Data Distribution-Based Change Detection Methods in SWaT

Máté Hekfusz, Vrushali Mahajan, Nada Al-Amodi, and Fatima Assendal

Faculty of Informatics, Eötvös Loránd University Budapest, Pázmány Péter stny. 1/C., 1117

1 Introduction

In this project, we have built a complete stream processing pipeline and analyzed data distribution-based change detection methods on the Secure Water Treatment (SWaT) dataset. Within the complicated field of water treatment systems, quick and accurate identification of changes, anomalies, and evolving patterns is important for operational security and resilience. Our work aspires to contribute insights into the efficacy of change detection algorithms, specifically on the dynamic and challenging water treatment data that SWaT provides.

Data distribution-based drift detection consists of using a a distance function to quantify dissimilarity between the distribution of historical data and new data. If the difference is statistically significant enough, drift is detected and the learning model is updated. [4] While they are less commonly used than error rate-based change detection (which includes classic methods like DDM and ADWIN), data distribution-based methods can operate in an unsupervised or semi-supervised manner on unlabelled data, making them well-suited for our data.

The linchpin of our project is Docker, a containerization platform which provides a harmonized environment for our array of open-source technologies, one that anyone can replicate by running a single compose file. Within Docker, we have setup and configured diverse services such as Apache Kafka, Apache Spark, Telegraf, InfluxDB, Grafana, and Chronograf to work in tandem, each tool fitting into our stream processing pipeline.

InfluxDB, a time-series database, is another particularly critical part of our framework. The SWaT stream is persisted in InfluxDB, making it available for comprehensive exploration and insightful visualization that single-pass systems cannot give us. Beyond persisting the data stream, InfluxDB also provides a structured foundation for batch training.

This report details our system architecture, the technologies we have used, and the algorithms we have implemented. We also show the results of our experiments with several different change detection methods on the SWaT dataset. We aim to highlight the necessity of stream processing frameworks for today's complex, data-heavy systems and improve understanding of data distribution-based change detection on the specific field of water treatment infrastructure.

2 Dataset

For this project we use the A4 & A5 versions of the Secure Water Treatment (SWaT) dataset provided by iTrust [1]. The SWaT testbed can be viewed as a smaller version of a real industrial water treatment plant in Singapore. It is a cyber-physical system (CPS), where each part of the system is controlled by a PLC. The sensors on each machine can change how the system operates by telling the computer to take certain actions. [3]
Our specific version of the dataset, A4 & A5, captures the operation of the water treatment plant testbed during a single run on July 20, 2019. The testbed was run for about 4 hours, recording sensor data at a frequency of one observation per second. In total, the dataset comprises 15,996 records, each containing the values and sates of dozens of sensors at every part of the water treatment processs. The temporal structure of the dataset unfolds as follows: a predominant period of normal operation spans approximately three hours, setting the baseline for routine system behavior. Subsequently, the dynamics shift in the last hour, during which the system experiences a series of six attacks. These attacks are strategically arranged to target various parts of the water treatment system, injecting anomalies and disturbances into the otherwise stable operational environment.

The deliberate introduction of attacks in the final hour serves to challenge the robustness and resilience of the water treatment processes, simulating real-world scenarios where unexpected events or malicious activities may compromise the integrity of the system. This intentional variation in operational conditions provides a unique and valuable testing ground for the development and evaluation of anomaly detection and change detection algorithms. Researchers and data analysts can leverage this SWaT dataset to explore, analyze, and develop methodologies that enhance the security and reliability of water treatment systems. [2]

3 System architecture

Our project's system architecture consists of a complete streaming pipeline within a Dockerized environment, creating a unified framework for thorough data processing and analysis.

3.1 Phase 1: Data Ingestion and Preprocessing

Our project commences with Kafka, a platform for data ingestion. Here, we convert our dataset into a streaming format and the relevant columns (the ones that are attacked at some point of the run, plus the timestamp) are ingested into Kafka, leveraging its resilience and scalability. To monitor the flow of our data, we deploy CMAK, a Kafka cluster manager. The dataset, loaded from a sheet, undergoes preprocessing in Spark Streaming. Each record, uniquely distinguished by its (original, 2019) timestamp, is formatted and labelled according

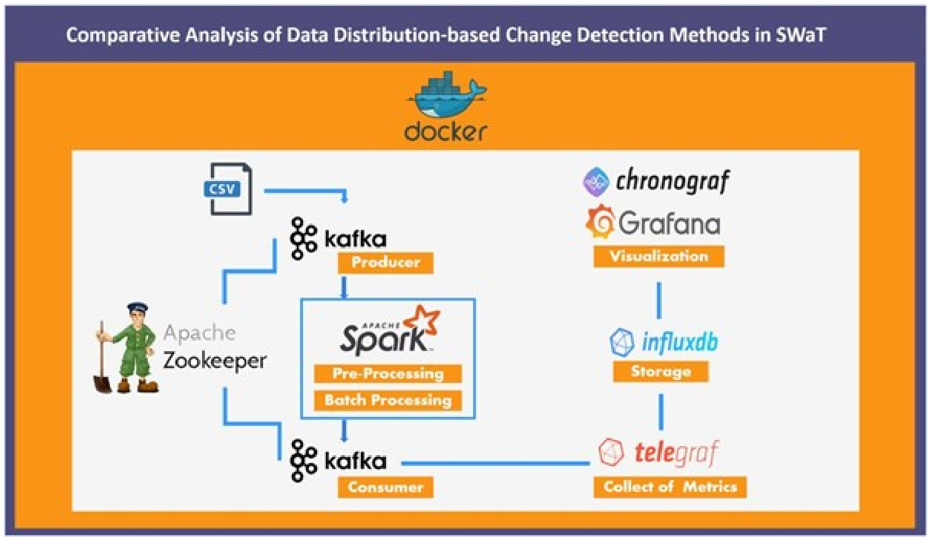

Fig. 1. System architecture

to whether it belongs to normal operation or an attack. Aggregate statistics (mean, standard deviation, min/max, etc) are calculated over data received over 5 and 10 minute windows. This enriched data then flows back into Kafka under a different topic, setting the stage for subsequent analysis.

3.2 Phase 2: Change detection

We employ data distribution-based streaming algorithms to analyze the preprocessed data and detect changes and drifts. We implement seven of these algorithms: kdq-tree, PCA-CD, 1-class-SVM, CUSUM, Windowed Kolmogorov–Smirnov test, Cramér–von Mises test, and RDR. We run each of them on the entirety of the SWaT data stream, recording true and false detections, as well as detection delays for every algorithm. We then plot graphs to visualize the performance of each algorithm, which we will show in our Results section.

3.3 Phase 3: Data storage and batch training

Telegraf takes the preprocessed Kafka records and delivers them to InfluxDB, where time-series data is persisted. Each record's timestamp in InfluxDB aligns with its original timestamp. In batch training, we implement a Decision Tree model taken from the popular open-source scikit-learn package. The model is trained on the initial three hours of normal data and the first three attacks, then tested on the remainder of the data with the last three attacks.

3.4 Phase 4: Visualization

As mentioned before, we implement the TIG stack: picking up Kafka records with Telegraf and storing them in InfluxDB. The last part of the stack is Grafana, which we use to visualize the data now persisted in InfluxDB. Grafana is connected to InfluxDB through an auth token generated by the latter. We also integrate Chronograf into this part of the pipeline, giving us a greater ability to meaningfully visualize our data. Using queries written in Flux and InfluxQL, we craft insightful dashboards to show the change of our sensor values over time, highlight correlations between them, and take a visual look at the attacks themselves.

4 Experiments

4.1 Stream processing

Sliding window aggregation

We implemented sliding window aggregation on streaming data for real-time analysis in Spark. The key steps involve defining a window configuration with a size of 10 minutes and a sliding interval of 5 minutes. The streaming DataFrame is then grouped by both the specified time window and the attack label, allowing the computation of various aggregations, including count, mean, minimum, maximum, standard deviation, rate, and 10-minute average for each numerical column. The use of a watermark on the timestamp ensures that late data is appropriately handled. The results of these aggregations provide a continuous and evolving summary of key statistical measures within sliding time windows. This approach enables the monitoring of dynamic trends and patterns in the water treatment process over time.

Change detection algorithms Data distribution-based change detection methods aim to identify shifts or alterations in the statistical properties of the data distribution. In the context of the SWaT (Secure Water Treatment) dataset, which represents the operational data of a water treatment facility, detecting changes in the data distribution is crucial for identifying abnormal behavior, potential attacks, or shifts in the underlying system.
The below algorithms were implemented using the open-source libraries Menelaus[1] and River[2].

1. **KDQ-tree:** The KDQ-tree [9] algorithm is a robust statistical method designed for detecting changes or drifts in high-dimensional streaming data. Unlike traditional methods, KDQ-tree utilizes a combination of bootstrapping techniques and statistical tests to continuously monitor the evolving

[1] Available at https://github.com/mitre/menelaus
[2] Available at https://github.com/online-ml/river

distribution of the data. In context of the Secure Water Treatment dataset, the algorithm dynamically adapts to variations in the operational characteristics of the water treatment system. As streaming data is consumed from Kafka, the KDQ-tree algorithm analyzes the features, and when a significant change or drift is identified, it triggers a detection event. This adaptive approach allows the algorithm to differentiate between normal operational fluctuations and anomalous events, contributing to the system's resilience in identifying potential attacks or deviations from expected behavior. The KDQ-tree's ability to operate on streaming data in real-time makes it particularly well-suited for applications where prompt detection of changes is crucial for maintaining the security and integrity of the underlying system.

2. **CUSUM:** The Cumulative Sum (CUSUM) algorithm [6] [10] is a statistical technique employed for identifying shifts or alterations in the mean of a chronologically ordered set of observations. [5] CUSUM involves the calculation of a running sum representing the cumulative deviations of individual data points from an anticipated mean. When this cumulative sum surpasses a predefined threshold, it signals the presence of a change point, indicative of a significant shift in the underlying process. The algorithm initiates by computing the mean of the initial observations, serving as the baseline or reference value. With each new observation in the time series, the algorithm computes the deviation of the observed value from the expected mean. The cumulative sum is continually updated based on these calculated deviations. The cumulative sum is then compared against a predetermined threshold. A surpassing of this threshold indicates a substantial departure from the expected mean. Once the cumulative sum exceeds the threshold, the algorithm identifies a change point. Threshold is a parameter which dictates the significance level for considering a change point. When a change point is detected, the algorithm outputs pertinent details such as timestamp, value, and the nature of the change (true positive or false positive based on the attack label). The flexibility to adjust parameters, including the threshold, caters to the specific sensitivity and specificity requirements of the monitoring system without compromising on originality and integrity in reporting.

3. **Cramer-von Mises (CvM) Test:** This algorithm uses the Cramer-von Mises (CvM) test and magnitude-based criteria. The algorithm monitors a time-ordered sequence of observations, specifically for water treatment processes, and aims to detect deviations or anomalies. The algorithm used is a change point detection algorithm based on two main components: the Cramer-von Mises (CvM) test [7] and a magnitude-based criteria. The CvM test is a statistical test used to assess the goodness of fit of a dataset to a theoretical distribution. The algorithm dynamically evaluates the goodness of fit of the observed data to a normal distribution using the CvM test. It then combines this statistical assessment with a magnitude-based criterion to identify significant changes in the values of sensors in water treatment process. The incorporation of attack labels ensures the accuracy of the de-

tection system by distinguishing between genuine attacks and other process variations. The performance metrics such as detection delay and false alarms provide a quantitative evaluation of the system's effectiveness in detecting relevant events.

4. **PCA-CD:** PCA-CD (Principal Component Analysis Change Detection) algorithm [11] is used for real-time detection of changes or anomalies in a streaming dataset. PCA-CD leverages the principles of principal component analysis, a dimensionality reduction technique, to identify significant variations in the data. In this implementation, the algorithm is initialized with parameters such as the window size, divergence metric (utilizing Kullback-Leibler divergence), and a delta value. The PCA-CD algorithm is continually updated with incoming records. This method evaluates whether the detected change corresponds to a normal operation or an attack, updating detection-related metrics such as actual detections, detected attacks, detection delays, and false alarms. This implementation offers a real-time and dynamic approach to monitoring the Secure Water Treatment dataset, providing insights into the effectiveness of the PCA-CD algorithm for change detection in a streaming environment.

5. **KS-WIN:** KSWIN (Kolmogorov-Smirnov Windowed) algorithm [8] is used for change detection in a streaming dataset. KSWIN is a drift detection method designed to identify shifts or anomalies in data distribution over time. The KSWIN algorithm functions by maintaining a sliding window of data observations during an initial training phase. In this phase, the algorithm builds a reference window to establish the statistical characteristics of the normal data distribution. Subsequently, as new data points arrive, KSWIN continuously updates its internal statistics and compares them to the reference window. When a significant deviation is detected, signaling a potential change or drift in the data distribution, the algorithm triggers a warning or drift state. KSWIN is a univariate detection method; in our case, it monitors the 'LIT 301' feature which we found to give the best results for detecting attacks on the entire dataset. During the initial records, the algorithm learns the baseline behavior. After the training, it dynamically assesses incoming data points and raises warnings or identifies drifts when the observed values deviate significantly from the established normal distribution.

6. **One-class-SVM:** The One-Class SVM (Support Vector Machine) [8] [12] is a machine learning algorithm used for anomaly detection, particularly in scenarios where the majority of the data belongs to one class, and anomalies are the exceptions. The algorithm learns the characteristics of the normal class during the training phase, constructing a boundary that encapsulates normal instances. During the testing or detection phase, data points falling outside this boundary are considered anomalies. The key parameter 'nu' controls the proportion of training errors and serves as an upper bound on the

fraction of margin errors and a lower bound of the fraction of support vectors. One-Class SVM is enhanced with a Quantile Filter, introducing a threshold for anomaly detection. As streaming data is consumed from the Kafka topic, the One-Class SVM continually updates its model, dynamically adapting to changes in the data distribution. This method evaluates the accuracy of these detections by comparing them with the known attack labels.

7. **RDR:** The algorithm Relative Density Ratio (RDR) is implemented for drift detection in streaming data. For each incoming record, the algorithm calculates the current distribution of features and employs Kernel Density Estimation (KDE) to model the probability density functions. The RDR is then computed by comparing the log-densities of the current and reference distributions. The algorithm maintains a history of RDR values, allowing it to track changes in data distribution over time. Drift is detected when the maximum RDR in the history surpasses a predefined threshold. This mechanism provides a robust means of identifying shifts in the statistical properties of the streaming data, enabling timely detection of potential anomalies or changes in the underlying data dynamics.This method continuously analyzes the streaming data and dynamically adapts to evolving patterns, offering a valuable tool for real-time monitoring and detection in dynamic datasets.

4.2 Batch processing and training

We implemented batch training for a machine learning classifier using historical data stored in InfluxDB. In batch training, the machine learning model learns from a fixed dataset, allowing for an efficient and thorough exploration of historical patterns. The data is retrieved from InfluxDB using a time-range query, focusing on relevant features and the target variable (attack or normal record). This historical dataset is then divided into two sets: a training set used for model training and a testing set used for model evaluation. The Decision Tree classifier, chosen for its simplicity and interpretability, is initialized, trained on the training set, and subsequently evaluated on the testing set. The accuracy score is computed, providing a quantitative measure of the model's performance in correctly classifying instances as normal or indicative of an attack. This approach reflects the batch learning paradigm, where the model is trained offline on a static dataset, making it well-suited for scenarios where real-time updating is not a requirement. The use of InfluxDB as the data source underscores its role in facilitating efficient historical data retrieval for machine learning tasks. In our case, the training set was the initial three hours of normal data and the first three attacks, while the test set was the remainder of the data which included the last three attacks. The result is an accuracy of approximately 71%. While this is better than the RF and SVM models we also tested, it also shows the difficulty of detecting the various-length (some only lasting a few minutes) attacks which all target different parts of the system.

5 Results

5.1 Comparative analysis of change detection algorithms

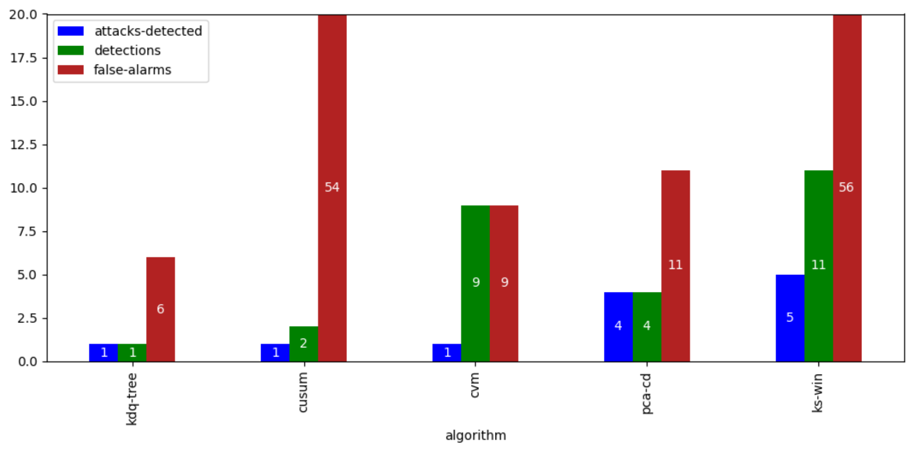

Fig. 2. Attack detections and false alarms

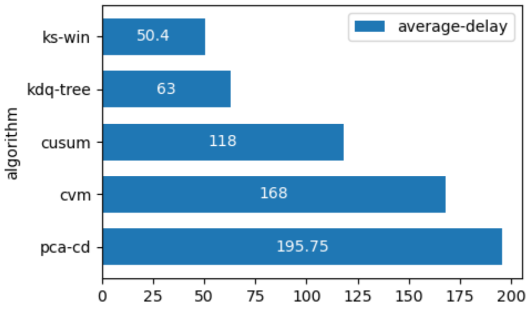

Fig. 3. Average detection delays

As mentioned above, we ran all seven of our change detection algorithms over the entire SWaT data stream and recorded their performance. From the seven, RDR and one-class-SVM performed poorly, likely because of a bad fit to the dataset (which we were unable to compensate for with parameter tuning) and thus are excluded from the graphs. The detection performance of the remaining five is show in Fig. 2. KDQ-Tree detected a single change event while generating six false alarms. PCA-CD was more effective and detected four attacks but reported eleven false alarms. Surprisingly, the univariate KS-WIN performed the best, detecting five of the six attacks at some point, though it did also produce 56 false alarms. CUSUM detected two change events but reported 54 false alarms. Lastly, the CvM algorithm detected nine change events with nine false alarms. Notably, the algorithms exhibited varying performance in detecting specific attacks, with detection delays ranging from 19.0 to 585.0 time units, with their averages (total delays divided by number of attacks detected) shown in Fig. 3.

5.2 Dashboards

The Chronograf dashboard shown in Fig. 4. shows the visualization of the LIT 301 and MV 501 columns of the SWaT dataset. The plotted Timed Moving Average of the MV 501 provides a dynamic representation of the average value, giving more weight to recent data points. This emphasizes the trends and changes in MV 501 over time, offering a responsive insight into the column's evolving patterns. Mean of both column values are plotted using the gauge tool as shown on the dashboard. Sampling periods of MV 501 column is also shown.

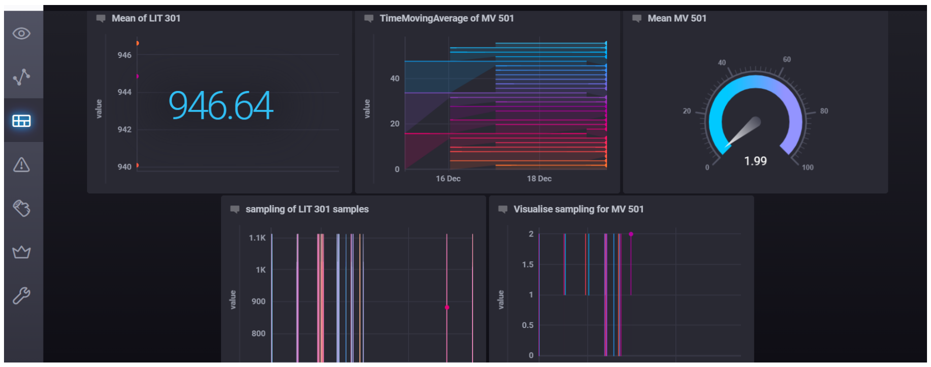
Mean of LIT 301
946.64
TimeMovingAverage of MV 501
16 Dec
18 Dec
Mean MV 501
1.99
sampling of LIT 301 samples
Visualise sampling for MV 501
value

Fig. 4. Chronograf dashboard

The next dashboard, created in Grafana, is shown in Fig. 5. It shows several graphs including a pie chart which shows the ratio of normal and attacked records in the data. The 10 seconds aggregate mean of LIT 301 is plotted showing a contextual anomaly (in other words, a change that is classified as such by its surrounding context, as the values themselves aren't extreme) starting at 7:30. This dashboard also shows when exactly each attack occurs. The line plot of the MV 501 and FIT 401 shows that the latter changes even when the former is attacked, highlighting the correlation between them. Finally, the heatmap at the bottom of teh dashboard shows the columns MV 201 and P101. They were attacked at the same time, and the heatmap shows why: at every point in time, their values are the same, so an efficient attack had to target both, or else it would be easily detected.

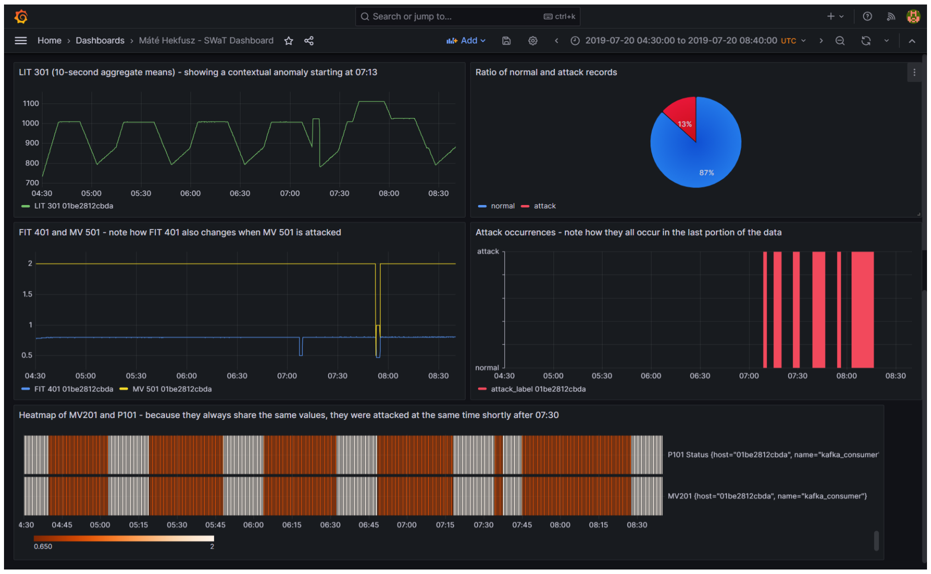
Search or jump to...
Home › Dashboards › Máté Hekfusz - SWaT Dashboard
Add
2019-07-20 04:30:00 to 2019-07-20 08:40:00 UTC
LIT 301 (10-second aggregate means) - showing a contextual anomaly starting at 07:13
LIT 301 01be2812cbda
Ratio of normal and attack records
13%
87%
normal
attack
FIT 401 and MV 501 - note how FIT 401 also changes when MV 501 is attacked
FIT 401 01be2812cbda
MV 501 01be2812cbda
Attack occurrences - note how they all occur in the last portion of the data
attack
normal
attack_label 01be2812cbda
Heatmap of MV201 and P101 - because they always share the same values, they were attacked at the same time shortly after 07:30
P101 Status (host="01be2812cbda", name="kafka_consumer"
MV201 (host="01be2812cbda", name="kafka_consumer")
0.850
2

Fig. 5. Grafana Dashboard

6 Future work

Potential future directions for our work involve expanding the repertoire of change detection algorithms, incorporating both traditional and machine learning-based approaches for a more comprehensive evaluation. Enhancing the system's adaptability to evolving attack scenarios and continuous monitoring of algorithm performance are critical considerations for future iterations. Moreover, exploring and integrating other open-source technologies, such as Apache Flink or Apache Beam, could provide alternative avenues for stream processing and change detection. Additionally, advancements in cloud-native solutions and edge computing present opportunities to optimize scalability and resource efficiency in the architecture. These future endeavors aim to fortify the project's technological landscape and ensure its relevance in the evolving domain of secure water treatment systems.

7 Conclusion

In conclusion, our project successfully establishes a robust and scalable architecture for real-time analysis and change detection in the Secure Water Treatment

(SWaT) dataset. Leveraging Docker, Kafka, Spark, and the TIG stack, we have crafted an integrated environment that seamlessly processes streaming data, implements multiple change detection algorithms, and offers insightful visualizations. The comparative analysis of algorithms sheds light on their performance across various criteria, providing valuable insights into their strengths and weaknesses. As we move forward, the project lays the groundwork for further research and advancements in the domain of secure water treatment, emphasizing the importance of continuous monitoring and adaptive analysis in safeguarding critical infrastructure.

References

1. Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo and Aditya Mathur : A Dataset to Support Research in the Design of Secure Water Treatment Systems. In: Critical Information Infrastructures Security (Lecture Notes in Computer Science), Grigore Havarneanu, Roberto Setola, Hypatia Nassopoulos, and Stephen Wolthusen (Eds.), Springer International Publishing, Cham, pp. 88–99.
2. Kevin Lamshöft, Christian Krätzer, Jana Dittmann, Tom Neubert and Claus Vielhauer : Information Hiding in Cyber Physical Systems: Challenges for Embedding, Retrieval and Detection using Sensor Data of the SWAT Dataset. In: Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security, 2021.
3. Cheah Huei Yoong and Jonathan Heng : Framework for Continuous System Security Protection in SWaT. In: Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control, vol. 31, pp. 1–6, 2020.
4. Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama and Guangquan Zhang : Learning under Concept Drift: A Review. In: IEEE Transactions on Knowledge and Data Engineering, pp. 2346-2363, 2019.
5. WAVE-CUSUM: Improving CUSUM performance in network anomaly detection by means of wavelet analysis. Computers and Security Vol 31, Issue 5, July 2012, pp. 727-735.
6. The modified CUSUM algorithm for slow and drastic change detection in general HMMs with unknown change parameters. In: Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.
7. Comparative analysis of nonparametric change-point detectors commonly used in hydrology.Hydrological Sciences Journal Volume 64, 2019 - Issue 14. pp.1690-1710
8. No Free Lunch Theorem for concept drift detection in streaming data classification: A review. In: WIREs Data Mining and Knowledge Discovery. Volume 10, Issue2.
9. An Information-Theoretic Approach to Detecting Changes in MultiDimensional Data Streams. In:Institute for Operations Research and the Management Sciences, 2006.
10. Change Detection with the Kernel Cumulative Sum Algorithm. In: 58th IEEE Conference on Decision and Control Control Conference 2019.
11. A PCA-Based Change Detection Framework for Multidimensional Data Streams: Change Detection in Multidimensional Data Streams. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2015, pp. 935–944.
12. One-class classifiers with incremental learning and forgetting for data streams with concept drift. In: Soft Computing, Volume 19, pp.3387–3400