a timestamp starting 4 hours before attack 22 that we find in binary form in the last plot, lasting only 8 minutes. Further analysis shows that the FPs for features MV301, MV303 and P602 are at the same place and time shortly after attack 22. We notice that the three features have a cyclic period until attack 22 intervenes, which does not target any of these features. After this attack, the three features miss their cyclic spikes which trigger, as we can see in red, their anomaly detection due to their longest sequence of value. This is an after effect of 2000 seconds which represents a third of the FP of the model. This makes us think that the False Positives of the Discrete LS are possibly only after effects.

6.2.5 Markov Chain

Confusion Matrix

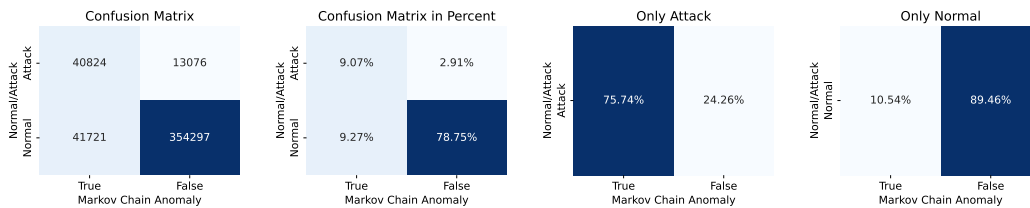


Figure 6.16: Confusion Matrix on the Markov Chain model

As one can imagine, the confusion matrix gives us the proof that we have a lot of False Positives which we knew was going to be one of the major issues and difficulties with this Markov chain model. On the other hand, we also have a lot of True Positives which is excellent because the model alone represents 76% of the detections on all the attacks but there is even better, the attack detection is not for a restricted type of attack. Indeed the Markov Chain model detects 28 of the 29 attacks (without failed attacks), the only one not detected is the attack n°1. This can be explained by the fact that the undetected attack consists of a long attack and that the Markov chain model is based on the slope of cyclic and acyclic features. This makes it a very good detector of fast and abrupt attacks on continuous features.

States counts

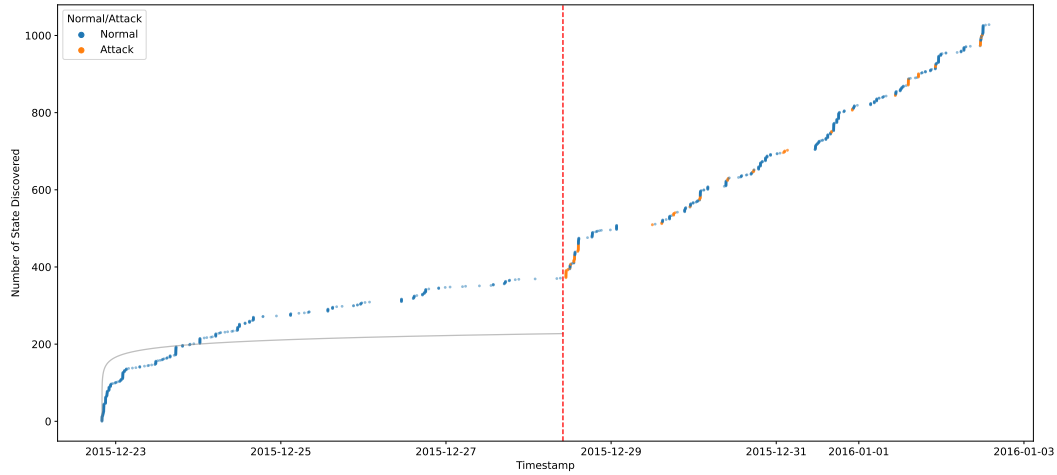


Figure 6.17: Number of states in time

Figure 6.17 gives us an excellent idea of what we could expect in the results of the Markov Chain model. We had already seen the number of states discovered in the train set at Figure 5.13, this graph is its continuation.

The red dotted line shows the limit between train and test set, the states discovered after this line are either True Positives or False Positives. The orange color of the discovered states shows where the states at a time t are labeled as attack and are therefore our True Positives called as "Unknow States" in the section 5.7.2.

The other points in blue are "Unknow States" but as False Positive, they are either due to states not having been discovered during the training phase which may be after effects.

The logarithmic curve ends at the limit between the training and testing set, it

shows us the discovery of extreme states (due to the chaos of the attacks), unseen in the training set 5.7.2.

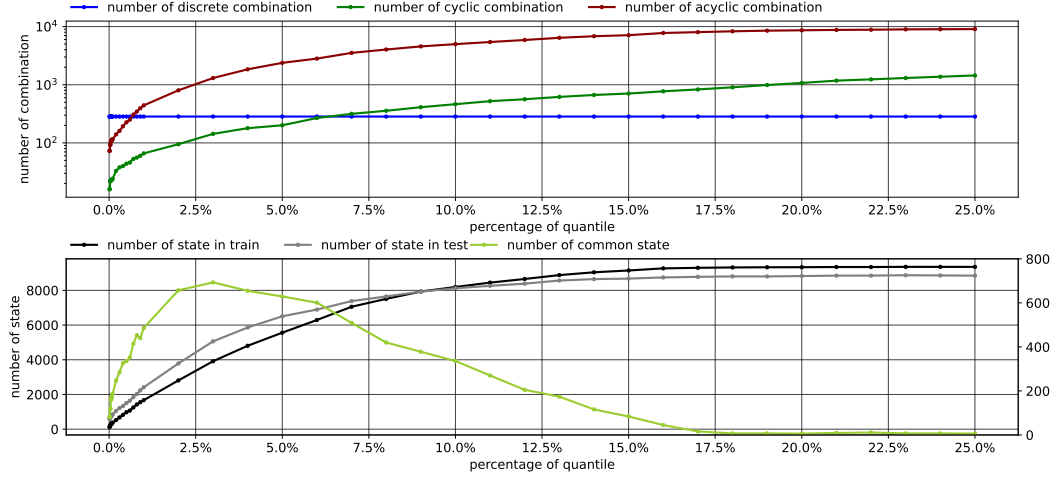


Figure 6.18: Number of states and combination in time

In the Figure 6.18 above we have the same data as in Figure 5.10 about states combinations, with the same specifications serving as a visual reminder. The second plot gives us two additional curves than in Figure 5.10. We can now compare the states in the train and test sets, here the gray curve shows us the states discovered as a function of the quantiles for the states discovered in the test set. The black curve is the same as we had already shown in Figure 5.10 for the states discovered in the train set. Now that we have all the states created in the train and test, it is easy to compare the number of states but also to compare the number of states in common. The value of the light green curve on the second plot shows us the number of states in common between the train and the test on the right ordinate. We can use this curve to confirm our hypothesis 5.7.5 about the coherence between quantile and number of states. Indeed, the smaller the percentage of quantile would have been, the more we would have minimized the chaotic interaction of the cyclic, acyclic and discrete not constant features. The light green curve informs us that indeed the smaller the quantile, the less states there are and the more states there are in common until a certain point.

6.3 Multi-model Evaluation

Multi-model evaluation is an important step in the process of developing and evaluating anomaly detection models, as it allows us to determine which model

or combination of models is most effective in identifying anomalous behavior or patterns in the SWaT dataset. When evaluating multiple models for anomaly detection, it is important to consider the confusion matrix for each model, as it provides valuable information about the model's performance in terms of accuracy, precision, and recall.

Model	TP	FP	TN	FN	Precision	Recall	F1-score
Cyclic LS	9.62	1.49	86.53	2.36	86.61	80.34	83.36
Discrete LS	8.12	1.38	86.64	3.86	85.51	67.75	75.60
Min-Max	9.70	6.16	81.86	2.28	61.17	80.97	69.69
Markov Chain	9.07	9.27	78.75	2.91	49.46	75.74	59.84
Variance Zero	8.18	12.00	76.02	3.80	40.54	68.27	50.87

Table 6.2: Summary of metrics for each model

The table above shows us the summary for each model. These metrics can be used to identify the model who is the most accurate and precise in detecting anomalies and normal behaviors. It allows also to know which one is the best suited for anomaly detection task as well as to understand the strengths and weaknesses of our models.

It is interesting to note that the Cyclic LS is the model with the best precision (2.2) as well as the best f1-score (2.4) which makes it a very good model even though it does not have the best metrics in terms of TP and FP. This means that it is a model that does not take many risks of being wrong in detecting anomalies.

Conversely, we can see that the Discrete LS is the model that predicts the least amount of anomalies being True Positive and yet has the lowest False Positive rate, this is due to the fact that it takes little risk in prediction.

The difference between the two is that the Cyclic LS will yield many True Positives but more False Positives than the Discrete LS model.

It should be noted that the Min-Max is the model with the most anomaly detection being True Positive which makes it a very good candidate in the future multi-model combination.

The Markov Chain model is also a very good candidate because it is the only one to detect many attacks but unfortunately also has a lot of False Positives because it is sensitive to after effects. The Variance Zero model is the worst model in terms of False Positives as we have already seen but it is better in terms of TP than the Discrete LS. The multi-model is going to be a mixture of all the models with their weaknesses and strengths so all the True Positives of each model will be accumulated. The same goes for the False Positives, which means that the multi-model is not going to be very accurate at first sight. But we will see that it

is still possible to create a multimodel by reducing the number of False Negatives.

6.3.1 Confusion Matrix on Multi-Model

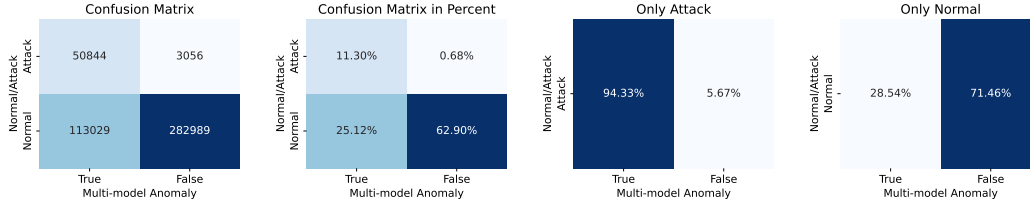


Figure 6.19: Confusion Matrix on the Multi-Model

The combination of all our models gives us a result that is disconcerting to say the least. On the other hand, we detect almost all the attacks in terms of duration (the number of seconds) with 94% of detections but we have a very high number of False Positives. As we have already seen in the confusion matrix of the Zero Variance model, the case of P201 alone is involved in the multi-model, giving 53732 seconds of False Positives. The confusion matrix for the multi-model is not what we expected and for a good reason, we expected a more decent number of False Positives. That being said, we still have to explore the different model combinations that give us the best compromise between True Positives and False Positives.

6.3.2 Combination of models

Not Used	Pr A	Re A	F1 A	Pr N	Re N	F1 N	TP	TN	FP	FN
Variance Zero	42.57	93.63	58.53	98.96	82.81	90.17	11.22	72.89	15.13	0.76
Markov Chain	38.24	91.64	53.96	98.60	79.85	88.24	10.98	70.29	17.73	1.00
Cyclic LS	30.84	92.81	46.29	98.65	71.67	83.02	11.12	63.08	24.94	0.86
Min-Max	33.42	90.35	48.79	98.29	75.50	85.40	10.82	66.45	21.57	1.16
Discrete LS	31.03	92.91	46.52	98.67	71.89	83.18	11.13	63.28	24.74	0.85

Combination	Pr A	Re A	F1 A	Pr N	Re N	F1 N	TP	TN	FP	FN
All models	29.05	94.37	44.43	98.90	68.63	81.03	11.31	60.41	27.61	0.67

Table 6.3: Combination of models using Method "OR"

The table 6.3 above gives us the metrics of 4 possible combinations of 4 models. This allows us to highlight a model that would pull the others down or conversely that pulls the multi-model up. The table has as first column the model which is not used in the combination of four models so it is easier to see which model has the most negative or positive impact. The last line is just a reminder of the metrics of the multi-model with the five models. The notations in the table are similar to the previous ones "Pr" being the acronym of Precision, "Re" stands for Recall, "F1" for F1-score. The letter which follows thoses notations means: "A" for the data concerning only the Attacks, and "N" for the data not comprising an Attack.

We based our evaluation of our multi-model on method "OR" that uses the models by taking all the anomalies detected by each model.

We can easily see that the Variance Zero model is the least essential on the table 6.3 with the method "OR". Indeed, when this one is not used, the multi-model has the best accuracy on Attacks as well as the best percentage in term of False Negative. It makes this combination without Variance Zero a better Multi-model. The Min-Max model has a prominent place in this ranking, without it we would have the worst multi-model combination. The Discrete LS model also has an important place in the multi-model, if we refer to the precision of Attack brought by this model we realize that without it the precision would be the lowest. It is important to note that the multi-model with the 5 models still has the highest True positive rate but unfortunately it also has the highest False positive rate which makes it a worse multi-model than what we could have without the Variance Zero.

Multimodel Analysis on the Attack 25 with an overview on the LIT401

Here is a concrete example of attack 25 using our multi-model to show how the multi-model evaluation works on each of our features to detect anomaly.

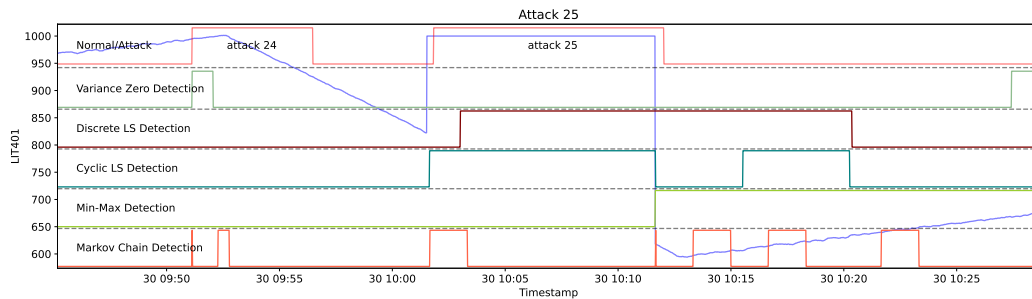


Figure 6.20: Attack 25 with Multi-Model Analysis

The Figure 6.20 represents very accurately the anomaly detection for each of our models on the attack 25 following the 24th one, as we can see on the left y-axis, each model is represented in binary form where 1 is equivalent to an anomaly detected, 0 otherwise. Starting from the bottom to the top we find the Markov Chain model in tomato red, the Min-Max model in yellow green, Cyclic LS in dark green, The Discrete LS value in brown, and finally the Variance Zero model in light green. On the top in red are the real attacks labeled by iTrust being at 1 when there is an attack and 0 if there is no attack. On the y-axis we find the value of the LIT401 feature with its curve in blue crossing the Figure.

Attack 25 consists in setting the value of LIT401 to 1000 and among other things to open the pump P402. We have chosen this demonstration attack to show the efficiency and advantages of each of our models. As we can see at the time of the attack, the Markov Chain and Cyclic LS is triggered because the value of the LIT401 having been set to 1000. This change of value is too abrupt and has thus created a new state in the Markov Chain model and revealed an anomaly. The triggering of the Cyclic LS is due to the fact that the value of the LIT401 has never been more than 5 seconds in a row with the same value.

For the Discrete LS model, the P402 pump was already open before the attack, the hackers left the pump open which triggered an anomaly on the Discrete LS. Unfortunately we can also see that the Min-Max model thinks it has discovered an anomaly when attack 25 ends, which is due to an after effect. Indeed, when the attack ends, the LIT401 continues its race and ends below its minimum equal to 774mm calculated during the training phase. All the plots of attacks can be found in our public repository on Github Bodenghien and Cornette [34].

By using all these models, we improve our detection of the attack. As usual, the Markov Chain Model detects the attack very quickly because it only needs an unusual behaviour to launch an alert. However, its detection does not last long, unlike LS Cyclic which detects the attack quickly and in its entirety. We can also see the problems linked in some cases by models notably with after effects which, once again, are a real problem for detection.

6.3.3 Multi-model Detection

We know the exact time in seconds each attack last and by having run each model on the test dataset, we know the number of seconds that were well predicted by each model. Thus we can give a detection percentage for each model for each attack.

It is important to mention that the Figure 6.21 does not contain the failed attacks mentioned in section 6.1.1 Dataset Test. As a reminder, the failed attacks are attacks 4, 13, 14, 19, 20, 24 and 29. We will see these cases in the next sub-section 6.3.4 Failed Attack.

The Figure 6.21 at the right shows all the physical attacks of the test dataset. For each attack, the percentage of detection by model and the percentage detected by the multi-model. These attacks are in descending order according to the type and duration of the attack. This order is not important but we can see a link because the shorter the attack is the less likely it is to be detected by our models.

The Figure 6.21 allows us to make a first observation, all the attacks are detected. Moreover, almost all of them are detected in their entirety except for attacks 3, 17, 34 and 38 for which our models had a little more difficulty.

Taking into account only successful attacks, 97.8% of them are detected as anomaly by our models. The model with the most detected attacks was the Min-Max model with 84.3% followed by the Cyclic LS model with 83.7% and then the Markov Chain model with 78.6%. The Discrete LS and Variance Zero models both detected only 70% of the total attack time which ranks them last. We can also notice the huge gain in the number of attacks detected using the Markov Chain model, which detects almost all of them.

	Min-Max Detection	Markov Chain Detection	Cyclic LS Detection	Discrete LS Detection	Variance Zero Detection	Multi-Model Detection	Attack Duration	Attack Type	
28	98	98	98	100	93	100	34209	SSSP	-28
41	81	18	56	0	0	81	1675	SSSP	-41
8	100	20	68	0	0	100	964	SSSP	-8
1	100	0	0	0	0	100	940	SSSP	-1
17	3	62	55	58	0	64	717	SSSP	-17
32	94	63	92	0	7	100	601	SSSP	-32
11	10	100	18	16	97	100	561	SSSP	-11
36	88	100	86	0	0	100	540	SSSP	-36
33	0	67	87	0	0	98	444	SSSP	-33
2	98	61	0	0	95	98	443	SSSP	-2
7	100	27	95	0	0	100	429	SSSP	-7
3	31	41	0	0	0	41	383	SSSP	-3
31	100	87	98	0	0	100	367	SSSP	-31
40	0	100	0	0	100	100	297	SSSP	-40
16	83	88	0	0	0	88	276	SSSP	-16
6	0	90	0	0	0	90	196	SSSP	-6
10	100	25	0	0	11	100	160	SSSP	-10
34	0	59	0	0	60	60	101	SSSP	-34
21	0	17	97	0	0	100	721	SSMP	-21
25	4	15	96	88	0	100	612	SSMP	-25
35	29	92	0	0	0	92	481	SSMP	-35
37	0	87	0	0	0	87	469	SSMP	-37
27	32	4	93	64	0	100	1691	MSSP	-27
26	100	44	98	0	98	100	1445	MSSP	-26
39	0	27	0	0	100	100	401	MSSP	-39
38	0	65	10	16	0	74	281	MSSP	-38
30	100	7	97	5	96	100	1171	MSMP	-30
23	100	29	11	11	0	100	697	MSMP	-23
22	0	100	0	0	96	100	463	MSMP	-22

Figure 6.21: Accuracy for each model for each attack

6.3.4 Failed Attack Detection

The Figure 6.22 is the same as the Figure 6.21 except it is about failed attacks. The attacks that failed but were labeled as real attacks are much less detected by our models. It is interesting to mention that only two models trigger a detection, the Markov Chain and the Variance Zero model.

On the attack 24, the Variance Zero model captured the first 60 seconds of the attack thanks to the P204 and P206 backups that woke up as we have seen in the Figure 6.20. As the attack did not have much impact, the backup pumps went back to sleeping mode and the rest of the attack was not detected. We can also notice the iTrust team's commentary 6.1.1 on attack 24 "Not much impact made due to closure of P-101 because tank LIT-101 became full" which explains why it is not detected in its entirety.

The Markov Chain detects four abnormal events. This may be explained by the fact that it captures some new states due to the sensibility of the model. Even if in some case these failed attacks have an impact (which is very low) on the features it may be detected.

	Min-Max Detection	Markov Chain Detection	Cyclic LS Detection	Discrete LS Detection	Variance Zero Detection	Multi-Model Detection	Attack Duration
14	0	0	0	0	0	0	431
20	0	14	0	0	0	14	395
4	0	0	0	0	0	0	390
24	0	10	0	0	17	27	321
19	0	8	0	0	0	8	259
13	0	0	0	0	0	0	233
29	0	34	0	0	0	34	121

Figure 6.22: Accuracy for each model for each failed attack

6.3.5 Anomaly Detection in Type of attack

Out of the 29 attacks (without failed attacks), here is the distribution of the number of attacks according to the type of attacks described in the section 6.1.2:

- 18 attacks of type SSSP
- 4 attacks of type SSMP
- 4 attacks of type MSSP
- 3 attacks of type MSMP

In terms of total attacks in seconds, our models detect 100% of attack of type MSMP, 95% of SSMP, 93.5% of type MSSP and finally our model detects 90% of SSSP. These percentages are explained by the multiple aspects of the attacks. Indeed, the more there will be entry points in the attack and the more there will be the use of multiple steps or stages in the SWaT, the easier it will be for our models to detect these attacks. Indeed, the more modifications there are in the features, the more chaotic the system becomes and the more anomalies are created.

6.3.6 Multi-model Delay

Now that we have the metrics concerning the percentage of detection for each attack, we can also calculate the delay between the first second of an attack having been labeled by the iTrust team and the first second of detection by each of our models.

The Figure on the right gives the detection delay for each model. On the abscissa every model is referenced and each attack is shown on the ordinate sorted by attack duration. As for the Figure 6.23 we have removed the failed attacks. The boxes without any numbers are attacks that have not been detected for a specific model.

We can see that there are many attacks with a delay of 0 seconds, in reality the Markov Chain model and the Cyclic LS have a delay expressed differently compared to what is shown in the Figure 6.23 that has no latency in its metrics. Indeed we detect anomalies from the first second of the attack but with a latency of 100 seconds. This latency is due to the moving median, as both models use this transformation, we have to wait until the window has finished to pass on the last 100 seconds (it corresponds to the size of the window being equal to 100) to be able to say if there was effectively an anomaly or not.

Moreover, it may be interesting to put the detection time in perspective with the total duration of an attack. For example, if we compare attack 28 which last several hours and attack 41 making half an hour, the detection times are similar but they do not have the same power. Indeed, a shorter attack will be more difficult to detect especially if the attack is of a long type (for example, increasing a tank by 1mm per second), which the models have more difficulty in detecting quickly.

	Min-Max Delay	Markov Chain Delay	Cyclic LS Delay	Discrete LS Delay	Variance Zero Delay	Multi-Model Delay	Attack Duration
28.0	692.0	0.0	681.0	0.0	2237.0	0.0	34209
27.0	0.0	0.0	7.0	615.0	-	0.0	1691
41.0	325.0	299.0	699.0	-	-	299.0	1675
26.0	0.0	303.0	0.0	-	0.0	0.0	1445
30.0	0.0	0.0	0.0	1065.0	0.0	0.0	1171
8.0	0.0	0.0	290.0	-	-	0.0	964
1.0	0.0	919.0	-	-	-	0.0	940
21.0	-	0.0	6.0	-	-	0.0	721
17.0	697.0	259.0	309.0	302.0	-	259.0	717
23.0	0.0	0.0	582.0	617.0	-	0.0	697
25.0	589.0	0.0	0.0	71.0	-	0.0	612
32.0	39.0	21.0	46.0	-	0.0	0.0	601
11.0	0.0	0.0	360.0	441.0	0.0	0.0	561
36.0	66.0	0.0	73.0	-	-	0.0	540
35.0	341.0	38.0	-	-	-	38.0	481
37.0	-	59.0	-	-	-	59.0	469
22.0	-	0.0	-	-	0.0	0.0	463
33.0	-	9.0	59.0	-	-	9.0	444
2.0	0.0	62.0	-	-	0.0	0.0	443
7.0	0.0	0.0	0.0	-	-	0.0	429
39.0	-	5.0	-	-	0.0	0.0	401
3.0	243.0	225.0	-	-	-	225.0	383
31.0	0.0	47.0	7.0	-	-	0.0	367
40.0	-	0.0	-	-	0.0	0.0	297
38.0	-	0.0	153.0	155.0	-	0.0	281
16.0	32.0	34.0	-	-	-	32.0	276
6.0	-	0.0	-	-	-	0.0	196
10.0	0.0	120.0	-	-	142.0	0.0	160
34.0	-	41.0	-	-	40.0	40.0	101

Figure 6.23: Delay for each model for each attack

Unfortunately, many attacks have a duration of less than 10 minutes.

The Cyclic LS model is particular in its detection speed. It will take more time to detect ascending and descending slopes than neutral ones. Indeed, if the aim of an attack is to fix a value on a tank and therefore to create a neutral slope attack, the detection time will be fast because neutral slopes are generally short for cyclic features. Conversely, ascending and descending slopes are long by nature and are therefore less quickly detected because it is necessary to wait until they exceed the maximum length observed in the train set. This explains the heterogeneity of the detection times for this model.

The Discrete LS and Cyclic LS models have a slow detection because they have to wait for their methodology, which does not make them good models for blitz attacks. The Markov chain model, on the other hand, has the ability to detect these fast attacks making it an excellent model for rapid detection as does the Min-Max model which detects rapidly changing attacks outside of the system boundaries.

The main thing to remember from this analysis is that some models capture attacks but are too slow to be effective in an IDS, like the Discrete LS.

As we have just seen, each model has a different detection speed, a better way to see this is a boxplot of each model according to the delay of each attack detected by the model.

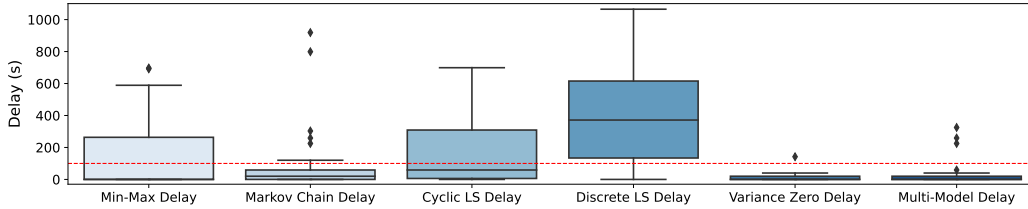


Figure 6.24: Boxplot of delay for each model

On the Figure 6.24 we can see the boxplots with on the ordinate the delay of detection for each attack detected by a model and on the x-axis our models names plus our multi-model (containing all of them).

We can see on this graph that the Discrete LS model has a much higher detection median than the other models, the most probable hypothesis is that as it is a model that must wait for the feature to exceed a certain time before being detected as an anomaly, the detection times are intrinsically linked to the limit set by the maximum values calculated in the training dataset.

The red line represents the latency of the Markov Chain and Cyclic LS Detection equal to 100 seconds also equivalent to the window size on the moving median used by both models.

The Variance Zero has an incredibly fast detection, as the model is based on waking up backups, when a hacker damages a pump, the backup wakes up creating an instant anomaly.

Overall the multi-model detects 16 attacks instantly using the Min-Max (detects 11 out of the 16), Discrete LS (detects 1 out of the 16) and Variance Zero model (detects 8 out of the 16). Counting the latency of the Markov chain model and the Cyclic LS, 6 attacks are detected in less than 100 seconds, 4 attacks are detected between 100 and 200 seconds and finally, 3 attacks with more than 200 seconds. These attacks with long detections are the soft attacks where the hacker penetrated the system gently and slowly modified the sensors which is harder for our models to detect quickly.

Attack 17, seen earlier, was designed to leave a pump open indefinitely, the Discrete LS was able to detect this but only when the maximum length was exceeded which no model could detect instantly. Similarly, attack 3 aimed to artificially increase the LIT101 tank by 1mm every second, which was detected by the Min-Max and the Markov chain model but with more than 200 seconds delay.

In the end, what counts is the impact of the attack, a fast and abrupt attack will be detected instantly. Conversely, a slow attack aimed at infiltrating the system to break it will be slower to detect but will have less impact on the system. It is therefore less serious to detect slow attacks with a delay than not to detect fast attacks instantly.

6.4 Limitations and discussion

Our implementation has some limitations but also positive points as we based the approach of our IDS on the ideal properties introduced by Porras and Valdes [9]. The goal of this section is to discuss the drawbacks and benefits from our IDS based on those properties.

6.4.1 Accuracy

The accuracy depends on the models themselves but also on the combination of the models. As we have seen in the table 6.3, some models create many False Positives and degrade performance. It should also be borne in mind that failed attacks, which are not detected by our IDS, are included in the confusions matrix presented in the evaluations, which degrades our results.

However, it is evident that we detect many False Positives especially compared to methods using specifications as mentioned in chapter 3 Related Work.

Among our five models, they all turned out to have strong and weak points. Some are good at detecting certain attacks more than others but one model proved to be particularly ineffective with our dataset: the Variance Zero model. Although it was expected to give excellent results, it has quickly highlighted having a high number of FP and FN but also a very low TP and TN rate. All this is due to a feature that makes it our least efficient model and our results are much better when we do not use it in our multi-model.

Some combinations of models give better results than others. Indeed, there is a profusion of combinations, it is possible to do "AND" and "OR" operations between models and to privilege certain models. We could for example have taken the anomalies of the model with the following formula:

$$(\text{Min-Max} \wedge \text{Markov Chain}) \vee (\text{Cyclic LS} \wedge \text{Discrete LS}) \quad (6.1)$$

However, care must be taken with the selection in order to improve the results. This specific combination 6.1 would have given us an accuracy of 90 % with a recall of 68 % and a detection of 17 attacks out of the 29 attacks.

As it has been said many times, our accuracy is limited by the fact that we cannot detect after effects. In order to greatly improve these metrics, we would need a detection system to know when an attack is over. This detection should be able to tell when there are only after effects left from a finished attack. Also, it has to be able to detect if the distortion of a feature is an indirect effect of an attack on another feature.

6.4.2 Performance

A very important point in our IDS is the speed of calculation of our models. In fact, as our IDS is a multi-model based on mathematical aspects, it is fast to detect anomalies in real-time. Mathematically our models require only small operations to be performed in order to detect anomalies. The computational power and memory required by our models are derisory to detect all attacks. This shows us that it is not necessarily necessary to have complex and time and memory-consuming models. Our models could therefore be used on machines with limited computing power.

Indeed, without counting the pre-processing, our models mostly only compare the few values they have in memory learned in the train set and the real-time data. There is obviously also a need for computing power for the transformations on the data but these are not heavy operations.

More specifically, the Min-Max model detects most of the anomalies instantly because it is a simple condition to satisfy on each feature to detect anomalies. The same situation happens with the Discrete LS model on discrete features as well as the model Variance Zero. The temporal complexity of the Cyclic LS and the Markov chain model are slightly higher than the other models. Indeed, these two models require a transformation with the moving median 100 as well as the calculation of the slope and the creation of states and status. This makes them slower to predict but it is still very low as we only have 100 seconds of data to process.

The Min-Max, Variance Zero, Discrete LS, and Cyclic LS models only require a few variables to be calculated in pre-processing compared to the Markov Chain model. The Markov Chain model has a transition matrix, a stationary matrix and a list of states, so it takes a bit longer to compute. Furthermore, the moving median, slope and status transformations must also be calculated over the entire train phase. Checking whether a state exists or not takes only a short time as it is simple to access a states list.

6.4.3 Completeness

Our IDS detects all the attacks as we saw in the section 6.3.3, although it also detects some failed attacks as shown in the section 6.3.4 where only the Markov chain detected false attacks because some had a slight impact. This is the case of attack 24, detected by the Markov model.

The multi-model detected 97.8% (94.37% with attacks failed) of the total time of successful attacks but with an accuracy of 29% on attacks using all of our models.

Our models, unfortunately, failed to detect 1111 seconds of attacks which represents 2.2% of the 51735 seconds of successful attacks. The percentage of detection is very high. There are however many false positives.

6.4.4 Fault Tolerance

This section goes beyond the basic scope of our thesis, especially as we have not taken into consideration the network part of SWaT. However, it is necessary for the good functioning of an IDS. Based on the assumptions made in the threat model 5.1, we found only one limitation concerning the fault tolerance part of our models. In the case where there is a crash of the system and we have to restart it, we won't be able to detect real attacks during the process of initialisation since we have cut this part of the process from the dataset as explained in 4.2.5. During this part everything will be detected as an anomaly since this behavior is unknown, the Markov Chain model will trigger anomalies event every second until the stabilization of the system.

Since we haven't focus on work on the network part, we can make even more assumptions as what have been stated in the Threat model : other measures such as the use of encrypted connections, firewalls, and managing access with a least-privileged method can limit the risk of attack.

6.4.5 Timeliness

The aim of our IDS is to be able to detect anomalies in real-time. As we saw in the delay section, some attacks cannot be detected in real-time. Nevertheless, our solution allows us to detect attacks within a reasonable time. Overall the multi-model detects 16 attacks instantly as well as 6 attacks detected in less than 100 seconds. And finally, it detects 7 attacks with a delay of more than 100. The 100 second latency for the Markov Chain and Cyclic LS models is due to the constraint of the median moving time required for their operation. This trade off between speed and quality of detection is very important in our implementation. We have favored the quality and reliability of anomaly detection by increasing the detection time. This choice makes our IDS more reliable by creating less false alarms and more true anomaly detection. We insist that only these two models are impacted by this latency, which does not concern all attack detections.

6.5 Future work

Our work could be improved on several points. The detection time of the Markov chain model and the Cyclic LS could be faster, indeed the features containing noise require more time before we can have a tangible information to analyze.

This time constraint could be abolished with a scaling solution. The use of 100 Markov Chain models and 100 Cyclic LS models, each with a one second lag, all calculating a moving median with a size of 100 calculated at each second, could give better results. The time complexity would then be higher, making it slower for real-time anomaly detection but the detection time would probably be reduced.

We believe that it would be possible to improve the cyclic and acyclic feature detection system. The autocorrelation technique that we used has proved its worth, but we think that in the future, it would be better to use a more complex technique. The autocorrelation only gives a visual result which we had to analyze by creating a threshold to separate the cyclic and acyclic features. It would be interesting to test a solution with a signal processing technique using the fast Fourier transform method. This technique would analyze the sum of the frequencies of each feature in more depth and would possibly be more robust in determining the cyclicity of a feature but it would also make it more complex to interpret.

The choice of quantiles on the continuous features when choosing the status for the Markov Chain model could have been done less visually and more mathematically. We believe it would be possible to create a simple method for choosing quantiles. The principle would be to test all the quantiles and return only the one most similar to a logarithmic curve and therefore the most efficient in terms of state creation.

In spite of these possible improvements, we think we have done the trick of simple but robust models taking into account a Markov Chain. It would undoubtedly be more interesting to start on a new method of detection with for example.

Chapter 7

Conclusion

This thesis proposes an implementation of an IDS designed for anomaly detection with a multi-model that has been successfully developed to detect potential attacks on the system, as predicted in our original hypotheses 5.1. We used the SWaT dataset of the iTrust team to train and test our model, which showed a good performance in terms of complexity/benefit.

In particular, we used a multi-stage modeling approach that consisted of data pre-processing, followed by training of five anomaly detection models. We also studied the impact of different pre-processing and parametrization selection techniques on the performance of our models, and determined that some approaches are more effective than others in detecting the specific attacks envisioned in our study.

It should be noted that our model is not foolproof and there may be false alarms or undetected anomalies. However, overall, we have demonstrated that our model is a simple and effective tool in detecting anomalies in industrial control systems and can be used as a first line of defense against potential attacks. Furthermore, our models have been designed to be general and easily adaptable to any ICS. However, it is necessary to configure some parameters for the creation of status as well as for the transformations to be done on the data in order to use our approach in an optimal way.

Furthermore, our detection models are all transparent, so it is possible to see how our models make decisions and how they work. This makes them easier to use than an opaque model with which it is more difficult to understand how the model makes its decision.

Finally, we discussed future directions for research in this area, including ex-

ploring new pre-processing approaches to further improve the performance of our models. Also improving the accuracy of anomaly detection through after-effect detection. In summary, our multi-model anomaly detection could have had better accuracy but highlights good features such as its simplicity, ease of understanding the results as well as its performance.

Chapter 8

Appendix

No	Name	Type	Description
1	FIT-101	Sensor	Flow meter; Measures inflow into raw water tank.
2	LIT-101	Sensor	Level Transmitter; Raw water tank level.
3	MV-101	Actuator	Motorized valve; Controls water flow to the raw water tank.
4	P-101	Actuator	Pump; Pumps water from raw water tank to second stage.
5	P-102 (backup)	Actuator	Pump; Pumps water from raw water tank to second stage.
6	AIT-201	Sensor	Conductivity analyser; Measures NaCl level.
7	AIT-202	Sensor	pH analyser; Measures HCl level.
8	AIT-203	Sensor	ORP analyser; Measures NaOCl level.
9	FIT-201	Sensor	Flow Transmitter; Control dosing pumps.
10	MV-201	Actuator	Motorized valve; Controls water flow to the UF feed water tank.
11	P-201	Actuator	Dosing pump; NaCl dosing pump.
12	P-202 (backup)	Actuator	Dosing pump; NaCl dosing pump.
13	P-203	Actuator	Dosing pump; HCl dosing pump.
14	P-204 (backup)	Actuator	Dosing pump; HCl dosing pump.
15	P-205	Actuator	Dosing pump; NaOCl dosing pump.
16	P-206 (backup)	Actuator	Dosing pump; NaOCl dosing pump.
17	DPIT-301	Sensor	Differential pressure indicating transmitter; Controls the backwash process.
18	FIT-301	Sensor	Flow meter; Measures the flow of water in the UF stage.
19	LIT-301	Sensor	Level Transmitter; UF feed water tank level.
20	MV-301	Actuator	Motorized Valve; Controls UF-Backwash process.
21	MV-302	Actuator	Motorized Valve; Controls water from UF process to De-Chlorination unit.
22	MV-303	Actuator	Motorized Valve; Controls UF-Backwash drain.
23	MV-304	Actuator	Motorized Valve; Controls UF drain.
24	P-301 (backup)	Actuator	UF feed Pump; Pumps water from UF feed water tank to RO feed water tank via UF filtration.
25	P-302	Actuator	UF feed Pump; Pumps water from UF feed water tank to RO feed water tank via UF filtration.

Figure 8.1: Description of all actuators and sensors

26	AIT-401	Sensor	RO hardness meter of water.
27	AIT-402	Sensor	ORP meter; Controls the NaHSO ₃ dosing(P203), NaOCl dosing (P205).
28	FIT-401	Sensor	Flow Transmitter ; Controls the UV dechlorinator.
29	LIT-401	Actuator	Level Transmitter; RO feed water tank level.
30	P-401 (backup)	Actuator	Pump; Pumps water from RO feed tank to UV dechlorinator.
31	P-402	Actuator	Pump; Pumps water from RO feed tank to UV dechlorinator.
32	P-403	Actuator	Sodium bi-sulphate pump.
33	P-404 (backup)	Actuator	Sodium bi-sulphate pump.
34	UV-401	Actuator	Dechlorinator; Removes chlorine from water.
35	AIT-501	Sensor	RO pH analyser; Measures HCl level.
36	AIT-502	Sensor	RO feed ORP analyser; Measures NaOCl level.
37	AIT-503	Sensor	RO feed conductivity analyser; Measures NaCl level.
38	AIT-504	Sensor	RO permeate conductivity analyser; Measures NaCl level.
39	FIT-501	Sensor	Flow meter; RO membrane inlet flow meter.
40	FIT-502	Sensor	Flow meter; RO Permeate flow meter.
41	FIT-503	Sensor	Flow meter; RO Reject flow meter.
42	FIT-504	Sensor	Flow meter; RO re-circulation flow meter.
43	P-501	Actuator	Pump; Pumps dechlorinated water to RO.
44	P-502 (backup)	Actuator	Pump; Pumps dechlorinated water to RO.
45	PIT-501	Sensor	Pressure meter; RO feed pressure.
46	PIT-502	Sensor	Pressure meter; RO permeate pressure.
47	PIT-503	Sensor	Pressure meter;RO reject pressure.
48	FIT-601	Sensor	Flow meter; UF Backwash flow meter.
49	P-601	Actuator	Pump; Pumps water from RO permeate tank to raw water tank (not used for data collection).
50	P-602	Actuator	Pump; Pumps water from UF back wash tank to UF filter to clean the membrane.
51	P-603	Actuator	Not implemented in SWaT yet.

Figure 8.2: Description of all actuators and sensors

Bibliography

- [1] Mordor Intelligence. Industrial control systems market - growth, trends, covid-19 impact, and forecasts (2021 - 2026), 2021. <https://www.mordorintelligence.com/industry-reports/industrial-control-systems-market-industry> Accessed: 23-12-2022.
- [2] Kevin R Skinner, Nils Ole Tippenhauer, Stephen Checkoway, Konrad Rieck, Dennis Friess, Yan Shoshitaishvili, Christopher Kruegel, and Giovanni Vigna. Cyber attacks on industrial control systems: Past, present and future. *Journal of Cyber Security Technology*, 2(3):155–169, 2018. <https://journals.sagepub.com/doi/full/10.1177/1550147718794615> Accessed: 20-12-2022.
- [3] Reciprocity. Cloud security vs. traditional security, 2021. <https://reciprocity.com/blog/cloud-security-vs-traditional-security/> Accessed: 23-12-2022.
- [4] Josh Fruhlinger. Stuxnet explained: The first known cyberweapon, Aug 2022. <https://www.csoononline.com/article/3218104/stuxnet-explained-the-first-known-cyberweapon.html> Accessed: 22-12-2022.
- [5] Itrust datasets, Dec 2015. <https://itrust.sutd.edu.sg/itrust-labs-datasets/> Accessed: 31-07-2022.
- [6] ITrust. Itrust labs home, 2021. https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/ Accessed: 07-11-2022.
- [7] Cyber physical systems security. <https://www.dhs.gov/science-and-technology/cpssec> Accessed: 20-12-2022.
- [8] Jingyi Wang, Jun Sun, Yifan Jia, Shengchao Qin, and Zhiwu Xu. Towards ‘verifying’ a water treatment system. In *Formal Methods*, pages 73–92. Springer International Publishing, 2018. doi: 10.1007/978-3-319-95582-7_5. https://doi.org/10.1007/978-3-319-95582-7_5 Accessed: 27-12-2022.

- [9] Phillip A Porras and Alfonso Valdes. Live traffic analysis of tcp/ip gateways. In *Proceedings of the 1998 ISOC Symposium on Network and Distributed System Security (NDSS'98)*. San Diego, CA, 1998.
- [10] A survey of intrusion detection on industrial control systems. <https://journals.sagepub.com/doi/full/10.1177/1550147718794615> Accessed: 21-12-2022.
- [11] David Allison, Paul Smith, Kieran McLaughlin, Fan Zhang, Jamie Coble, and Rodney Busquim. Plc-based cyber-attack detection: A last line of defence. *iaea*, 2019.
- [12] Adithyavegi. Time series in machine learning, Nov 2020. <https://medium.com/analytics-vidhya/time-series-in-machine-learning-c3299742b2e1>, Accessed: 04/01/23.
- [13] Jim Frost. Interquartile range (iqr): How to find and use it, Mar 2022. <https://statisticsbyjim.com/basics/interquartile-range/> Accessed: 22-12-2022.
- [14] Pralhad Teggi. 5 steps to get an understanding on correlation, auto-correlation and partial auto-correlation, Jan 2020. <https://medium.com/analytics-vidhya/5-steps-to-get-an-understanding-on-correlation-auto-correlation-and-partial-auto-correlation-e71f6b4bba81>, Accessed: 03/01/22.
- [15] Joos Korstanje. The f1 score, Aug 2021. <https://towardsdatascience.com/the-f1-score-bec2bbc38aa6>, Accessed: 04/01/23.
- [16] Ivana Burgetová, Petr Matoušek, and Ondřej Ryšavý. Anomaly detection of ics communication using statistical models. In *2021 17th International Conference on Network and Service Management (CNSM)*, pages 166–172, 2021. doi: 10.23919/CNSM52442.2021.9615510.
- [17] Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, 9:120043–120065, 2021. doi: 10.1109/ACCESS.2021.3107975.
- [18] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018. <https://openreview.net/forum?id=BJJLHbb0-> Accessed: 21-12-2022.

- [19] Yeji Choi, Hyunki Lim, Heeseung Choi, and Ig-Jae Kim. Gan-based anomaly detection and localization of multivariate time series data for power plant. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 71–74, 2020. doi: 10.1109/BigComp48618.2020.00-97.
- [20] Tailai Wen and Roy Keyes. Time series anomaly detection using convolutional neural networks and transfer learning, 2019. <https://arxiv.org/abs/1905.13628> Accessed: 21-12-2022.
- [21] Bin Zhou, Shenghua Liu, Bryan Hooi, Xueqi Cheng, and Jing Ye. Beatgan: Anomalous rhythm detection using adversarially generated time series. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4433–4439. International Joint Conferences on Artificial Intelligence Organization, 7 2019. doi: 10.24963/ijcai.2019/616. <https://doi.org/10.24963/ijcai.2019/616> Accessed: 21-12-2022.
- [22] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018. <https://arxiv.org/abs/1803.01271> Accessed: 21-12-2022.
- [23] Laze Gjorgiev and Sonja Gievska. Time series anomaly detection with variational autoencoder using mahalanobis distance. In Vesna Dimitrova and Ivica Dimitrovski, editors, *ICT Innovations 2020. Machine Learning and Applications*, pages 42–55, Cham, 2020. Springer International Publishing. ISBN 978-3-030-62098-1.
- [24] Chao Wang, Bailing Wang, Hongri Liu, and Haikuo Qu. Anomaly detection for industrial control system based on autoencoder neural network. *Wireless Communications and Mobile Computing*, 2020:8897926, Aug 2020. ISSN 1530-8669. doi: 10.1155/2020/8897926. <https://doi.org/10.1155/2020/8897926> Accessed: 22-12-2022.
- [25] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15 (9):1779–1797, jul 2022. ISSN 2150-8097. doi: 10.14778/3538598.3538602. <https://doi.org/10.14778/3538598.3538602> Accessed: 22-12-2022.
- [26] Cheng Feng, Venkata Palleti, Aditya Mathur, and Deeph Chana. A systematic framework to generate invariants for anomaly detection in industrial control systems. 01 2019. doi: 10.14722/ndss.2019.23265.
- [27] Igor Nai Fovino, Alessio Coletta, Andrea Carcano, and Marcelo Masera. Critical state-based filtering system for securing scada network protocols.

- IEEE Transactions on Industrial Electronics*, 59(10):3943–3950, 2012. doi: 10.1109/TIE.2011.2181132.
- [28] David J. Hand. Classifier Technology and the Illusion of Progress. *Statistical Science*, 21(1):1 – 14, 2006. doi: 10.1214/088342306000000060. <https://doi.org/10.1214/088342306000000060> Accessed: 27-12-2022.
 - [29] Moving median is robust to anomalies, Dec 2018. <https://anomaly.io/moving-median-robust-anomaly/index.html> Accessed: 28-12-2022.
 - [30] False data injection definition, examples, prevention, Nov 2022. <https://crashtest-security.com/false-data-injection-attack/> Accessed: 30-12-2022.
 - [31] Sourabh Mehta. 5 real-world use cases of the markov chains, May 2022. <https://analyticsindiamag.com/5-real-world-use-cases-of-the-markov-chains/>, Accessed: 30-12-2022.
 - [32] Chaîne de markov, Oct 2022. https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markov, Accessed: 07/01/23.
 - [33] Stationary distributions of markov chains. <https://brilliant.org/wiki/stationary-distributions/>, Accessed:04/01/23.
 - [34] David Bodenghien and Guillaume Cornette. Swat analysis for this master thesis, 2022. https://github.com/dbodenghien/SWaT_Analysis_TFE.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl