CS6748 Final Report, FORT6
Unsupervised Anomaly Detection Approaches to Secure Critical Infrastructure

Michael Gonzalez, mgonzalez94, GtID: 903543370

Sean Lee, slee3413, GtID: 903648883

Vince Villegas, vvillegas6, GtID: 903469249

INTRODUCTION

In the current day and age, technology is increasing at an exponential rate. We have been able to create a globalized computer network called the internet, develop self-driving cars, and create hand-held computers as phones. We can even update and integrate our current technology into older infrastructures. As we continue to develop our technology, the limits of our creations seem boundless. Unfortunately, with every advancement, there will always be individuals that plan to undermine and take advantage of these technologies. Nowadays, cyberattacks have been an ever-increasing threat. They not only include attacks on our digital capital but can also affect our physical infrastructure. There have been cyberattacks that take down power grids, shut down fuel pipelines, attacks on energy plants, and even attacks on our water treatment systems in efforts to poison the drinking water. This OMSA Practicum project is a collaboration with Fortiphy Logic, in an effort to detect cyberattacks on SWaT™ and WADI™ water treatment systems.

PROBLEM STATEMENT / OBJECTIVE

Simulated cyberattacks were executed on small-scale water distribution test beds producing the following data sets: SWaT™ 2015, SWaT™ 2019 and WADI™ 2017. The team evaluated anomaly detection techniques on each data set with the objective of identifying the simulated attacks with high accuracy and specificity. Anomaly detection was approached on a system wide basis and per variable (i.e. sensors and actuators).

DATASET OVERVIEW

Secure Water Treatment (SWaT)™ and Water Distribution (WADI™) testbeds provide datasets simulating water treatment and distribution plants. SWaT™ datasets from 2015 and 2019, and one WADI™ dataset from 2017 were used to develop and assess anomaly detection techniques.

SWaT™ 2015

The SWaT 2015 data was delivered as 3 separate .xlsx files consisting of variables representing the SWaT test composed of six main processes. The three datasets are labeled as SWaT_Dataset_Attack_v0, SWaT_Dataset_Normal_v0, SWaT_Dataset_Normal_v1. The processes control the physical components and behavior of a water treatment facility composed of a six-stage filtration system.

The data encompasses an approximately 11 day period ranging from 12/22/2015 to 1/02/2016 with 53 variables consisting of both sensor and actuator data. The dataset contains a “Normal/Attack” variable representing when a system was under attack as denoted by the categorical “Attack” value. Oppositely, the categorical variable “Normal” signifies a system that was not under attack.

Given that the SWaT_Dataset_Attack_v0 file contained attacks while the SWaT_Dataset_Normal_v1 contained non-attack data points, the files were combined to create a complete dataset with both normal and attack datapoints present. The SWaT_Dataset_Normal_v0 and SWaT_Dataset_Normal_v1 were redundant, therefore only one was included in the final combined dataset.

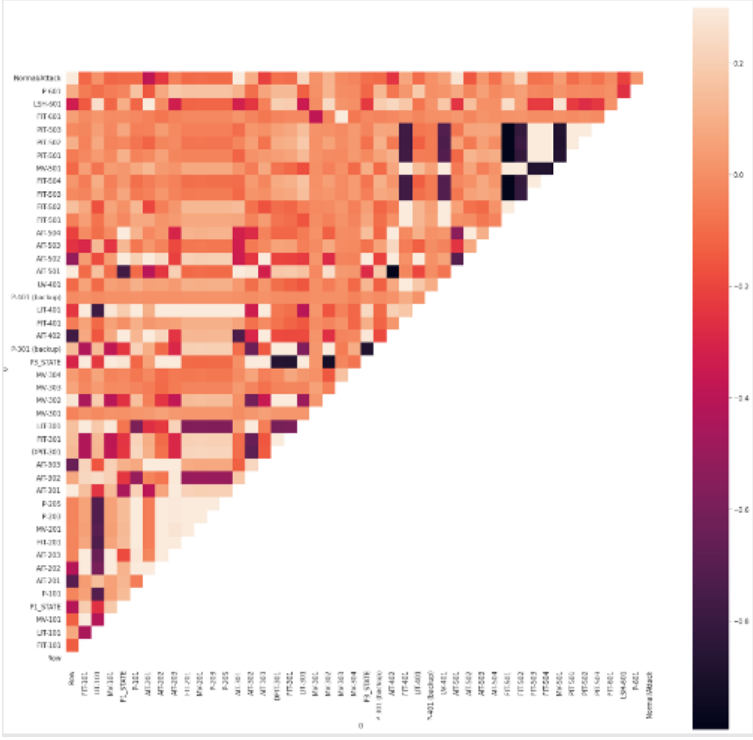
Finally, the combined dataset contains 946,720 rows of data 41 separate attack periods. The attack periods consisted of Single Stage Single Pont, Single Stage Multi-Point, and Multi-Stage Single Point, and Multi-State Multi-Point Attacks.

SWaT™ 2019

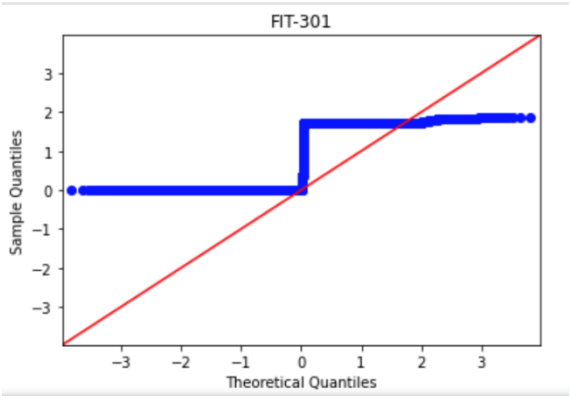
SWaT 2019 v2 is an .xls file that was converted to .csv for formatting and readability purposes. It consisted of 78 columns and 14,999 rows. There are 27 Sensors(variable) and 50 Actuators (categorical) variables. Compared to the SWaT 2015 dataset, it is much shorter in the number of rows but consists of 27 additional columns. The time frame for which this dataset is recorded is from July 20, 2019, 12:30 PM to July 20, 2019 4:30:55 PM. The dataset was recorded at GMT +0, but the "SWaT data collection_20-07-2019 v2" PDF noted that the times of the attacks were in GMT +8, with plant start time from 12:35 PM to 4:35 PM. This meant that we had to convert the timing of GMT +0 to GMT +8 by adding 8 hours, consider ramp up periods from 12:30 PM to 12:35 PM, and ramp down period from 4:35 PM to 4:39:55 PM. There were 6 attacks, with the variables and times as followed:

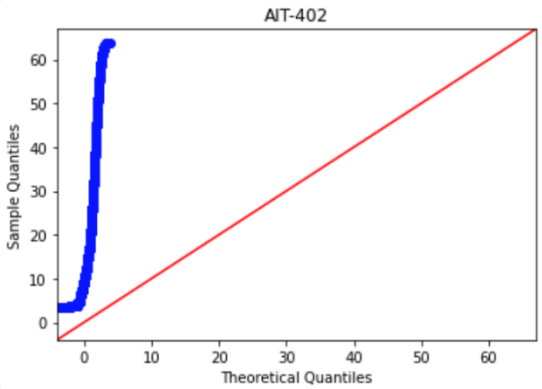
- multi-point attack on MV-201 and P-101 from 3:38:50 PM to 3:46:20 PM
- single-point attack on P-301 from 4:02:56 PM to 4:16:18 PM
- single-point attack of FIT-401 from 3:08:46 PM to 3:10:31 PM
- single-point attack of LIT-301 from 3:15 PM to 3:19:32 PM
- single-point attack of MV-501 from 3:54 PM to 3:56 PM
- single-point attack of P-601 from 3:26:57 PM to 3:30:48 PM

For the purpose of the business question of identifying system and variable-level attacks, we added a "Normal/Attack" column, with "0" s for Normal levels and "1" s for Attacks. Rather than specifying which variable was attacked at the given time, attack labels were applied to entire datapoint so that the model could first detect system-level attacks. Basic statistics for correlation and normal distribution (Q-Q Plot and Shapiro-Wilk test) were applied to understand the variable's data. From the correlation matrix, it implied that variable FIT-501 was inversely correlated to PIT-501, PIT-502, PIT-503, FIT-503, and FIT-504, as seen in Figure 1 below.



From the Q-Q Plot, all the variables did not exhibit normal distribution, as seen in Figure 2 and 3 below. A normal distribution would have the Q-Q plot blue line follow along the red line, but none of the variables had this sort of distribution.





For the Shapiro-Wilk test, if the P-Value of the Shapiro Wilk Test is larger than 0.05, we assume a normal distribution. If the P-Value of the Shapiro Wilk Test is smaller than 0.05, we do not assume a normal distribution. All of the variables had p-values smaller than 0.05, so we cannot assume a normal distribution for the variables. Even after standardization and normalization, the p-values were not high enough for us to consider a normal distribution.

Shapiro Wilk Test for Normalization, DF3

```
from scipy.stats import norm
from scipy.stats import shapiro

# If the P-Value of the Shapiro Wilk Test is larger than 0.05, we assume a normal distribution
# If the P-Value of the Shapiro Wilk Test is smaller than 0.05, we do not assume a normal distribution

for i in range(len(df3.columns)):
    print(str(df3.columns[i]))
    print(shapiro(df3.iloc[:,i]), "\n")
```

```
AIT-303
ShapiroResult(statistic=0.6673246622085571, pvalue=0.0)

DPIT-301
ShapiroResult(statistic=0.6725665330886841, pvalue=0.0)

FIT-301
ShapiroResult(statistic=0.6498887538909912, pvalue=0.0)
```

Shapiro Wilk Test for Normalization, DF5 (after normalization and standardization)

```
# If the P-Value of the Shapiro Wilk Test is larger than 0.05, we assume a normal distribution
# If the P-Value of the Shapiro Wilk Test is smaller than 0.05, we do not assume a normal distribution

for i in range(len(df5.columns)):
    print(str(df5.columns[i]))
    print(shapiro(df3.iloc[:,i]), "\n")
```

```
AIT-303
ShapiroResult(statistic=0.6673246622085571, pvalue=0.0)

DPIT-301
ShapiroResult(statistic=0.6725665330886841, pvalue=0.0)

FIT-301
ShapiroResult(statistic=0.6498887538909912, pvalue=0.0)
```

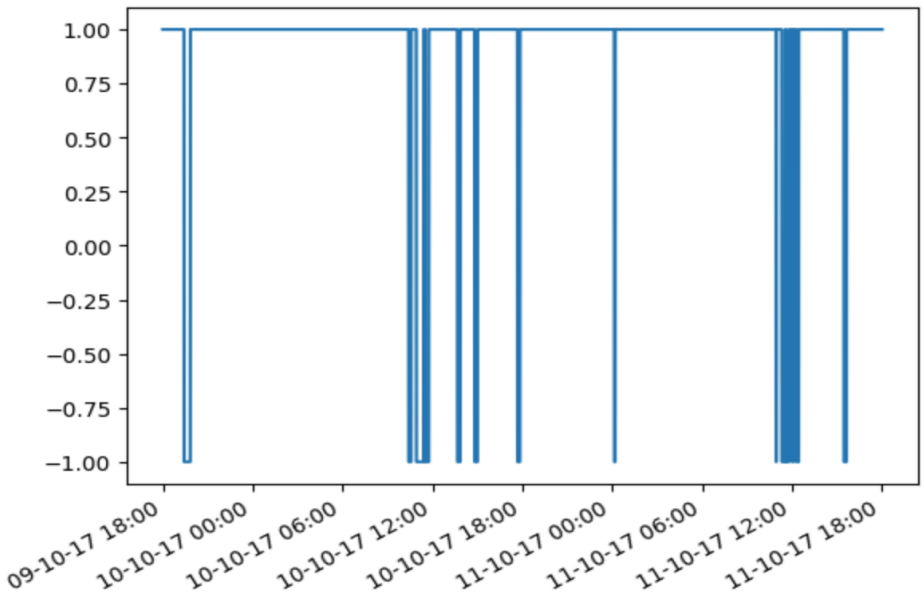
WADI™ 2017

The WADI™ 2017 presented data from a complete and realistic water treatment, storage and distribution network. The data set is comprised of signals from sensors and actuators sent through the communication network of the WADI test bed. There are two data frames included as part of the set. The 'normal' data set contains no cyberattacks and the 'attack' data set has 15 attacks.

The normal data set is 784,571 rows x 130 columns, where the rows represent time and the columns represent the different variables of the WADI system. The normal data set represents 14 continuous days of normal operation of the water distribution system. Of the 130 variables of the data set, one column was used for index, 'Row', and two were for time, 'Date', 'Time'. The other 127 columns represented signals from the sensors or actuators. The variables contain either continuous (71) or categorical (56) data. The categorical variables such as those with suffixes '-P', '-MV', are associated with signals that represent status such as open/close, pump on/off, etc. These variables are factorized in the original data set. The variables with continuous data are from sensor signals. The variables such as those with suffixes 'LT', 'FIT', 'AIT', 'PIT' represent tank levels, flow rates, physiochemical signals (pH, turbidity, conductivity) and pressure.

Variables can be considered part of three groupings, 1, 2 or 3. These groups represent the subsystems of WADI. They are P1 – Water Supply, P2 – Distribution Network, P3 – Return Water System. The variable's prefix indicates which subsystem they belong in, e.g. 1-AIT-001 is the conductivity detector signal from the P1-Water Supply subsystem. The understanding of the subsystems and the relationship between signals plays an important role when applying anomaly detection.

The attack data set is 172,803 rows x 131 columns. The data presentation is similar to the normal data set described above with regards to time and variables. The additional variable of the attack data set are the labels indicating when attacks occurred. These labels are '1' for normal and '-1' to indicate an attack. Over the time period of 48 hours, 15 attacks occurred.



Both normal and attack data sets contain approximately 3% NaN. The normal data set has 8 columns that contain NaN while the attack data set has 130 columns with NaN. The number of NaN to span the entire attack data set comes from two rows at the tail of the data frame. Overall the normal and attack data frames are relatively complete data sets. NaN's were removed prior to normalization.

Supplemental to the data set is a description of the attacks that occurred. The document describes points of attacks, date and time. The information from this document is used to tune anomaly detection approaches on subsystems (further description in the following sections).

DATA CLEANING AND TRANSFORMATION

SWaT™ 2015

For the SWaT 2015 dataset, the follows steps were taken to clean and transform the data for further analysis:

- SWaT_Dataset_Normal_v0 & SWaT_Dataset_Attack_v0 were combined to create a SWaT Dataset Combined file encompassing the total 11 day period with both normal and attack periods included
- Dummy variables were made for the Normal/Attack column. The Normal/Attack values were translated to a numeric type where a Normal state was equal to 0 and an attack state was equal to 1 for ease of data processing
- For transformation and normalization, the sklearn.preprocessing in Python was utilized. Before analysis, the data was fitted and transformed via the StandardScaler, fit, and transform functions.

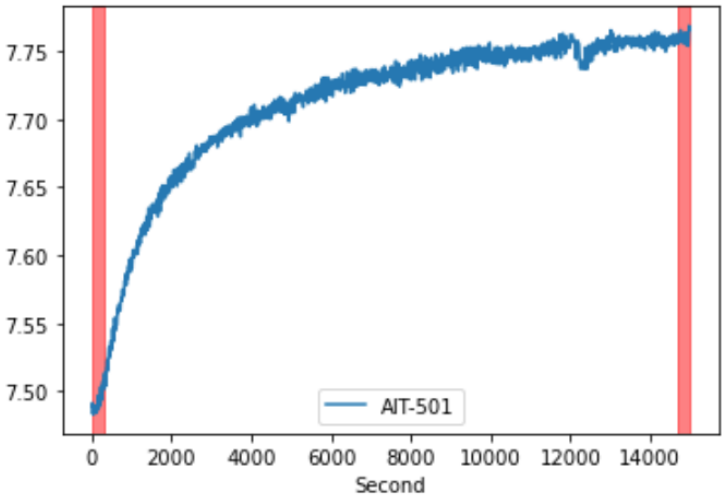
SWaT™ 2019

For SWaT 2019 v2, the data cleaning methods include:

- Renaming variables to match that of "A Dataset to Support Research in the Design of Secure Water Treatment Systems" nomenclature
- Converted the timing of GMT +0 to GMT +8 by adding 8 hours,
- Adding a "Second" column to count the number of seconds from the system recording start time of 12:30 PM
- Labeled variables as either sensor or actuator (for our own reference)
    - For actuators, changed "Inactive" values to "0" and "Active" values to "1"

- Variables that were stagnant (did not change in value) were removed, as seen below

- Depending on our tests, some of them we removed ramp up, ramp down periods, or both to see how they impacted model performance.



For SWaT 2019 v2, the transformation methods include:

- Normalization of data to scale in range [0,1]
- Standardizing data

WADI™ 2017

Data preparation of the WADI data set consisted mainly of removal of NaN, formatting of time and normalization of variables. These data munging activities are driven by the types of anomaly detection techniques used as described in later sections.

The Date and Time columns of the attack data set were formatted to allow proper plotting of time series. The Time column data is sequential in steps of seconds. However, the time is not sequentially presented in hours, i.e., 00:00 to 24:00 hour time period. Instead, the time column is presented in

cycles from 00:00.0 to 59.59.0, repeated 24 times for a full day. The only full 24-hour day for the attack data set is 10/10/17. The starting time of the data set had to be manually calculated and was determined to be 10/9/17 18:00. The ending date and time of the attack data set is 10/11/17 18:00.

For the date and time format of the normal data set, no processing was performed. The reason being is that time only had to be tracked to identify the location of attacks, which are absent in the normal data set.

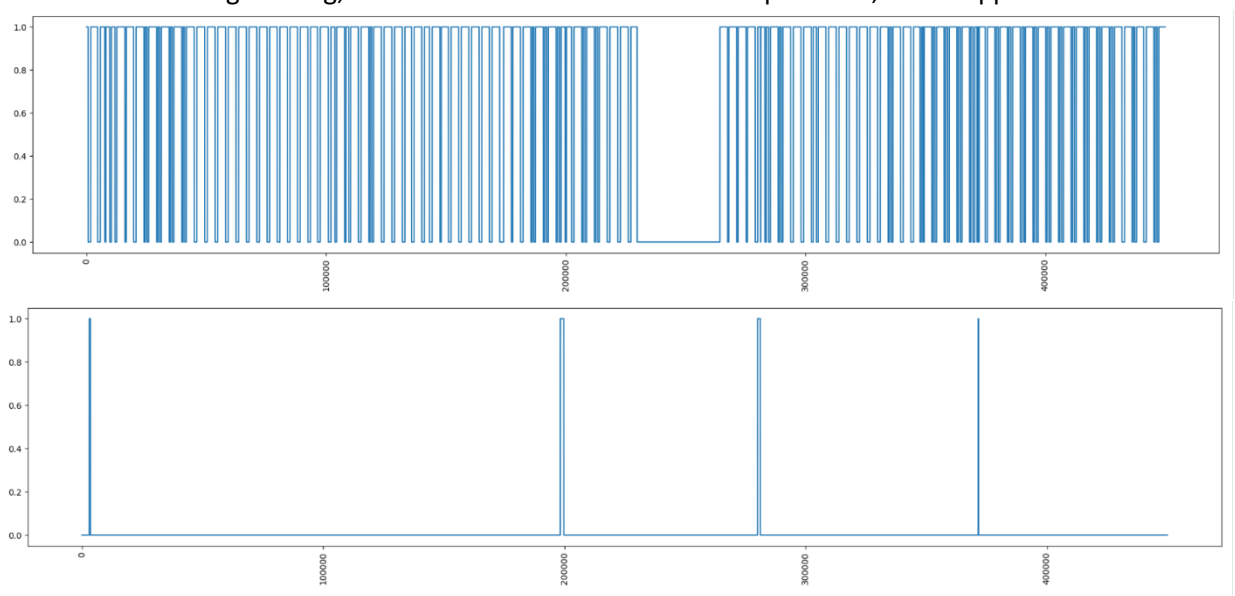
Continuous data in the attack data set vary in magnitude, which had to be handled in order to use some anomaly detection techniques. Variables with continuous data were normalized using sklearn's MinMaxScaler preprocessing library. All but Isolation Forest required normalization of data.

Variables with no data or 1 unique data point were removed. This is only advisable if training and testing are done with the same data set. These variables were removed as they do not provide useful information when building the models that are intended to classify two factors (e.g. normal vs attack). For the attack data set, this reduced the variable count to 96 variables (decrease of 31 variables). One must be cautious when removing variables with only one unique value. It was observed in the SWAT 2015 data set, a variable with one unique value in the normal data set may have more than one unique value for the equivalent variable in the attack data set.

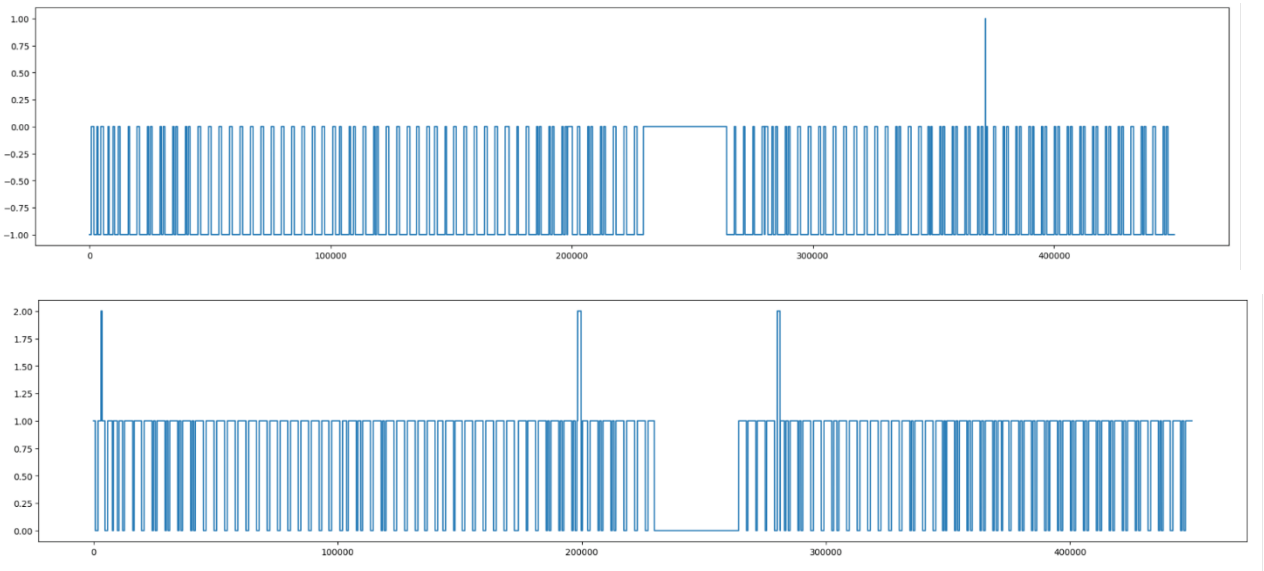
A data preparation technique that was evaluated was to group variables by their respective subsystems and run the anomaly detection algorithm. For example, a first attempt at K-Means used the entire data set. This approach made no assumptions on the correlation of the variables. However, if we assume that variables within subsystems have a greater chance of correlation, then grouping them together to perform K-Means would enhance the algorithm's chance to find anomalies (discussed further in sections below).

While the data munging techniques discussed above provided useful outputs needed to perform anomaly detection, knowledge and understanding of water systems could have greatly helped the data preparation process. For example, the SWAT 2015 variables P101 and P102 appear to exhibit a correlation that neither pump can be 'on' at the same time. This knowledge was determined from the description of the attacks. By applying this knowledge, attacks can be found more easily by creating variables using the sum and differences of the original signals.

Without feature engineering, P101 and P102 show ‘on’ and ‘off’ positions, which appear normal



With feature engineering, the sum and difference of the variables exposes attacks:



As both the normal and the attack dataframes from the WADI data set are fairly complete, no imputation of data was performed.

APPROACH AND METHODOLOGY

Numerous approaches for anomaly detection as well as attempts at novelty detection were performed. Long Short-Term Memory and supervised machine learning techniques were initially evaluated but eventually abandoned either due to lack of accuracy, overfitting or processing constraints. The team moved its focus to dimension reduction followed by non-supervised techniques as its anomaly detection approach. A two-phase approach using K-means and CUSUM for anomaly detection was the eventual focus of the team's efforts.

Initial Approaches

Long Short-Term Memory (LSTM)

LSTM was a candidate for anomaly detection as it is well suited for classification based on time series data. LSTM is a subtype of recurrent neural network. The main advantage of LSTM vs 'vanilla' RNN is that the former allows for past information to be processed in the network for a longer length of time. The network effectively holds information for earlier values of the time series. The usefulness of this characteristic of LSTM is that the network can perform backpropagation through time. The system can move backwards and calculate derivatives, e.g. the evolving error between real output and predicted output. Afterwards, the network uses Gradient descent to adapt its weights and decrease prediction error.[1]

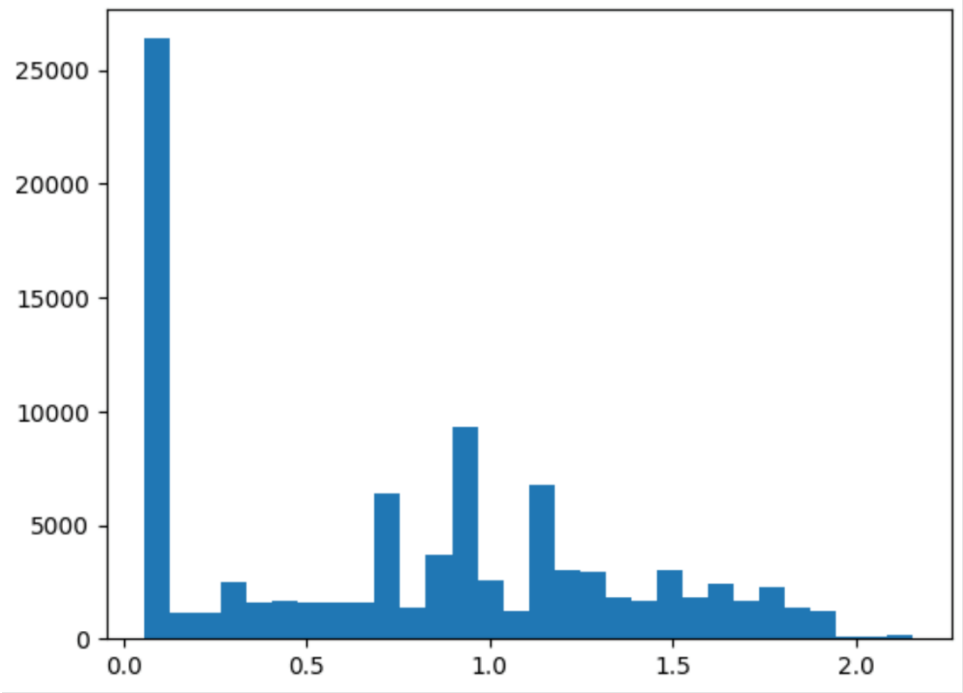
For the evaluation of LSTM, training and testing were both performed on the attack data set. As this was the initial phase of the project, feasibility of using LSTM was prioritized vs tuning and refining. The following parameters were assessed on the variable 1_FIT_001_PV.

LSTM Parameters	
Hidden Layers	4
Dropout Rate	0.2
Epochs	1
Batch Size	120

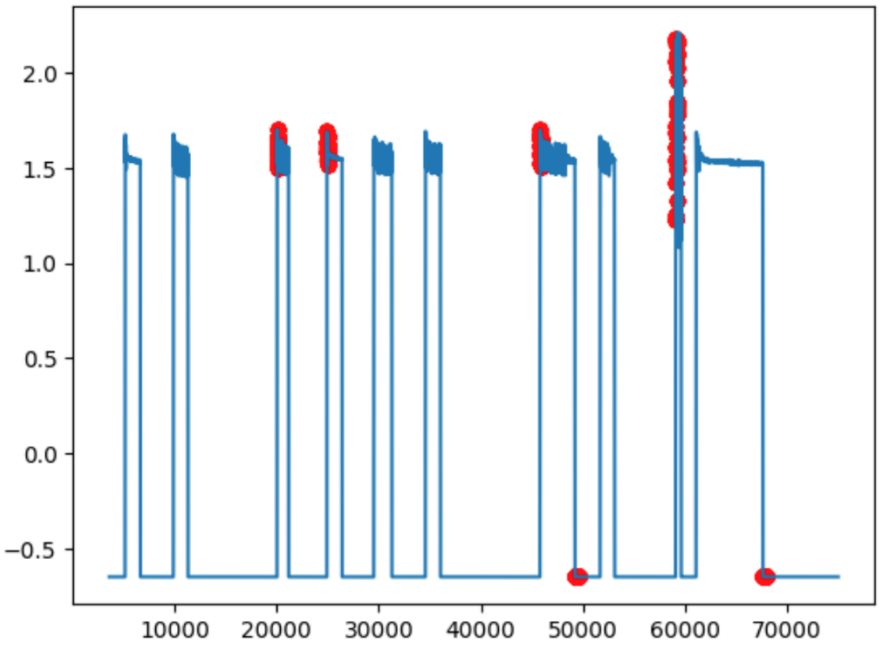
Training of the LSTM model was performed on a segment of the attack data set where no attacks occurred. The model was then used to predict on the exact same segment of training data. The errors were plotted to find the appropriate threshold which is later used in the test data set.

On this run through of the model, the mean absolute error (MAE) was plotted in a histogram. The chosen threshold which will be used for anomaly categorization was at 1.9.

[1] Nguyen Thi, N., Cao, V.L., Le-Khac, NA. (2017). One-Class Collective Anomaly Detection Based on LSTM-RNNs. In: Hameurlain, A., Küng, J., Wagner, R., Dang, T., Thoai, N. (eds) Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVI. Lecture Notes in Computer Science(), vol 10720. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-56266-6_4

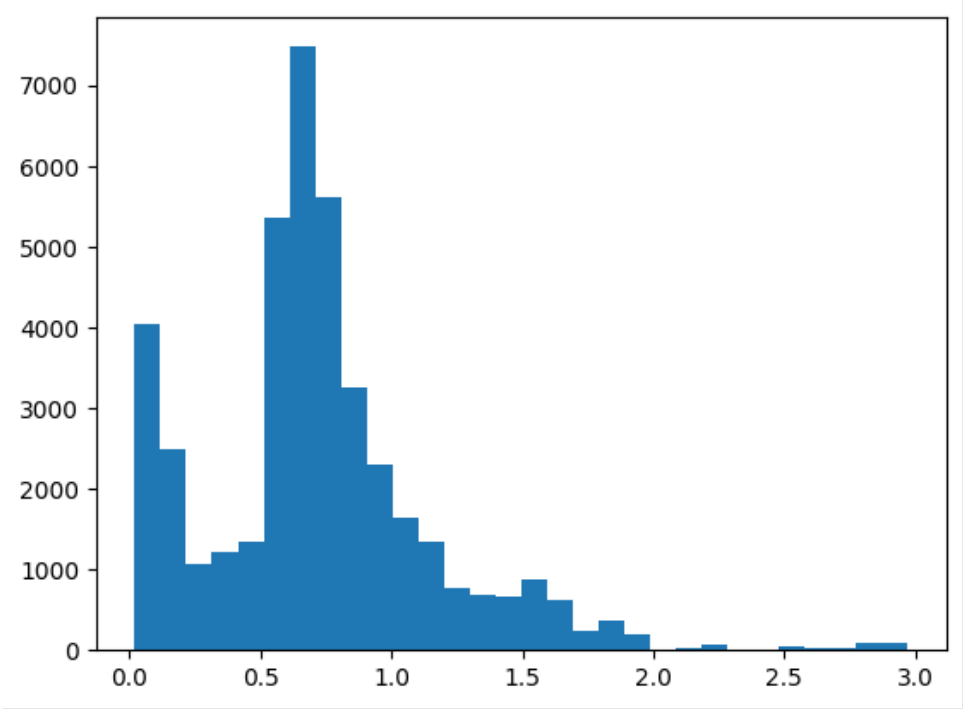


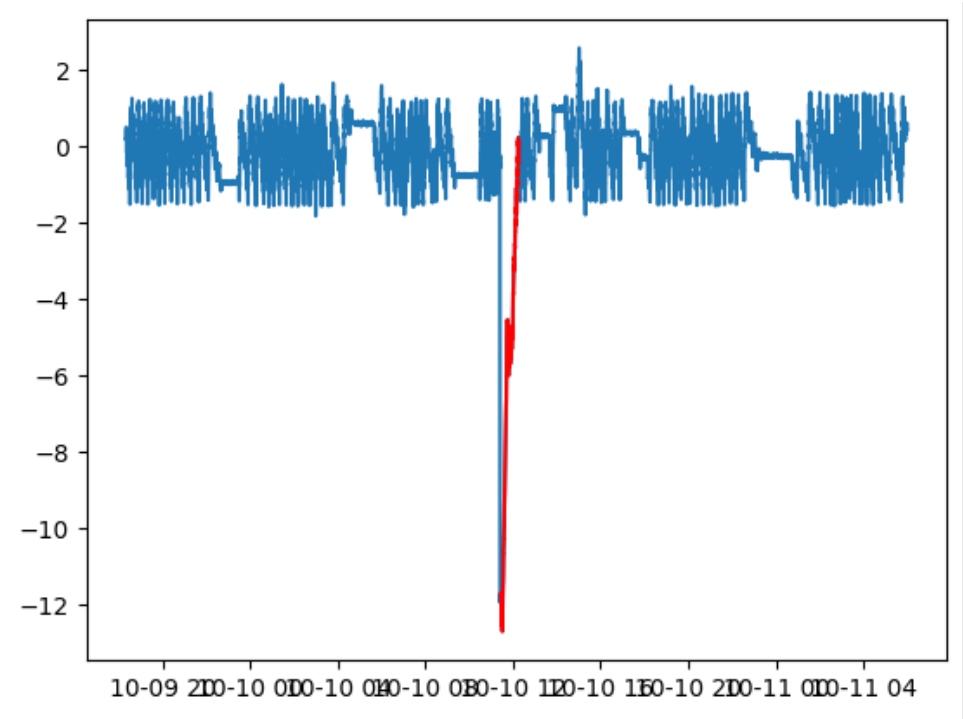
The LSTM was then performed on the test data set which contained the attack. MAE was calculated from the LSTM prediction and the actual test data. MAE greater than the 1.9 threshold was flagged as an attack.



This instance of LSTM was able to find the attack. However, false positives at x-axis of approximately 20000, 25000, 47000, 50000 and 70000 were also flagged.

Using the same parameters, LSTM showed promising accuracy with tank level reading '2_LT_002_PV'. The chosen threshold which will be used for anomaly categorization was at 2.2.



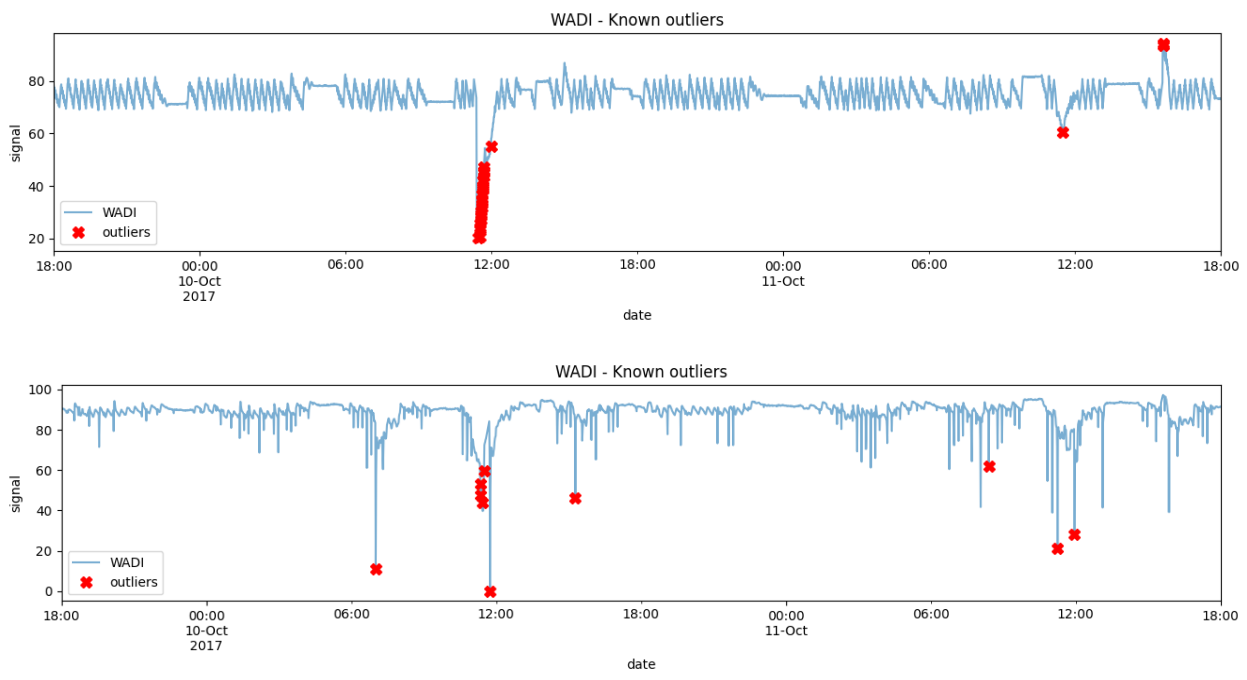


The feasibility of LSTM for anomaly detection is demonstrated, but as previously mentioned, the approach was abandoned. The main reason is the long durations it took to train and deploy a model on the train and test data set. The parameters listed above were the outcome of trying to reduce the processing time. Specifically, the 'Hidden Layers' and 'Epochs' influence the processing duration. First attempts at LSTM took about 20 to 40 minutes. It was eventually reduced. However, the performance of LSTM subsequently declined with a less dense neural network. Additionally, the sequence size (not

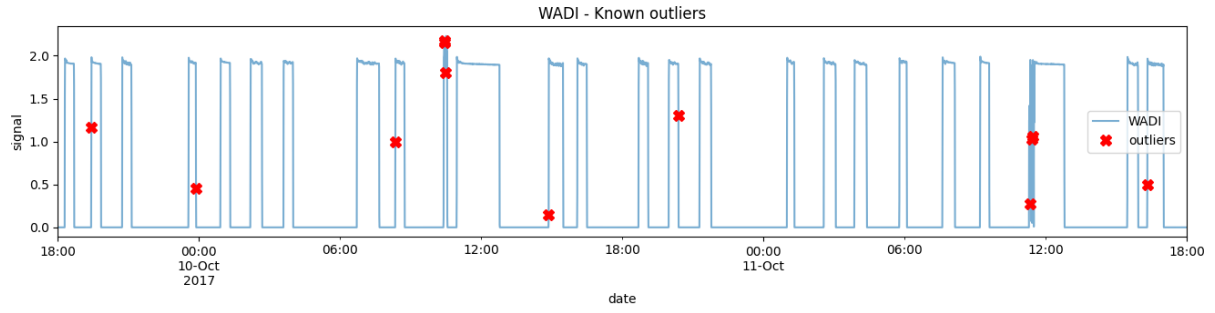
an LSTM parameter but rather a data processing step) cannot be too large. The sequence size is the number of previous steps to predict the next step. A large sequence size contains more information for prediction, but uses too much resources. A small sequence size appears to lead to lower accuracy due to the limited information available for prediction.

KNN For Outlier Detection

As the approach using LSTM appeared promising, albeit resource heavy to execute, another approach to detect anomalies in time series data was evaluated using KNN. Unlike LSTM which incorporates neural networks, KNN is a distance-based algorithm. KNN anomaly detection was performed similar to LSTM, on a single variable, as opposed to the entire data set at once. The parameters used for the initial trial of LSTM assumed 10% outliers and k-nearest neighbors = 2, naively accounting for the two categories, normal and attack. Results were promising for some variables with continuous data such as tank level readings '2_LT_002_PV' and pressure readings '2_PIT_002_PV'.



However, performance of KNN was not as promising for variables that exhibit a more structured (e.g. stepwise) signal, such as '1_FIT_001_PV'. While it was able to flag the area of attack, there were more false positives than the other variables described above.



The faster and relatively accurate detection of attacks using KNN makes this a viable means for anomaly detection. However, the team eventually moved to CUSUM as an anomaly detection for single variables.

Supervised Machine Learning Models

For our early data analysis, we attempted a few supervised machine learning models just to see if any of them were efficient in predicting system-level attacks. A few of the models we attempted were Logistic Regression, KNN, SVM, Gaussian Naïve Bayes, Decision Tree, and Random Forest. Logistic regression estimates the log likelihood of a prediction, given independent variables. KNN is a supervised learning approach that predicts the classification of the new data point, based on previously labeled data point classifications. Support vector machine is another supervised learning approach that takes labeled datapoints and creates a hyperplane that best separates the datapoints. The new datapoint to predict will fall on one side of the hyperplane for classification. Naïve bayes is a supervised classification model that assumes linear independence of features. It predicts the conditional probability of the target vector for the given feature vector and chooses the label with the highest probability. Our gaussian naïve bayes assumes normal distribution and feature independence, which might have been reasons why it was not good at predicting attacks. Decision tree creates a classification tree of internal nodes and decision leaves to label the datapoints as attacks or normal. Though decision trees are good for both numerical and categorical data, it is sensitive to small variations in data and is often subject to overfitting from the training data. Random forest creates a number of decision trees, each creating its own prediction, and the prediction that occurs the most is the model's prediction.

Confusion Matrices

```
Results for Logistic Regression:

cm:
[[2515   79]
 [ 190  215]]

lr_acc:  0.9103034344781594
```

```
Results for SVM:

cm:
[[2550   44]
 [ 196  209]]

svc_acc:  0.9199733244414805
```

```
Results for DTC:

cm:
[[2593    1]
 [   4  401]]

dtc_acc:  0.9983327775925308
```

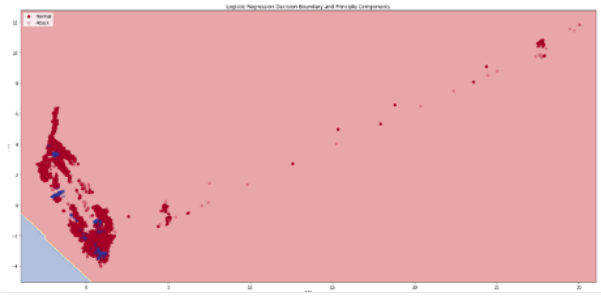
```
Results for KNN:

cm:
[[2590    4]
 [   8  397]]

knn_acc:  0.995998666222074
```

```
Results for GNB:

cm:
[[1164 1430]
 [   0  405]]

GNB_acc:  0.5231743914638213
```

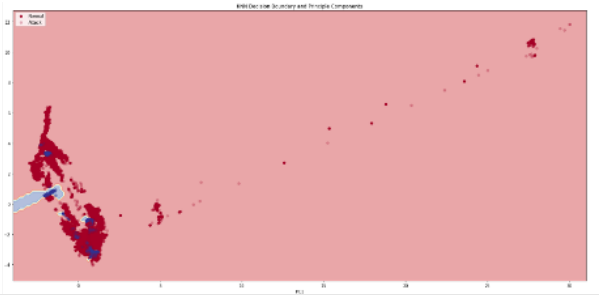
```
Results for RF:

cm:
[[2592    2]
 [   2  403]]

rf_acc:  0.9986662220740247
```

Unfortunately, for our supervised machine learning models, they were receiving abnormally high accuracies around 98%-99%. These high accuracies indicate that there might have been high amounts of overfitting to the training data, which would not be useful in classifying attacks of a new dataset.

Logistic Regression



KNN



Even the decision boundaries of the models could not produce accurate or helpful images of how the models were predicting the datapoints. The decision boundaries plot the first two principal components against each other to show how each model classifies the test data. In the end, we decided not to use any supervised learning models, as they were providing unreasonably high prediction rates and would not have been as applicable to the problem statement, since future datasets would not have labeled data.

Principal Component Analysis and K-means Clustering

K-means clustering for outlier detection was initially considered, given the objective to find identify two distinct states in the dataset: normal state vs. an attack state. K-means clustering was used to cluster the dataset and identify data points that were outliers and behaved differently than the behavior characterized by their respective clusters. The initial prototypes of our implementation sought to identify these outliers and assess if these outliers corresponded to attack states, indicating these outliers behaved differently than normal as expected from a system under attack.

Given the high dimensionality of the datasets, principal component analysis was also implemented to aid in visualizing and clustering the dataset. Initial results from this approach showed favorable accuracy metrics and proved to be a viable solution.

K-means of Subsystems

In addition to PCA, a different approach for incorporating dimensionality reduction using K-means is the grouping of variables per sub-system. This approach assumes that correlation between variables are higher when part of the same subsystem. Therefore, an attack on a variable will have a higher chance of manifesting in an anomalous signal in the subsequent or prior processing step of the water system. In other words, a cause and effect will have a higher chance to be observed within the same subsystem (e.g, a malicious attack on a water valve '1_MV_001_STATUS' will have impact on the flow, which should be visible in the signal '1_FIT_001_PV').

This approach of grouping subsystem variables, and then perform K-means, was executed on WADI P1 and P2 subsystems. The data frame was sliced to only include the variables belonging to each subsystem. After data normalization, K-means was performed assuming 2 clusters. To assess the accuracy, the labeling of attacks was masked in such a way that attacks only on the specific subsystems were counted towards accuracy. For example, when performing K-means on subsystem P1 variables, all the attacks on P2 variables were masked as normal.

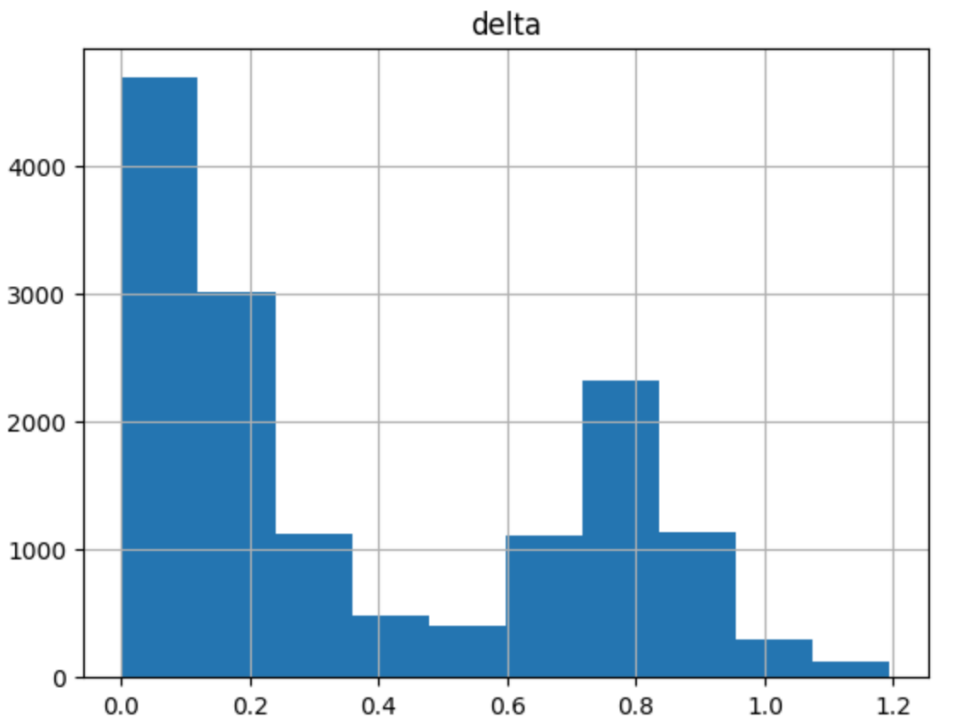
The results for K-means on subsystems are mixed. For P1, the overall accuracy is 74%, sensitivity is 73% and specificity is 74%. For P2, the overall accuracy is 43%, sensitivity is 47% and specificity is 43%. It is not clear why there is a difference in accuracy when using the same approach on the P1 and P2 subsystems. Many of the attacks occurring in the P2 subsystem are focused on valves, e.g. '2_MCV_101', '2_MCV_201', etc. It is hypothesized that K-means is not effective at clustering attacks for these categorical variables.

K-Means for Novelty Detection

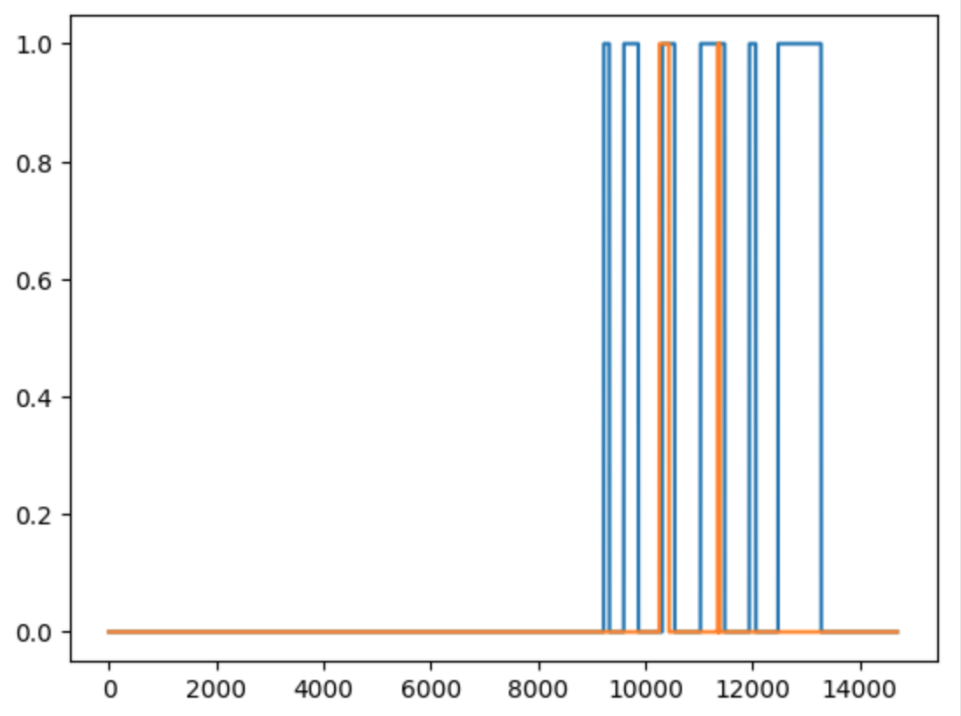
As the team moved towards K-Means for unsupervised detection of anomalies, other approaches using K-Means were evaluated. K-Means for novelty detection was possible as the details of where attacks occurred were provided. Knowing this, training can occur on portions of the data that are not

contaminated by outliers. When the trained model detects a 'new' observation, it is considered an outlier. In this context an outlier is also called a novelty.

For this approach, K-Means was used on a segment of the data set that does not contain attacks. The strategy is to use K-Means to find the centroid for the 1 cluster of 'clean data'. The next step is to determine the Euclidean distance of the centroid. For the entire data set, find the Euclidean distance of each row entry. Subtract these values from the centroid Euclidean distance. Plot a histogram of differences and determine a threshold to separate outliers. In this case of the SWAT 2019 data, the threshold was chosen at > 1.0.



The rows with Euclidean distances greater than 1.0 were finally flagged as outliers, i.e. attacks. Accuracy is at 87%. However, the sensitivity of this approach, the ability to find true positives, is low. Only 7% of the attacks were identified (see below, blue – true, orange – predicted). Tuning the novelty detection by adjusting the threshold is possible. However, the balance of accuracy and specificity was found to suffer by increasing sensitivity.

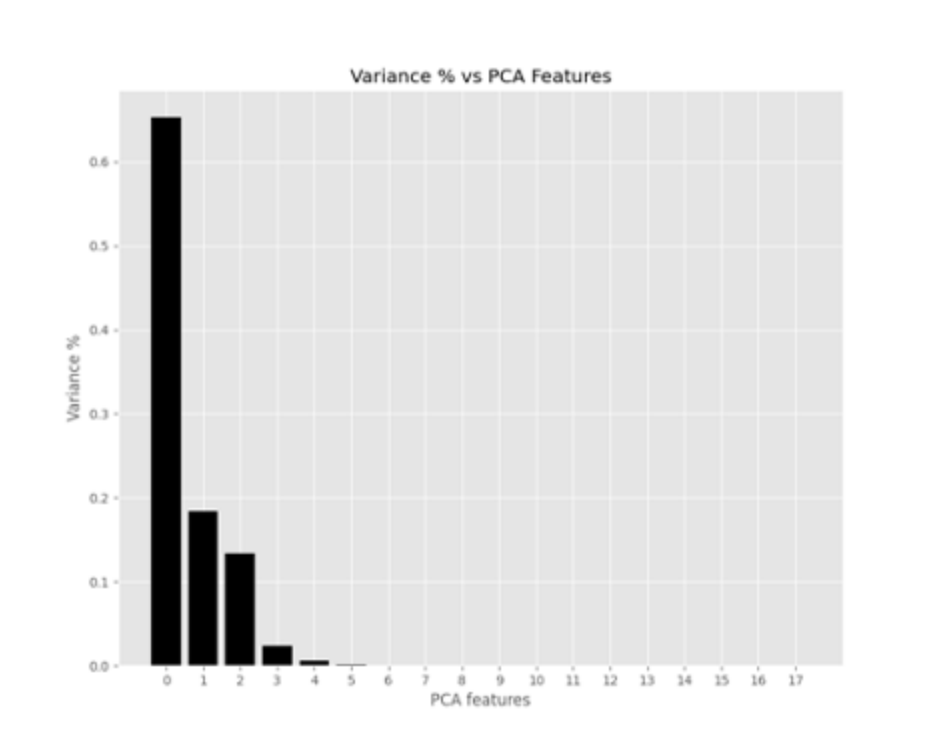


Implemented Approaches

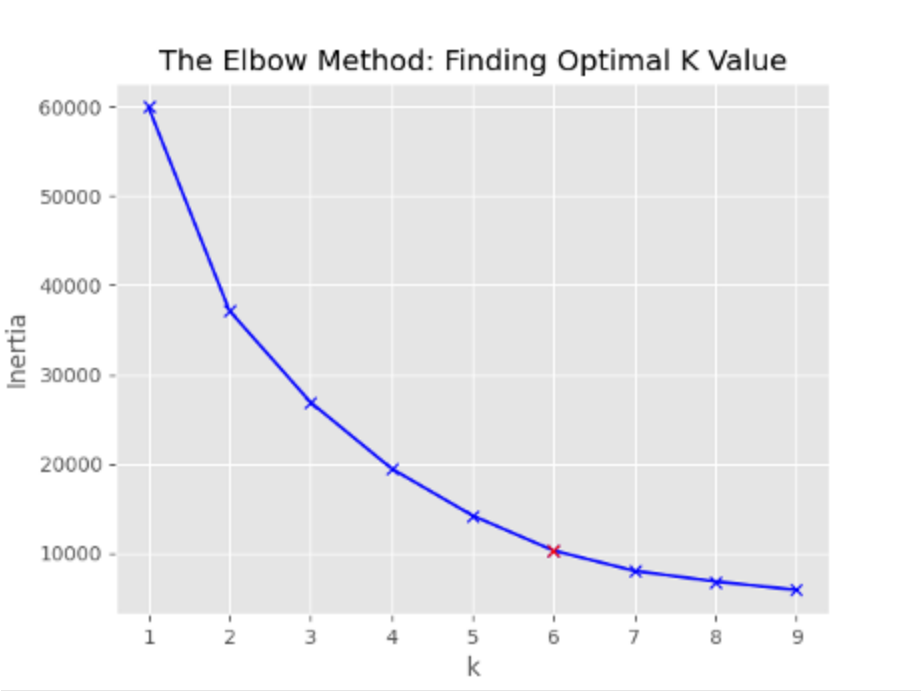
Principal Component Analysis and K-means Clustering

Analysis of the various datasets showed that each dataset contained many features. Given the high dimensionality of the dataset, principal component analysis was implemented on the cleaned datasets to decompose the data. The PCA function via the sklearn python library was utilized to transform and decompose the data, resulting in a data frame with the respective datasets principal components.

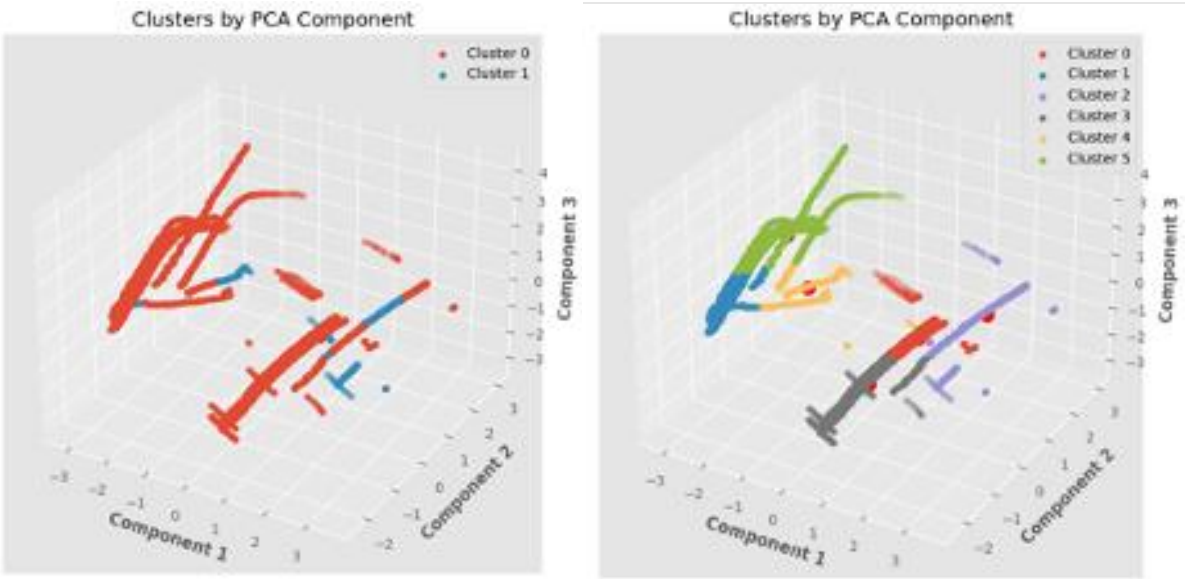
The resulting data frame also allowed for visualization of the dataset to better understand the data and to decide what parameters to implement in the K-means algorithm. For example, to understand how many principal components to include in the K-means model, a variance vs. PCA features bar plot was produced to understand how much variance would be accounted for with the inclusion of each principal component. In general, 80-90% variance was the desired value for represented variance which typically resulted in 3-5 principal components included for clustering.



With methodology to assess how many principal components to include in clustering, similar methodology was utilized to identify the optimum number of clusters to execute K-means clustering with for each dataset. To find the optimum number of clusters (k), the preliminary K-means models were initialized to visualize the inertia vs. number of clusters (k) and select potential k values. Per the Elbow Method, the general objective was to select k values ranging from where the largest inertial drop off occurs to where the slope of inertia becomes linear. In the example below, the candidate k values range from a k value of 2 to 7 and ultimate a k=6 value was selected. The final selected k value depends on the ultimate accuracy assessed of the models.



In addition to k value selection, the preliminary K-means models allowed for visualization of the clustered principal component data. To build a K-means model and produce these visualizations, the KMeans function via the sklearn library was utilized. These visualizations aided in the characterization and understanding of the behavior of the datapoints.



In the figure above, the left image shows the first three principal components plotted where Cluster 0 represents a normal state and Cluster 1 represents a system under attack. The right image shows the same first three principal components where clusters 0 through 5 represent how the K-means algorithm clustered the data points.

K-Means for outlier Detection

For optimum clustering, the dataset was broken into individual subsystems per feedback from the midterm report. Therefore, for each subsystem within the dataset, an induvial K-means model was fitted. For each model, several variables were tuned to find models with acceptable accuracies. Namely, the number of clusters (k), the number of PCA components included in clustering, and the threshold value (T) for outlier detection were the key values varied for each model.

Once the dataset was clustered, the Euclidian distance of each dataset with respect to its cluster centroid was found. The standard deviation and mean of all distances for all datapoints within a cluster were found. The standard deviation was multiplied by the threshold value T and added to the average distance to the centroid value to find the overall threshold that defined an outlier. If the Euclidian distance of a datapoint to its respective centroid was beyond this value, the datapoint was considered an outlier. These outliers were then regarded as attacks on the subsystem as the distance away from the centroid indicates the behavior of this datapoint is outside of the "normal" behavior of clustered datasets as defined by the threshold. Overall, this concept was inspired by the concept of Bollinger Bands in financial technical analysis.

The formulas defining the overall threshold are below. The np represents the functions used that were primarily from the NumPy python library.

$$STD = np.std(distances\ to\ centroid\ for\ datapoints\ in\ cluster\ k)$$

$$AVG = np.mean(distances\ to\ centroid\ for\ datapoints\ in\ cluster\ k)$$

$$MSTD = STD * T$$

$$Threshold = AVG + MSTD$$

$$Outlier = Distance\ to\ Centroid\ for\ individual\ datapoint > Threshold$$

To tune each K-means model, several test cases where the cluster (k), principal component, and threshold (T) values were varied were executed. The overall accuracy of each model was compared and the test case with the highest accuracy and most desirable confusion matrix was ultimately the one selected as the final K-means model. An example of test cases and how these values were varied is below with respect to the P1 system from the SWaT 2019 dataset.

Case	Clusters (k)	PCA Comp.	Threshold (T)	True Positive	True Negative	False Positive	False Negative	Accuracy
1	5	5	2.5	12765	216	245	1770	0.865630
2	6	5	2.5	12839	216	171	1770	0.870565
3	7	5	2.5	12839	216	171	1770	0.870565
4	7	3	2.5	12857	201	153	1785	0.870765
5	7	4	2.5	12828	199	182	1787	0.868698
6	7	5	2.5	12839	216	171	1770	0.870565
7	7	3	1.5	12409	218	601	1768	0.842024
8	7	5	1.5	12060	242	950	1744	0.820352
9	7	3	2	12857	201	153	1785	0.870765
10	7	5	2	12839	216	171	1770	0.870565

11	7	3	3	12916	10	94	1976	0.861963
12	7	5	3	12878	26	132	1960	0.860496
13	7	3	1.25	12409	218	601	1768	0.842024
14	7	5	1.25	12060	242	950	1744	0.820352
15	3	5	2	12009	15	1001	1971	0.801813

In the above example for subsystem P1 from the SWaT 2019 dataset, Case 9 with the combination of k =7, PCA components = 3, and a threshold value of T=2 produced the best accuracy and highest true negative rate, therefore, the final model was fitted with these tuned parameters.

The overall accuracy for test cases was calculated via a confusion matrix and the definition of each case as it relates to the output of the K-means models and the original normal/attack labels provide are as follows:

Parameter	**Definition**
True Positive (TP)	Not identified as outlier AND is not an attack per original labels
True Negative (TN)	Identified as outlier AND is an attack per original labels
False Positive (FP)	Identified as outlier AND is not an attack per original labels
False Negative (FN)	Not identified as an outlier AND is an attack per original labels

CUSUM

For our model, K-means is a good method to detect outliers as system-level attacks. It spans all variables across the different sections of the water treatment system. However, it is also important that our model can detect the variables that are under attack. We decided that for variable-level attacks, we would implement CUSUM anomaly detection to recognize when a large enough change would occur to the variable, it would flag the change. The CUSUM model was based off of "Adaptive Filtering and Change Detection" by Fredrik Gustafsson[2] and the work by Marcos Duarte[3]. The main equation used for the CUSUM change detection is:

$$g_t = g_{t-1} \pm s_t - v$$

- s_t is the distance measure of measurements up to index t minus measurements up to index t-1
- v is the drift parameter that prevents change in the absence of change. To prevent positive drifts, eventually yielding a false alarm, a small drift term v is subtracted at each time instant. To prevent a negative drift, which would increase the time to detection after a change, the test statistic is reset to 0 each time it becomes less than a negative constant a.

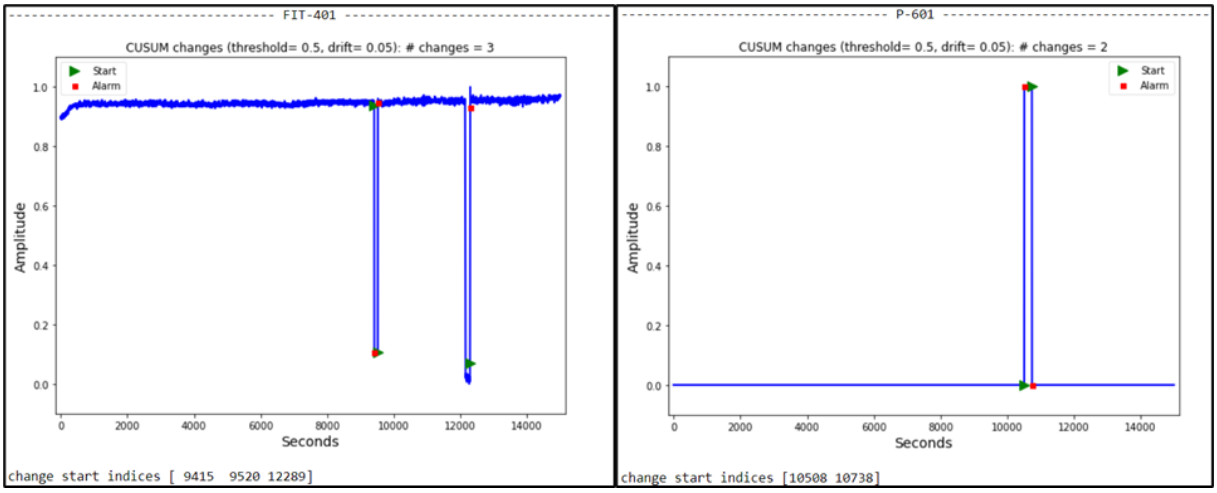
[2] Gustafsson, F. (2001). Adaptive filtering and change detection. https://doi.org/10.1002/0470841613

[3] https://github.com/demotu

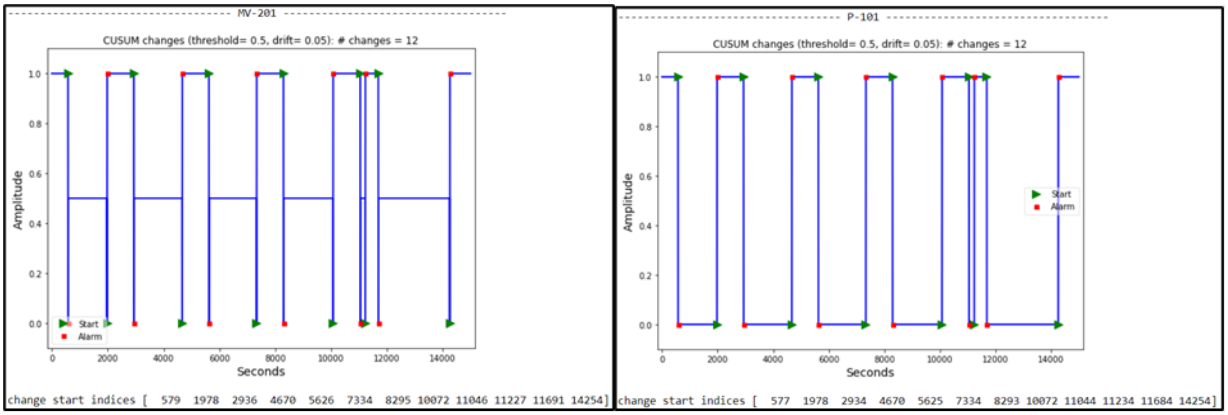
- g_t is an auxiliary test statistic used for alarms using threshold. An alarm goes off if g_t > threshold, indicating that the threshold has been exceeded in either positive or negative value.

Once a change has been detected from exceeding the threshold value, the start index of when the change initiates is saved. For our model, these time indices are important since they flag the initiation time when a variable will have a drastic change and will be used to cross-reference with the K-means anomalies.

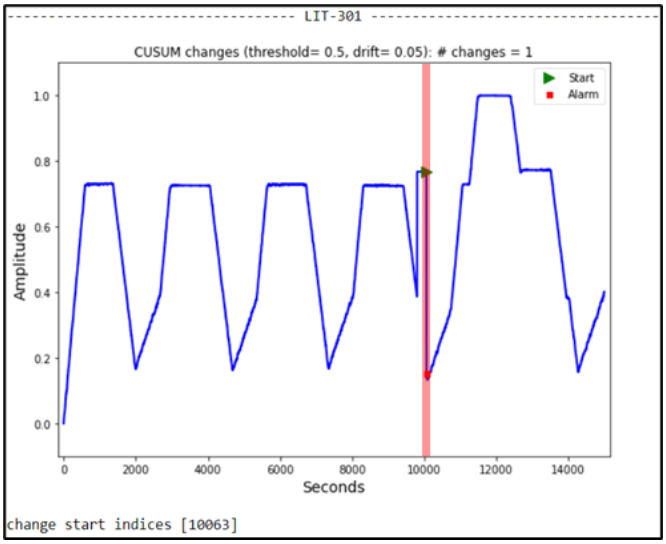
For the CUSUM to work on all the datasets, it needed to be applied to the normalized datasets, which ranged from 0 to 1. Otherwise, each variable would have differed range of amplitudes, but the same threshold value would be applied to each variable. Once the datasets were normalized, a little bit of parameter tuning needed to be performed on the threshold and drift values. Using SWaT 2019 as an example, we tuned the threshold value to 0.5 and the drift to 0.05. This means that whenever a change in amplitude of 0.5 (or 50%) of the normalized data occurred, a change would be detected. Tuning the parameters to the dataset is important, as this could drastically affect the model's accuracy on which variables it selects. Below are a few examples of the variables that experienced an attack in SWaT 2019. CUSUM worked well for variables that were constant but then had abrupt changes in their values, like in FIT-401 and P-601.



The CUSUM also worked for the actuator variables, since they alternated between 0 and 1 values throughout the test. However, CUSUM detected at every point that the actuator variable changed from 0 to 1 or 1 to 0. This would inevitably create more False Positive readings, but we also used these readings to cross reference with the K-means to detect both a system and variable-level change. This will be explained further in the report.

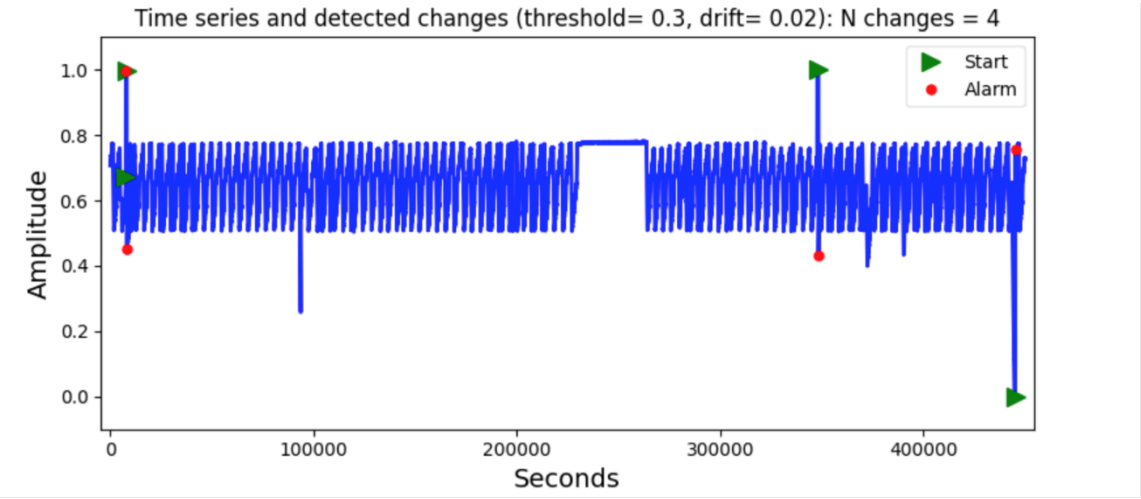


Lastly, CUSUM did a fair job at detecting changes in continuous, cyclic variables. CUSUM was surprisingly able to discern anomalies between the cycles, while still detecting appropriate changes that would exceed the threshold. In the instance of LIT-301, the attack occurred between 3:15PM – 3:19:32PM, which translates to the 9,900 – 10,172 second index. Out of all of the cycles, CUSUM picked out the one drop in value that coincided with the attack time.



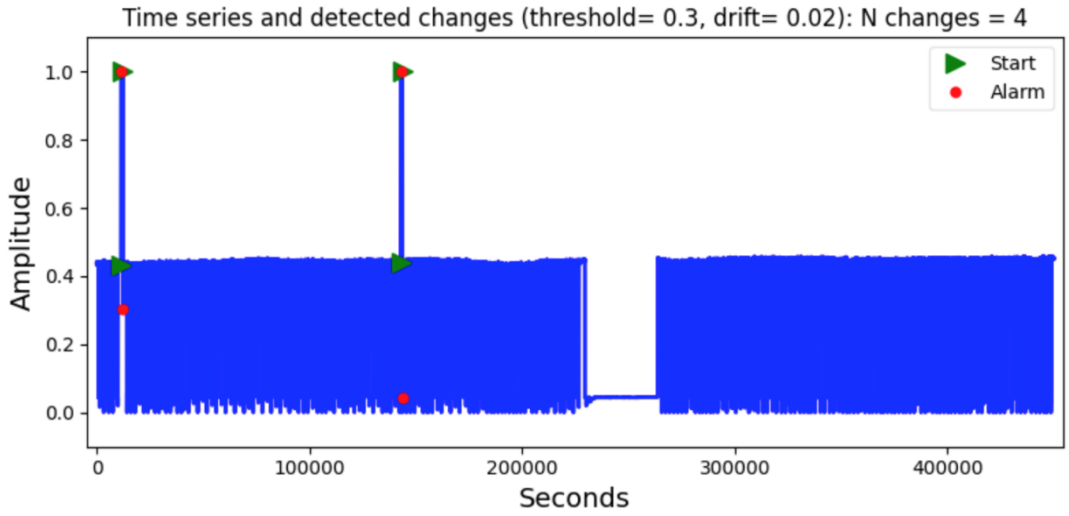
CUSUM continued to perform well on the SWAT 2015 data set. Although the general approach is the same as with the SWAT 2019 data set, the parameters were adjusted to accommodate the variable intensities of the signals. In other words, there is no global set of parameters that were used to process CUSUM for each variable. Although, there are general variables that appear to apply to many variables. For example, setting the threshold to 0.3 and drift to 0.02 were found to be suitable for the following variables, 'LIT301', 'DPIT301', 'FIT401', among others. The ability to detect attacks with these parameters for the variables can be observed as follows (* for each Timestamp indicates actual attack found by CUSUM):

LIT301



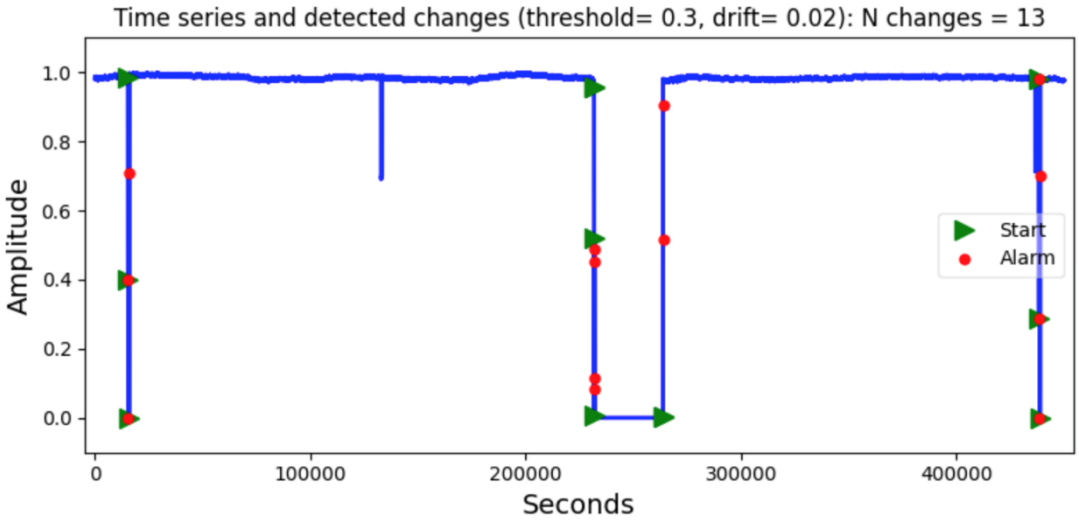
Timestamp ('2015-12-28 12:08:04'), Timestamp ('2015-12-28 12:15:12')*, Timestamp ('2016-01-01 10:46:35')*, Timestamp ('2016-02-01 13:41:11')*

DPIT301



Timestamp ('2015-12-28 13:09:44'), Timestamp ('2015-12-28 13:25:54')*, Timestamp ('2015-12-30 01:42:09'), Timestamp ('2015-12-30 01:53:30')*

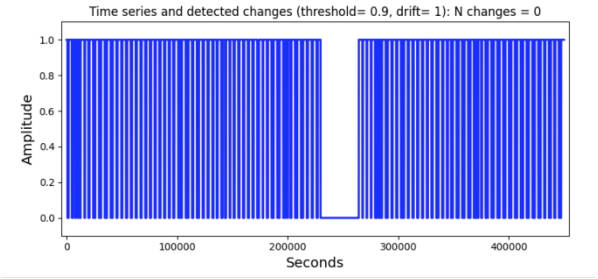
FIT401



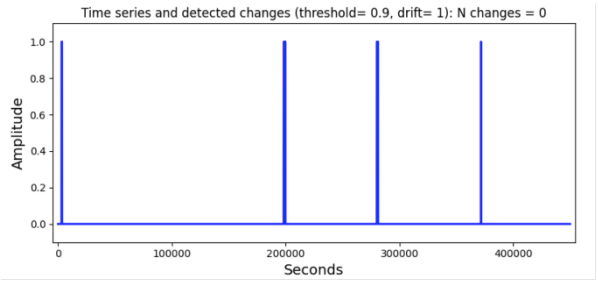
Timestamp ('2015-12-28 14:16:00')*, Timestamp ('2015-12-28 14:18:41')*, Timestamp ('2015-12-28 14:28:02')*, Timestamp ('2015-12-31 02:22:39'), Timestamp ('2015-12-31 02:22:39'), Timestamp ('2015-12-31 02:24:14'), Timestamp ('2015-12-31 02:24:21'), Timestamp ('2015-12-31 11:16:35'), Timestamp ('2016-02-01 11:44:24')*

The ability of CUSUM to detect anomalies can be aided with simple feature engineering. Take for example two pumps P101 and P102. Without previous feature engineering, CUSUM does not flag any attacks.

P101

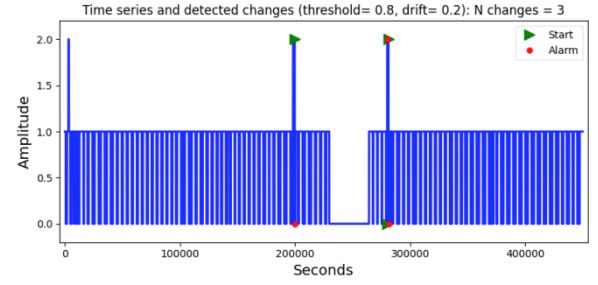


P102

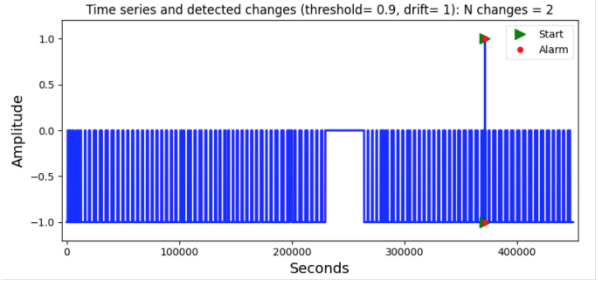


Based on the description of the attacks, we know that P102 is described as a backup pump for P101. It can be assumed that both pumps cannot be turned on at the same time. Therefore, by combining the sum and difference of P101 and P102, the attacks become more apparent and subsequently, flagged by CUSUM.

P101 + P102



P102 - P101



CUSUM and K-means Time Match

Now that our model was able to find the system-level attacks as well as the variables that had detected changes, we decided to cross-reference these two values to confirm which variables were attacked at specific times of the water treatment test. From the output of the K-means, we have a csv file of all of the datapoints labeled as either “False” for Normal system or “True” for outlier. Each of these datapoints equate to a second within the test. From the CUSUM, we have a list of detected change start time indices for each variable. Taking both the labeled seconds from K-means and detected change start times from CUSUM, we found the matching times between both models, which would point to specific variables in question. Once we found the matching times and variables, the model checks the variables against the actual variables that were attacked and displays the model’s accuracy in selecting the correct variables.

RESULTS AND DISCUSSION

Per the clustering optimization and tuning methodology previously discussed, viable test cases were produced that provided reasonable accuracy values at the subsystem levels. Overall, the accuracy values ranged from 75% to upwards of approximately 94%.

On further analysis of the results, a general trend is that the accuracy metrics are largely influenced by the model's ability to correctly label True Positives where the data points are not outliers and the systems are not under attack. Although the system was able to find attacks (True Negatives) in every case assessed, the clustering also produced a noticeable number of False Positives and False Negatives.

With respect to the problem statement, a high number of False Positives are not as problematic as False Negative. Although a False Positive indicates the clustering labeled an attack where there is no actual attack, this alert would allow system operators to take defensive actions although ultimately none would be needed. Alternatively, the False Negative scenario in which an attack is not indicated while the system is truly under an attack could lead to potential damages.

The test cases allowed for analysis of tradeoffs between the clusters, PCA components, and threshold values for each system. Overall, the number of clusters (k) and the threshold value(T) appeared to have the greatest impact on accuracy values. The results for both the SWaT 2015 and SWaT 2019 datasets are below.

SWaT 2019 Results								
Subsystem	Clusters (k)	PCA Comp.	Threshold (T)	TP	TN	FP	FN	Accuracy
P1	7	3	2	12857	201	153	1785	0.870766
P2	6	5	2.5	12708	245	302	1741	0.863764
P3	5	5	2	12720	279	290	1707	0.866831
P4	7	5	1	11279	58	1731	1928	0.756002
P5	5	5	3.5	12878	37	132	1949	0.86123
P6	5	5	1.1	12811	40	199	1946	0.85696

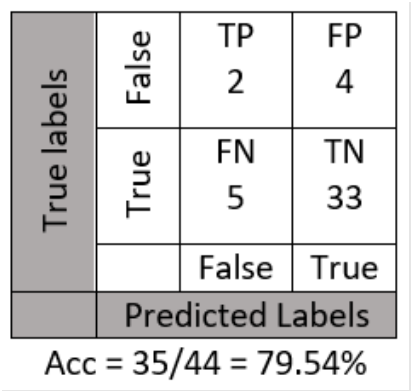
Overall, the SWaT 2015 results are satisfactory although there is room for improvement. Namely, further optimization could be warranted to find models that would lower the overall False Negative rate. Optimization for the SWaT 2015 dataset proved more difficult than the SWaT 2019 dataset, due to the much larger dataset resulting in significantly longer data processing times.

SWaT 2015 Results								
Subsystem	Clusters (k)	PCA Components	Threshold (T)	TP	TN	FP	FN	Accuracy
P1	7	5	2.5	865807	1929	26291	52692	0.91657
P2	6	5	2.5	864114	348	27984	54273	0.91311
P3	6	4	3	879284	3212	12814	51409	0.93216
P4	7	5	2.5	874699	5462	17399	49159	0.92969
P5	6	5	2.5	875292	3002	16806	51619	0.92772
P6	7	5	2.5	889665	133	2433	54488	0.93987

Similarly to the SWaT 2015 dataset, the SWaT 2019 data showed that the model correctly picked out attacks in every scenario but had numerous False Negative. Interestingly in the SWaT 2019 dataset, the model struggled with the P4 and P6 subsystems, as the attack datapoints did not display the same distance from the cluster centroids as other subsystems.

As obvious drawback and future consideration to this approach is the assumption that all subsystems must have outliers when this may not be the case as the subsystem may never be under attack. In these instances, a False Positive value would be a likely outcome and, as previously discussed, would cause system operators to undergo defensive actions that may not be needed.

For SWaT 2019, the CUSUM model used a threshold of 0.5 and drift of 0.05. After matching the system-level outlier times of K-means to the detected changes of our CUSUM, our model was able to correctly match 79.54% of the variable-level attacks with the K-means outlier detection.



For SWaT 2015, CUSUM model used a threshold of 0.3 and drift of 0.09 to correctly match 63.64% of the variable-level attacks with the K-means outlier detection.

True labels			
	False	TP 17	FP 3
	True	FN 13	TN 11
		False	True
	Predicted Labels		

Acc = 28/44 = 63.64%

For the CUSUM anomaly detection we found that while it was important for our model to correctly identify the attack variables, it was just as important, if not more important, that the model did not identify false negatives. We were worried that the CUSUM and K-means would not line up in the same times as each other. We had initially created a time buffer zone around the CUSUM times, to account for the change detection to be a few seconds off from the actual attack time. Although this had identified more correct attack variables, it had also flagged other variables that remained under normal conditions (so False Positives increased). Eventually we tuned the CUSUM parameters per dataset to both minimize the number of False Positives, while maximizing the True Positives and accuracy score.

CONCLUSION

In this project, the team has proposed a model for system wide anomaly detection based on K-means clustering and single variable anomaly detection using CUSUM. The team investigated dimensionality reduction using PCA prior to executing K-means and evaluated different number of clusters to achieve optimal accuracy scores. K-means served as the first phase of anomaly detection on a system wide scale. CUSUM is the second phase of anomaly detection by which the individual parameters are evaluated for anomalies. By cross referencing the system wide anomalies with single variable anomaly detection, the intent was to identify exactly which variables in the system the attack occurred.

The proposed model is evaluated on data collected from water distribution systems. Three data sets were derived from test beds representing small scale water treatment and distribution networks. The results suggest that K-means can achieve accuracy but with limited sensitivity. CUSUM results suggest this model is capable to detect anomalies within a single variable. However, CUSUM does not provide the duration of an attack but rather the onset or start of an attack. The combination of the two models to identify where the source of attack occurs has been demonstrated to a suitable degree.