**1b** SCADA is a system/network of software and hardware components in which the data from sensors and other industrial components like pumps and valves can be monitored and controlled from a centralized computer. These kind of systems are vunerable to cyber-physical attacks.

The BATADAL dataset is a representation of (measured) data over time from tanks (7), pumps (11), valves (4, of which 1 actionable), and other hydraulic components like joints (12 in total) from the C-Town water distribution system. The information that is embedded in the dataset contains the water level in each tank, the flow through each pump and valve and their status at that time (0 for off, 1 for on), and the pressure in the joints. Each record additionally stored the date and hour, and the attack flag status (0 for no attack, 1 for attack, and -999 as unknown(?)).

The signals that can be extracted from the dataset thus are the levels in the water tank, the flow through the pumps and valves, the status of each pump/valve over time, and the pressure in the joints. In Figure 1 all four signals can be seen over about a month in time for illustration purpose only. We can clearly see that the patterns for all signals are between boundaries, and look like to be repeating/cyclic in some form. We also see that the status indeed switches from off and on, although the value is only measured once every hour.

To see if the signals are correlated we can determine the correlation between any columns in the dataset, and create a heatmap to easily visualize the correlation (see Figure 2). We see that most signals have a no correlation (value +-0), while there exist some perfect correlated signals (excluding the signals with theirselves) and perfect negatively correlated signals (signals move in opposite direction). For example the correlation of the flow of pump 2 (F\_PU2) with the status of pump 2 (S\_PU2) is 0.998614, meaning they are perfectly correlated. This is ofcourse not strange since if the pump is enabled (status code 1) the flow in the pump is also enabled, see Figure 3 for pattern comparison of a part of the signals. If we look at other signals (that are not from the same component), we see that for instance the flow in pump 1 is almost perfectly correlated with the pressure in joint 280, with a correlation value of 0.907052. This probably indicates that joint 280 is the connection leaving pump 2, while it is connected to something that is not flowing at the same speed as pump 2. We also see (Figure 4) that for instance the signals from F\_PU1 and F\_PU2 are perfectly negatively correlated with a value of -0.949012, meaning that when the signal from one of the two is going up or down, the other one is moving in the opposite direction.

From the signals we can easily see that they look to be cyclic. From the flow in pump 2 signals in Figure 3 we see that in 24 hours (although it is measured ones per hour) the frequency of the signal repeating is about 14 times. The cycle is not a perfect sinusoidal, but the similarity between them is very well visible.

For the prediction of the signals we use an AutoRegressive model that is trained by a training set, and for the prediction it takes previous observation(s) of the time series into account. The lag has been chosen to be closer to zero and not bigger since the more recent previous observations should be more important than extra previous observations (adding more lag decreases the influence of all other observations). In Figure 5 the original and predicted signals of the water tanks can be seen, including their RMSE. We can see that for all signals did well, although for signals L\_T4, L\_T5 and L\_T7 the prediction is less accurate than for the others. If we compare signals L\_T1 and L\_T6, at first sight L\_T1 looks like to be predicted better by a lot. If we look at the RMSE the difference is just a fraction.

If instead of the water levels of the tanks we try to predict the signals of the flows of the 11 pumps, the result is very different. The average RMSE is increased to 7.59, and the predicted signal does not follow the curves of the original signal at all for most signals.

Predicting the next value in a series thus differs if the signal is applicable for it. From what we've seen is that the water levels in the tanks are very fitting for prediction, while the flow in the pumps are not.

**2b** To test the influence of the number of neighbors used in the algorithm on the detection of anomalies,

the number of neighbors is plotted as a function of the number of anomalies detected. This is done for each type of signal(L\_T, F, P). The amount of neighbors used, is chosen from 1 till 25. The upper bound is chosen because in the documentation of the SK-Learn package it is stated that "20 neighbors appears to work well in general". To see if the performance could be increased when using a larger value than 20, the upper bound is set to 25. Another variable that has an impact on the performance of LOF is the threshold used to decide whether a data point is an anomaly. Several threshold values are tested and -1.5 was chosen as the final threshold value.

From the plots above it can be seen that indeed the upper bound of the number of neighbors used could be set to 20 since the number of anomalies detected does not change much after this. An even smaller number of neighbors, approximately 16, could be used for the F and P signal types because they already converged at that point.

Remarkable are the F\_P06 and the F\_PU11 signals because they follow a different pattern as can be seen in the second plot. If we look at the data points of these two signals in figure 4 and 5, the behavior could relatively easily be explained. As can be seen in the bottom boxplot, the it consists of four points with a value close to each other and all other points have a value close to zero. In the figure 2 it can be seen that when 3 or less neighbors are used, these points are not seen as outliers because they form a cluster by themself. However, when a higher amount of neighbors is used these are classified as outliers because it uses the distance of a data point with value zero. This results in detecting the four points as outliers.

Approximately, the same reasoning could be used for the behavior of the L\_PU6 line. Although, the number of neighbors to reach the detection of the outliers is 16.

Concluded could be that a 20 is a suitable value for the amount of neighbors. Since all signals are converged to a stable solution. This value is of course depends on the threshold value which decides whether a point is an anomaly.

Anomalies in signals could occur when, for example, hardware is not stable. It is very important to know the context around the type of signal. Take F\_PU11, as an example, 0.04%(4/8761) of the measured values differ a lot compared to the other 99.96%. Without knowing the context of the type of signal these values would be classified as anomalies. However, knowing that this is the flow through the pump, it could easily be reasoned that the pump is only used very few times.

LOF measures the local deviation of a given point with respect to its neighbours, therefore LOF is a suitable method to detect anomalies in signals, since signals most of the time repeat after some time. However, it is very important to find the optimal value for k. Fortunately, there are many techniques to find the optimal value for k.

**3b** Before we can apply PCA we must normalize the training dataset 1. Since PCA is affected by scale the features must be scaled before it can be applied to it. To obtain optimal performance in machine learning algorithms it requires the features of the dataset to be unit scale, with zero mean and variance one. We remove features that contains NaN values. The remaining 36 features can be used for PCA decomposition. For PCA to work with anomaly detection it must lower the number of components because otherwise it has a difficult time differentiating anomalies from normal cases. The cumulative variance of the principal components (n) determines the number of components for a defined threshold. To be able to detect the normal behaviour of the series, enough variance must still be present. However, too much variance can mean that the model has learned about the anomalies as well.

In Figure 3.1 the PCA residuals of all signals with the corresponding number of components can be seen. It looks like that 99% removes too much anomalies, while for lower than 97% it is hard to distinquish what is an anomaly and what not. The total error increases a lot if the variance is increased. We therefore choose to use a cumulative variance of 97%, containing only 13 components. In Figure 3.2 the number of components is displayed against the cumulative variance. The last 21 components add a lot of error to the residuals, and are considered to be of less use.

The next step is to remove anomalies from the dataset. To remove the outliers we have to set a threshold. We assume a normal distribution. The threshold for normal behaviour samples is estimated to be mean plus/minus three times the standard deviation [1] for each signal separately in the original dataset (after processing). For this interval there are 85 anomalies detected and removed. Some large anomalies are present in the dataset, see Figure 3.4. All signals now behave in somewhat the same boundaries. The model should be trained after these anomalies are removed as much as possible, otherwise the model also learnes information about the anomalies thinking it is normal data. In some situations it may occur that the values of the sensors in the SCADA system do not follow the correct patterns due to for instance an operator that is evaluating a problem, a broken sensor, or a software bug.

The performance of the PCA-based anomaly detection can be evaluated by applying training set 2 to the trained model. We search for a suited threshold on the residual error, in which most attacks can be recognized by just looking at the residuals. The threshold is found by optmizing the precision (maximizing TP and minimizing FP). Training set 2 contains attack labels, such that we can easily evaluate the performance. In Figure 3.5 the residual plot of training set with the calculated threshold is visualized, in combination with the attack labels. It has an accuracy of 91% and a precision of about 96%. There are only 5 false positives, compared to the 123 true positives, which shows good performance for the anomaly detection.

We can also look at the anomaly detections in detail in Figure 3.6, in which a couple of situations are visualised. The first plot indicates a point anomaly and the second one a collective anomaly. There is no contextual anomaly detected.

[1] VARUN CHANDOLA, ARINDAM BANERJEE, and VIPIN KUMAR. Anomaly Detection: A Survey

**4c** Before we can apply ARMA to the signals of the sensors we have to pre process them. We remove the columns that are not interesting for finding anomalies: binary values, mean zero, variance zero. We could also determine whether the signals of the sensor we will use are stationary, but since ARIMA (instead of ARMA) can deal with stationary, we will not research this. We first have to figure out the order of the ARMA models, by applying the Akaike's Information Criterion in a grid search to figure out which parameters suit the model best given the training data. We can also look at the autocorrelation and partial autocorrelation plots to verify the numbers.

The only sensors in the SCADA system are the ones measuring the water levels in the tanks (tank 1 to 7). For each of the 7 sensors we are going to train the model, using the computed parameters p and q, based on the training data 1, and predict the data based on the test data. The test data contains attack flags, which we can use to evaluate the performance. The residual error is computed by taking the absolute values of the difference between the values of the test set and the predicted values. The anomalies are then the values that are bigger than a defined threshold. The threshold is estimated to be equal to the mean plus three times the standard deviation of the residual error, which in total includes 99.7% of the points to be 'normal'. We plotted the figures containing the predicted anomalies, residual error, threshold and true attacks in one figure for each sensor to show the result. The metrics are also given for each sensor. We see that the overall accuracy is quite well, although the precision and recall aren't that well. The anomalies that are detected are point anomalies and collective anomalies. These types are suitable for the threshold-based detection. If we look at the results of the anomalie detection of each sensor, we can clearly see that most are not suitable to be modeled using ARMA. There are two sensors that look like to perform relatively fine compared to the others: L\_T2 and L\_T3. In most cases there are more FalsePositives than TruePositives, and a lot of FalseNegatives and TrueNegatives.

**5b** In the first figure the result of the discretization of the L\_T1 signal can be seen. The L\_T1 signal is chosen because the values in this signal differ much and therefore it is a challenge to detect anomalies. If, for example, a binary signal was chosen it would be very easy to detect anomalies.

Each value in the signal is categorized into one of the five bins. The ranges of the bins are defined by looking at the difference between the minimum and maximum value of the signal, the difference is then divided by five to have bins with an equal range window. I have chosen for five bins to simplify the pattern of the signal as much as possible.

From the figure we can conclude that the data is evenly distributed among the different bins. Remarkable, is that the test set have almost none data points categorized in the last category.

After the data has been discretized, N-grams are applied to several sliding windows. An matrix is created where the rows represent the different used sizes for the sliding windows and the columns represent the n-grams. Then each cell is filled with the normalized frequency of the n-grams in the different sliding windows. The sliding windows used are 30, 50, 100, 200, 400. These values are chosen to check what kind of influence the length of the sliding window has on the performance. A value of 10 is chosen for n. If n is too small it would be hard to see the relation between the data points. The step size is set to 5, this is mainly done to increase the runtime of the algorithm. This value is used to shift the sliding windows in time. When a step size of 1 is used it would take very long to create the n-gram matrix. The last parameter, L, is set to 50. This values takes the top 50 most frequent n-grams found in the training data. The n-grams in the test data are compared to these top L to classify the n-grams of the test data.

The above defined hyperparameters are chosen based on intuition and by looking at the data/figures. To increase the performance of the algorithm, hyperparameter tuning need to be done. However, because of limited time I was not able to do this thorough.

The results of 5 different signals are printed. For each signal, the top 25 n-grams that have the highest cosine distance compared to the top L n-grams of the training set are shown.

Remarkable, it can be seen that the distances differ a lot and also highly depend on the type of signal.

In the last plots, the anomalies detected for signal L\_T1 and L\_T2 are plotted for a specified range. This is done by translating the n-grams back to the original data points. So, the n-gram is looked up in the discretized signal. When the index is found we need to get the point which marks this n-gram as an anomaly. This is done by adding (n-1) to the found index.

In the plots it can be seen that the anomalies are most of the time detected when the signal was expected to go up or down but does the opposite. From the figures we can conclude that the anomalies detected for L\_T are more explainable than the anomalies detected for the P\_J signal. Look, for example, at the plot of signal P\_J302, it is hard to explain why these points are classified as anomalies. However, this could also be the reason of the chosen hyperparameters. So, as mentioned earlier, to increase the performance of the algorithm, the chosen hyperparameters need to be optimized per signal.

**6a** The comparison task involves comparing the four methods used in the previous tasks. However, the implementations of the two authors deviate too much to be able to make a good comparison. Therefore, we decided to compare the the PCA and ARMA methods, and the LOF and NGRAM methods against each other. This way we can still make a comparison.

For the comparison of PCA and ARMA two measures are used: the number of distinct anomalous regions in which at least one anomaly is decected, and the basic performance metrics (accuracy, precision, recall, and confusion matrix). The anomalous regions are an important measure since it explains how many different anomalous attacks are detected. If we just count the number of detected anomalies, they can for instance all be for the same attack and therefore not a very effective anomaly detection mechanism. The higher the number of regions, the more effective the method. The second measure that is used are the standard machine learning metrics. We are interested in a high precision in combination with a high number of true positives (also depends on number of 'true' cases there are) and a high accuracy. This means there are just a few false positives and a good amount of true positives in the prediction. False positives in this context has the meaning of a false alarm, and are very much unwanted for the SCADA system. Upon receipt of an alarm, the floor managers will stop the system from running until they diagnosed and fixed the issue. If this happens often, a lot of time and money is lost and the possibility of neglecting the alarms will increase.

The measures are exposed to both the PCA and ARMA methods. One important note is that the PCA method used the training dataset 2 for the prediction and ARMA used the test dataset. This is of course not a direct comparison, but due to time constraints it wasn't possible to convert one of the methods with the new data. The datasets also differ in size, thus we increased the confusion matrix results to be equal in the total number of samples for a more equal evaluation, although this is of course not equal.

If we now look at the numbers we can see that the number of regions is a lot higher for PCA than for ARMA. The precision and accuracy, as well as the number of TP, is also quite a lot higher for PCA. The number of FP is equal for both, while the number of false negatives (an alarm that never gone off) is almost double for ARMA than for PCA. We can conclude that based on these numbers the PCA method is clearly the better of the two.

To compare the performance of the LOF and N-grams methods, the top-k detected anomalies are compared with each other. This measure is chosen because both methods already created a ordered list of detected anomalies. Therefore it was relatively easy to check the similarities of the results.

The results of the algorithms is compared for three different signals: L\_T1, L\_T2, P\_J302. As can be seen above, the methods find totally different anomalies. There are multiple reasons why this could happen. One reason could be the hyperparameters used in both methods. Another reason could be a fault in the implementation of one the algorithms. Also it could be that the algorithms find different anomalies, however, such a huge difference is not reasonable.

Unfortunately the performance of the LOF algorithm could only be compared to the performance of the N-grams method because the assignment was interpreted in a different way between the team members.