Descriptive and Predictive Modeling of Water Distribution Network Dynamics using Multivariate Time Series Data

Susana Gomes
susana.correia.gomes@tecnico.ulisboa.pt
Instituto Superior Técnico
Lisbon, Portugal

ABSTRACT

Water distribution networks (WDNs) are hydraulic infrastructures that provide a continuous supply of pressurized safe drinking water to all consumers, playing an important role in public health. Leakages cause service interruptions, waste resources, and compromise water quality. Although we can find many methods that support the monitoring and control of WDNs, they exhibit limited ability to detect anomalies and are not yet consistently applied to Portuguese WDNs. We show that it is possible to (1) describe the dynamics of a WDN through spatiotemporal correlation analysis of pressure and volumetric flowrate sensors, and (2) analyze disruptions on the expected correlation to detect burst leakage dynamics using standard classifiers. Our approach is promising in a synthetic setting and offers initial support towards leakage detection in real WDNs despite the presence of highly irregular consumption patterns, a limited number of recorded leakages, and highly heterogeneous leakage profiles. We discovered that the disruption caused by leakages is higher shortly after the burst. Furthermore, a comprehensive pairing of heterogeneous sensors and data balancing in the real setting is also promising. Our results suggest that it is important to access data from WDNs with good sensor coverage and complete information about leakages. Accordingly, we believe that our WDN would benefit a lot from sensor expansion and relocation. Lastly, given the simplicity, novelty, and accuracy of the proposed correlation-based principles for anomaly detection in heterogeneous and georeferenced time series, we anticipate our work to contribute to the study and development of automated leakage detection in portuguese WDNs. **Keywords:** burst leakage detection, correlation analysis, multivariate time series, prediction, spatiotemporal data analysis, water distribution network.

1 INTRODUCTION

Water distribution networks (WDNs) are hydraulic infrastructures responsible for providing a pressurized and continuous supply of safe drinking water to all consumers [8]. WDNs have an important role in public health, and with climate change and population growth demanding higher consumption levels of water, the need for well-managed systems is more important than ever [15]. Moreover, leakages can cause service interruptions, contribute to resource wastes, and increase the risk of compromising water quality [6]. Although we can find many methods and algorithms that improve the monitoring and control of WDNs, such as WaterWiSe [23] in Singapore, the reality is that they show limited ability to detect anomalies and are not yet consistently applied to Portuguese WDNs.

In this work, we study the spatiotemporal correlations of pressure and volumetric flowrate sensors under normal conditions to detect anomalous behavior when the normal conditions suffer a disruption. Therefore, and as part of the WISDOM [4] project, our main focus is to detect burst leakages according to Portuguese necessities by identifying descriptors and predictors of WDN dynamics.

Due to the nature of the problem, we identified several challenges that may hamper the performance of the description and prediction tasks, namely, (1) poor sensor coverage, (2) highly irregular consumption patterns, (3) leakages with multiple profiles, (4) low number of leakages and lack of information regarding its size and exact beginning, (5) network changes and interventions that disrupt the natural behavior of the network, and (6) the necessity of detecting leakages as soon as possible. We also had to explore and analyze high volumes of data that contained gaps and irregular time steps of recording. The data comprised numerous time series from volumetric flowrate and pressure sensors placed throughout the WDN of a Portuguese tourist resort with extensive irrigation, large hotel units, and irregularities in the occupation of households.

Through the development of a new technique that uses the disruption of spatiotemporal correlations between sensors to detect burst leakages in WDNs, we provide (1) a comprehensive analysis of network dynamics in a real WDN versus in a synthetic WDN, (2) an assessment of two correlation methods as descriptors of network dynamics, (3) a study of how the correlation between sensors evolves over time, (4) a performance assessment of three standard classifiers on leakage detection, (5) a study of the importance of feature/sensor selection in leakage detection, and (6) a study of the impact of the leakage size in its detection. Additionally, these contributions could offer direct support to answer alternative problems in WDNs, such as, (1) optimal sensor placement through the installation of new sensors where current sensor correlations are weak, and (2) the detection of status' changes in active hydraulic elements, since these might also cause a correlation disruption.

2 BACKGROUND

This section introduces the fundamental topics of this work, which we separated into two parts. The first one involves concepts associated with WDNs, while the other comprises of technical concepts, more specifically, the ones related to time series data.

2.1 Water Distribution Network Background

Water distribution networks (WDNs) are hydraulic infrastructures that provide a continuous supply of pressurized safe drinking water to all consumers [8]. They are generally composed of a large number of active hydraulic elements, such as pumps and valves, and passive ones, such as pipes and reservoirs. Water utilities (WUs) are entities responsible for managing WDNs, relying on the existence of sensors and monitoring devices placed throughout the network.

Part of the data used in this work is obtained from sensors responsible for measuring water pressure and volumetric flowrate. The pressure is described as the normal force per unit area at a given point acting on a given plane within the water mass of interest. Regarding the volumetric flowrate, simply referred to as flowrate in this work, corresponds to the amount of water flowing in a certain point per unit of time [24]. Note that the flowrate is proportional to the square root of the pressure difference between two points.

Additional data was generated using an artificial EPANET model of the target WDN. EPANET is a public domain software created by the United States Environmental Protection Agency that simulates hydraulic and water quality behavior within pressurized pipe networks. It also provides an environment for editing network input data, running simulations, and visualize the results.

Lastly, our data (real and synthetic) contains leakages, which are commonly defined as the loss of treated water from the network through uncontrolled means [10, 19]. They are categorized in literature as (1) background leakages, which consist of the aggregation of leakages small enough to be undetected for long periods, and (2) burst leakages, which are occurring pipe ruptures that usually result in a large water discharge [9].

2.2 Technical Background

Since time series data is our main source of information, we define them and explore the challenges of its classification, as well as existent solutions.

2.2.1 Time Series Definition. A discrete time series is described as a sequence of T observations, each one being measured at a discrete time $t \in \{1, \ldots, T\}$, made sequentially and regularly along T instances of time. A real-valued observation of a time series, $\mathbf{x}_t = (x_{1t}, \ldots, x_{kt})$, where k is the multivariate order, $t \in \{1, \ldots, T\}$, and $x_{kt} \in \mathbb{R}$ [3]. A time series is called univariate when $k = 1$, and multivariate when $k > 1$.

2.2.2 Time Series Classification. Classification is the process of predicting a class label for a given unlabeled instance [25]. The prediction depends on the classifier, which uses training data to learn and create rules to classify new instances. Classical classifiers, such as naive Bayes (NB) and support vector machines (SVMs), assume that features are not dependent on the ordering [2]. Consequently, these classifiers will handle time series' time points as separate features. Several algorithms consider time series as a whole, i.e., without ignoring feature ordering. For example, the one nearest neighbor classifier with an Euclidean distance or with a dynamic time warping is usually the starting point for most researchers. According to Bagnall et al. [2], time series classifiers can be grouped into six categories, namely, time domain distance based, differential distance based, dictionary based, shapelet based, interval based, and ensemble classifiers. Note that interval based classifiers extract features, such as the mean and standard deviation, from time series' intervals.

2.2.3 Classical Classification. Extracting features from time series' intervals allows the creation of a new dataset without explicit feature ordering, making classical classifiers a viable solution. Note that this work does not present a comparative study of different classifiers. Since we only want to assess the potential of the extraction of features from time series in the context of WDNs, we only focused on two well-known classifiers, namely, NB and SVMs.

Naive Bayes. NB classifier uses a probabilistic classification approach that simplifies learning by assuming all features are conditionally independent given the class [25]. We assume a training dataset with m points $\mathbf{x}_i \in R^d$ in a d-dimensional space, where $i \in \{1, \ldots, m\}$, y_i is the class for each point, with $y_i \in \{c_1, c_2, \cdots, c_k\}$, and $\mathbf{D}_i$ representing the subset of points that are labeled as class c_i. Accordingly, NB uses the Bayes theorem to predict the class of $\mathbf{x}$ as the one that maximizes the posterior probability, i.e., $\hat{y} = \arg\max_i \{P(c_i \mid \mathbf{x})\}$. The posterior probability of class c_i, $P(c_i \mid \mathbf{x}) = P(\mathbf{x} \mid c_i)\, P(c_i)$, is the product between the likelihood $P(\mathbf{x} \mid c_i)$ and the prior probability $P(c_i)$ of class c_i. Essentially, the posterior probability of class c_i depends on the likelihood of that class taking its prior probability into account. The prior probability of class c_i is defined as $P(c_i) = \frac{m_i}{m}$, while the likelihood is defined as the probability of observing $\mathbf{x}$ assuming that the true class is c_i. Due to the initial assumption that all features are independent given the class, we can decompose the likelihood of class c_i into a product of the likelihood along each dimension X_j,

$$P(\mathbf{x} \mid c_i) = P(x_1, x_2, \cdots, x_d \mid c_i) = \prod_{j=1}^{d} P\left(x_j \mid c_i\right). \tag{1}$$

When a numeric feature is shown to be approximately normally distributed for a given class c_i, the likelihood for class c_i, for variable X_j, is given as

$$P\left(x_j \mid c_i\right) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left\{-\frac{\left(x_j - \mu_{ij}\right)^2}{2\sigma_{ij}^2}\right\}, \tag{2}$$

with μ_{ij} and σ_{ij}^2 as the mean and variance for variable X_j, for class c_i. Although feature independence is usually violated in real datasets, Rish et al. [21] shows that NB is a very competitive classifier and works best when the data either has completely independent features or has functionally dependent features.

Support Vector Machines. SVMs are a classification method that sees observations in a d-dimensional space and finds a hyperplane that maximizes the gap or margin between the observations from different classes [25]. In a two dimensional space, this optimal hyperplane corresponds to a line that separates the classes. When it is not possible to find a hyperplane in the current dimension that separates the classes completely, we can use the kernel trick to find an optimal hyperplane in a high-dimensional space.

Assuming a dataset $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ with m points in a d-dimensional space and with two class labels $y_i \in \{+1, -1\}$. An hyperplane serves as a linear discriminant, i.e., a decision boundary that predicts the class y for any point $\mathbf{x}$. An hyperplane $h(\mathbf{x})$ in d dimensions is defined as $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, where $\mathbf{w}$ is a d-dimensional weight vector, and b is a scalar called bias. The hyperplane $h(\mathbf{x})$ splits the original d-dimensional space into two half-spaces. When each half-space has points from a single class, we call it a separating hyperplane [25].

Considering that the distance of a point $\mathbf{x}$ from the hyperplane $h(\mathbf{x}_i) = 0$ is given by $\delta_i = \frac{y_i\, h(\mathbf{x}_i)}{\|\mathbf{w}\|}$, support vectors are all data points that achieve the minimum distance $\delta^* = \min_{\mathbf{x}_i}\left\{\frac{y_i\ h(\mathbf{x}_i)}{\|\mathbf{w}\|}\right\}$. Therefore, support vectors are the points closer to the hyperplane and their distance from it is called margin. These points, or vectors, influence the position and orientation of the hyperplane and are chosen to maximize the margin of the classifier. Additionally, we can normalize the parameters of the hyperplane so that the absolute distance of a support vector from it is 1, i.e., $\delta^* = \frac{1}{\|\mathbf{w}\|}$. We call them canonical hyperplanes and the main idea of SVMs is to find the canonical hyperplane with the maximum margin among all possible separating canonical hyperplanes,

$$h^* = \arg\max_h \{\delta_h^*\} = \arg\max_{\mathbf{w},b}\left\{\frac{1}{\|\mathbf{w}\|}\right\}. \tag{3}$$

Since not all datasets are linearly separable, we can use the kernel trick. The idea is to map the original d-dimensional points in a high-dimensional space, called feature space, via non-linear transformations, in hopes that the data will be linearly separable there. It is also important to note that this trick allows us to find the optimal hyperplane without actually having to map the points in the new space. Instead, kernel functions find the non-linear relationships among features [25]. One popular function is the polynomial kernel, which is defined as

$$K_q(\mathbf{x}, \mathbf{y}) = \left(c + \mathbf{x}^T\mathbf{y}\right)^q, \tag{4}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, q is the degree of the polynomial and $c \geq 0$ is a constant. The linear and quadratic kernels are special cases of the polynomial kernel when $q = 1$ and $q = 2$, respectively. The Gaussian radial basis function (RBF) is another well known kernel function and is defined as

$$K(\mathbf{x}, \mathbf{y}) = \exp\left\{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right\}, \tag{5}$$

where $\sigma > 0$ is a constant. It is important to note that the RBF kernel has infinite dimensionality. According to Hofmann [13], a low polynomial kernel or an RBF kernel are good alternatives to conventional classifiers.

3 RELATED WORK

This section provides an overview of the existing literature on topics related to some of our challenges, namely, the correlation between a pair of time series, feature selection, and burst leakage detection.

3.1 Correlation Measures

Since the first part of our solution focuses on proposing descriptive models of network dynamics through the correlation analysis of pairs of sensors, it is important to explore different correlation measures. Classical and modern approaches can be considered. For example, Pearson's cross-correlation coefficient (PCC),

$$PCC(X, Y) = \frac{\sum_{t=1}^{M}(x_t - \bar{x})(y_t - \bar{y})}{\sqrt{\sum_{t=1}^{M}(x_t - \bar{x})^2}\cdot\sqrt{\sum_{t=1}^{M}(y_t - \bar{y})^2}}, \tag{6}$$

is one of the best-known correlation measures, where X and Y are two univariate time series of length M, and $\bar{x}$ and $\bar{y}$ are their corresponding means [16]. It is also important to note that the sample size has a drastic effect on the results. De Winter et al [7] showed that in their dataset, a sample size of 25 was not enough to capture the true correlations. However, when doubling that size, the results got substantially better.

Rank correlation coefficients are used to measure an ordinal association, i.e., the extent to which one variable increases when the other increases without requiring that increase to be represented by a linear relationship. Both Spearman's and Kendall's rank correlation coefficients fall into this category. Since they are less sensitive to non-normality in distributions, they can be considered as an alternative to PCC [14].

Podobnik and Stanly [20] proposed an interesting solution called detrended cross-correlation analysis (DCCA), which is based on detrended fluctuation analysis (DFA) and allows the analysis of two non-stationary time series. Considering X and Y as two univariate time series with M observations, each one being measured at a discrete time $t \in \{1, \ldots, T\}$, DCCA starts by defining $R_k \equiv \sum_{t=1}^{k} x_t$ and $R'_k \equiv \sum_{t=1}^{k} y_t$, where $k \leq T$. Then, it divides both time series into $T - n$ overlapping boxes, each containing $n + 1$ values, where $1 \leq n < T$. Considering that each box starts at t and ends at $t + n$, DCCA defines the local trend as $\tilde{R}_{k,t}$ and $\tilde{R}'_{k,t}$, where $(t \leq k \leq t+n)$. We now calculate, for each box, the covariance of its residuals,

$$f^2_{\text{DCCA}}(n, t) \equiv (n-1)^{-1}\sum_{k=t}^{t+n}\left(R_k - \tilde{R}_{k,t}\right)\left(R'_k - \tilde{R}'_{k,t}\right). \tag{7}$$

Finally, to obtain the detrended covariance F^2_{DCCA}, we average the results of all boxes,

$$F^2_{\text{DCCA}}(n) \equiv (T-n)^{-1}\sum_{t=1}^{T-n} f^2_{\text{DCCA}}(n, t). \tag{8}$$

Since real-data can be categorized by a high degree of non-stationary, we can apply DCCA to a variety of fields.

3.2 Feature Selection

Feature selection methods are responsible for finding relevant and informative features that represent our data. Additionally, it can increase an algorithm's speed and improve its performance [5]. In our case, it could help reducing the number of sensors needed in the WDN. One approach is through individual relevance ranking methods, which assume feature independence [11]. Filter functions are the ones responsible for ordering the features according to their degree of dependence on the target [5]. Classical statistic tests, such as the Chi-squared, the Analysis of Variance (ANOVA) F-test, and the Kruskal–Wallis H test can be used as filter functions [11].

The Chi-squared test is used when in the presence of discrete features and measures the difference between the observed counts and expected counts of the data [25]. For continuous features that follow a normal distribution, ANOVA F-test can be used instead. It computes the ratio between the mean square error between groups and the mean square error within groups [12].

For non-normal distributions, we can use the non-parametric Kruskal–Wallis H test [18]. This test starts by assigning ranks to the data using all scores in the experiment. Then, it computes the sum of squares between groups,

$$SS_{\text{bn}} = \frac{(\Sigma R_1)^2}{n_1} + \frac{(\Sigma R_2)^2}{n_2} + \cdots + \frac{(\Sigma R_k)^2}{n_k}, \tag{9}$$

where K is the number of groups, n_i denotes the number of instances in the i^{th} group, and ΣR is the sum of the ranks in each group. Lastly, it computes the value of H as

$$H = \left(\frac{12}{N(N+1)}\right)(SS_{\text{bn}}) - 3(N+1), \qquad (10)$$

where N is the number of instances in the data [12]. If H exceeds a critical value, we may conclude that the groups do not come from the same population [18].

3.3 Burst Leakage Detection

Since the main focus of this work is burst leakage detection, we explored different methods that detect and discover them. These methods can be grouped into hardware-based methods, which aim to accurately locate the leakage, and software-based methods, which attempt to isolate possible leakage areas [17]. Software-based leakage detection methods, usually rely on a model or algorithm that uses additional information, such as pressure or flowrate data, rather than leakage noise information, and can be grouped into numerical and non-numerical methods [17]. The numerical ones mainly include methods based on transient events and on the traditional hydraulic model method, whereas the non-numerical ones use a data-driven approach.

The non-numerical methods, in which our work is included, analyze the monitoring data, and find their rules through data mining and artificial intelligence algorithms. It is important to note that some methods, for example, methods that use artificial neural networks, need to be updated frequently and require a lot of historical data to be trained [1]. Valizadeh et al. [22] proposed a solution using feature extraction and classification. They used time windows and time-domain features to transform the time signals of flow, pressure, and temperature at the inlet and outlet of the network into a matrix of features. The time-domain features include the mean value, root-mean-square, standard deviation, among others. Regarding the classifiers, they compare a k-Nearest Neighbors classifier to a Bayesian classifier. Although it is not clear which one is better, these two classifiers perform differently depending on the window size and leak characteristics, i.e., its diameter and location.

4 SOLUTION

The goal of this work is to detect burst leakages in WDNs through the classification of time series data collected by sensors. Accordingly, our solution is divided into the three steps, namely, preprocessing, correlation analysis, and prediction.

4.1 Dataset Description and Preprocessing

Our WDN is responsible for supplying water to a tourist resort with extensive irrigation, large hotel units, and irregularities in the occupation of households. Therefore, it suffers from highly irregular consumption patterns, which will allow us to test our solution in a challenging setting. We will also conduct our experiments in an equivalent synthetic setting without noise and network changes.

Regarding the synthetic setting, we have 18696 chunks of data from 7 volumetric flowrate and 21 pressure sensors of an EPANET model of the original WDN. The chunks have 145 time points and a granularity of 600 seconds. Each one includes a 24 time point leakage with a different combination of location (3116 network points) and size (six coefficients between 0.05 and 2.0). The higher the coefficient, the larger the leakage is. Due to faulty configurations of pipe directions in the model, we considered the absolute value of negative values. We also approximated to zero all flowrate values bellow 1×10^{-3} since there was no evidence of water flow.

Concerning the real setting, the data corresponds to the entire year of 2017, and was extracted from 7 volumetric flowrate and 6 pressure sensors. Since the monitoring system has a granularity of approximately 60 seconds, to estimate observations at a regular sampling (equally distant points), we applied a linear interpolation method available in Pandas[1] Python module to guarantee regular time steps of recording. The data is also missing the last day of every month, but since it is only 12 days, we did not apply any data imputation technique. Regarding leakages, we have complete information about 12 of them but we do not know their exact beginning nor its size.

4.2 Descriptors of Dynamics

Here we explain how we transformed our original time series dataset into a new dataset without explicit feature ordering and how we selected the most informative features. It is also important to remember that the disruption of the correlation between two sensors may indicate a leakage.

4.2.1 Feature Construction. Since the correlation value between two highly correlated sensors is expected to decrease when a leakage occurs between them, we created a feature construction strategy that calculates the correlation value between all pairs of sensors throughout time. Accordingly, each column of our new dataset corresponds to a pair of sensors and each row indicates the time window in which we calculated the correlation. For predictive purposes, we also have a column that indicates if an instance is positive or negative, i.e., if it contains a leakage or not.

Regarding the correlation methods, we used two correlation methods described in section 3.1. The first one is the classical and widely used Pearson cross-correlation coefficient (PCC), available in SciPy[2] Python module. Since our time series suffer from irregularities, we also decided to use the modern approach cross-correlation analysis (DCCA), available in Jaime Ide's profile on GitHub[3].

Concerning the synthetic dataset, each chunk originated two instances, specifically, one positive and one negative. To create the positive instances, either the beginning or the end of the leakage is in the middle of the time window. Regarding its size, we generated different datasets with windows between 16 and 40 time points.

For the real dataset, we have a sliding window that moves over the time series in intervals of 15 minutes, allowing the early detection of leakages. We considered as positive all instances that occurred between the leakage report and the beginning of the resolution work. Since we do not know the exact time the leakage started, we also counted as positive the instances that took place two hours before the official leakage report time. Note that we disgarded the instances that occurred during the leakage resolution.

[1]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.interpolate.html
[2]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html
[3]https://gist.github.com/jaimeide/a9cba18192ee904307298bd110c28b14

Finally, we generated different datasets with time windows between 60 and 240 minutes.

4.2.2 Feature Selection. Since sensors are not all correlated, reducing the number of features might lead to fewer sensors needed in the network and improve the predictors' speed and performance. Since our data does not follow a normal distribution, we used the non-parametric Kruskal– Wallis H test described in section 3.2 and available in SciPy[4] Python module. Additionally, we also created our own feature ranking approach that favors strong correlations in normal situations, i.e., without leakages.

In our approach, we (1) find the mean of each pair (feature) within negative instances. Then, (2) for each sensor, we sort its pairs by mean. Lastly, (3) we take the pairs that ranked first for each sensor, remove the duplicates, and sort them. We do that for the ones that ranked second and so on. Thus, this strategy prefers the highest correlated pairs without favoring one sensor in particular.

4.3 Predictors of Leakage Dynamics

To predict burst leakages, we used the two classifiers described in section 2.2.3 and available in the Scikit-Learn Python module, namely, NB[5], an SVM[6] with a linear kernel, and an SVM[7] with an RBF kernel. To assess their performance and tune their hyperparameters, we collected class-specific measures, namely, $recall_N$, $precision_P$, $recall_P$, and F_P (F_1 measure). Since binary classifiers usually use a threshold ρ to determine the instances' class, we used the receiver operating characteristics (ROC) plot to (1) understand what the best performance they can achieve is and to (2) assess if the threshold is preventing them from attaining desirable results.

Regarding the evaluation strategy, for the synthetic dataset, we used 20% as the test set and a 5-fold cross-validation on the remaining 80%. Concerning the real data, due to the severe class imbalance, the number of folds used is equal to the number of leakages, ensuring one leakage in the validation set, while the others are used for training. Please note that one leakage is equivalent to more than one positive instance due to the sliding time windows. We also put one of the folds aside to serve as the test set.

5 RESULTS: DESCRIPTIVE SETTING

This section presents a comprehensive analysis of the steps performed to learn the descriptive models of network dynamics.

5.1 Exploratory Data Analysis

As previously mentioned, the synthetic setting has a total of 28 sensors placed throughout the WDN that provided time series with a clear seasonality. In these conditions, disruptions in the network are more noticeable, which is the ideal setting for the first experiments. Moreover, when the flowrate drops, the pressure increases, and vice-versa, which is a sign of inverse correlation.

Unlike the synthetic setting, all 13 sensors from Infraquinta are concentrated in three areas. Consequently, the lack of sensor coverage may negatively impact the ability to detect leakages that do not occur between these areas. The time series from Infraquinta's WDN, as exemplified in Fig. 1, are generally more complex than the synthetic ones due to the exposure to different consumption patterns. For example, we decomposed the time series using an

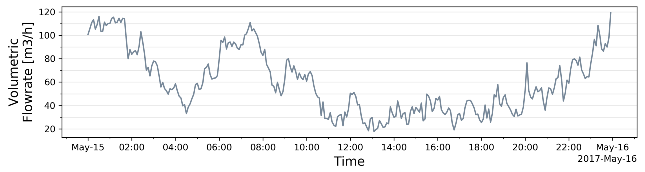

Figure 1: A real time series during May 15, 2017.

additive decomposition model and found large residuals and an uninformative trend.

Lastly, the data from two pressure sensors from the synthetic setting and four pressure sensors from the real setting were mostly zero, and thus, were not considered for the next steps.

5.2 Correlation Analysis

Next, with the help of heatmaps, we overview the datasets obtained through feature construction. Accordingly, one heatmap corresponds to one instance of the dataset, and each cell represents the correlation value (DCCA or PCC) between a pair of sensors.

Regarding the synthetic setting, since we considered 26 sensors, our new dataset has a total of ${}^{26}C_2 = 325$ features. Fig. 2 shows the heatmaps of two representative synthetic instances, one positive and one negative, from the same chunk, using DCCA as the correlation method. Observing the negative instance, sensors of the

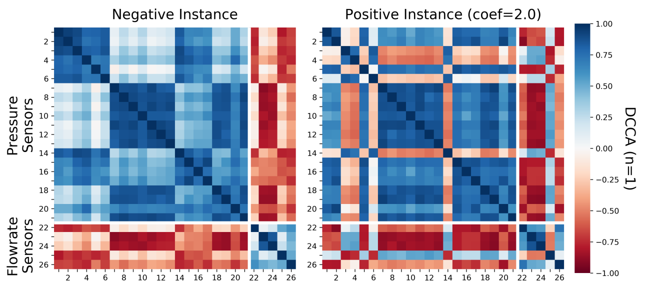

Figure 2: DDCA heatmaps of two synthetic instances.

same type are directly correlated, while pairs of different types are inversely correlated. PCC also obtained similar color patterns, but the difference between instances is less apparent than with DCCA.

Regarding the real setting, our new dataset has a total of ${}^{9}C_2 = 36$ features. Although we detected some similarities between DCCA and PCC heatmaps, it is not as apparent as in the synthetic ones. Comparing the negative and positive instances, while most correlations became weaker in PCC, some grew stronger in DCCA.

5.3 Correlation Over Time

Another critical element to assess is how correlation evolves, especially before and during the leakage. Fig. 3 shows the DCCA and PCC over time between two synthetic sensors, flowrate and pressure, from the same chunk.

[4]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.mstats.kruskalwallis.html
[5]https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
[6]https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
[7]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

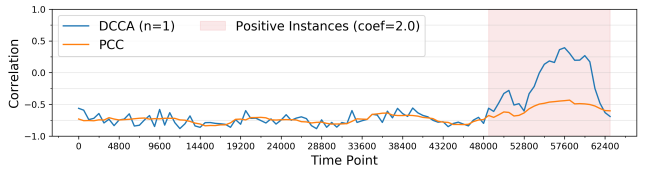

Figure 3: Correlation over time between two synthetic sensors.

Before the leakage, DCCA and PCC remained between -0.5 and -1, negative or positive. Note that the disruption peaks when the middle of the time window aligns with the beginning of the leakage. Moreover, DCCA oscillates a lot more than PCC, causing the impact of the leakage to be more apparent.

Regarding the real setting, the behavior of DCCA and PCC is very similar in this setting. Both are very unstable before the leakage, which is probably an indication that they can not accurately quantify the correlation between time series due to the large volume of noise present. Although the differences between the negative and the positive instances are not as striking as in the synthetic setting, we can still notice a subtle change after the leakage.

5.4 Correlation in Small Leakages and Different Time Windows

For the experiments with the synthetic dataset, we have been using a leakage coefficient of 2.0 and a window of 40 time points. Fig. 4 shows the DCCA variation with the leakage coefficient and time window size between the same representative sensors of figure 3.

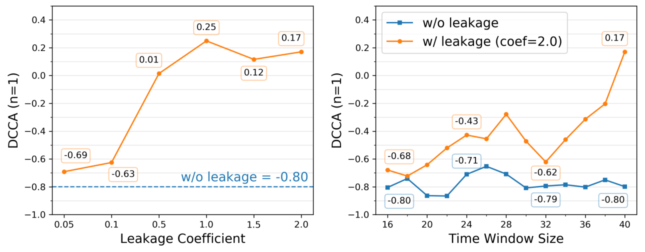

Figure 4: DCCA variation with the leakage coefficient and time window size between two synthetic sensors.

As expected, larger leakages ($coef \geq 0.5$) seem to cause a higher disruption in the correlation than the smaller ones. Although we can not perform this analysis for the real setting because we do not have information about the leakage sizes, it is expected for smaller ones ($coef \leq 0.1$) to go undetected.

Regarding the time window, the correlation values start diverging around 32 time points, peaking at 40. One possible explanation is that DCCA could not accurately identify the degree of correlation with less than 32 points. For the real setting, we have been using a time window of 120 minutes. Although the 60 minute window fluctuates a lot more than the others, all behave similarly over time. Contrarily, 240 minutes causes so much stability that it sometimes camouflages the leakage. Lastly, a time window of 120 minutes seems to be the most balanced, i.e., it does not fluctuate a lot, but it still allows us to notice leakages.

5.5 DCCA Parameterization

Since DCCA has a parameter n that affects the size of the overlapping boxes, it is important to access its impact on both settings. For the synthetic one, we analyzed n values between 1 and 39. With $n = 1$, the correlation difference between negative and positive instances reaches its peak. For $n > 1$, the difference between instances slowly grows until $n = 39$, its best value after $n = 1$.

To understand how n affects DCCA in a real setting, we chose four different values so that the boxes' sizes would be $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, and $\frac{1}{5}$ of our time window. Using a time window of 120 minutes, n corresponds to 59, 39, 29 and 23. Fig. 5 shows the DCCA variation between two representative real sensors, flowrate and pressure, during February 7, 2017. Although all four values of n seem to follow the same pattern, $n = 59$ stands out from the others since it fluctuates a lot more. The other three do not fluctuate as much but vary enough to let us notice the leakage.

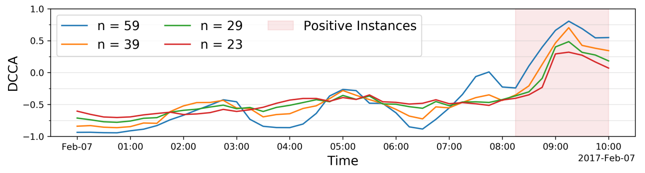

Figure 5: DCCA variation with the leakage coefficient between two real sensors.

6 RESULTS: PREDICTIVE SETTING

This section assesses the predictive ability of detecting leakages in controlled and real settings, presenting a comprehensive analysis of the hyperparameterization steps performed to identify the most suitable leakage dynamic predictors for our WDN.

6.1 Artificial Water Distribution Network

Here we focus on understanding how a classifier performs on a WDN under ideal conditions. Note that the synthetic dataset obtained through feature construction has 37392 instances, where half is negative and the others are positive (3116 per coefficient).

6.1.1 Initial Results. To get an overall idea of the classifiers' performance in our synthetic setting, we started by gathering the results of three different classifiers using DCCA and PCC. Accordingly, Table 1 presents the mean results for a 5-fold cross-validation on the training set. Note that TWS and #F refer to the time window size and the number of features used, respectively.

Table 1: Initial training set results of the synthetic setting.

Classifier	Correlation	TWS	#F	$Prec_P$	$Recall_P$	F_P
NB	DCCA ($n = 1$)	40	325	0.88	0.71	0.78
	PCC	40	325	0.93	0.64	0.76
SVM (Linear)	DCCA ($n = 1$)	40	325	1.00	1.00	1.00
	PCC	40	325	1.00	0.97	0.99
RBF (RBF)	DCCA ($n = 1$)	40	325	1.00	0.75	0.86
	PCC	40	325	0.94	0.70	0.80

Overall, NB performs worse than our two SVMs, with the linear kernel achieving the best results. As the experiments in sections 5.2

and 5.3 indicated, DCCA provided better results than PCC. Hence, the next sections will only include the results obtained using DCCA. It is also important to note that all iterations of the 5-fold cross-validation obtained similar results, meaning that the classifiers do not seem to be overfitting the data. Unless otherwise stated, the reader can always assume that for this setting.

6.1.2 Feature Selection. Here we assess if we can reduce the number of pairs used without negatively affecting the classifiers' performance. Fig. 6 shows NB and SVM (linear) results along with the top k features selected, starting from the top 15 to the top 100.

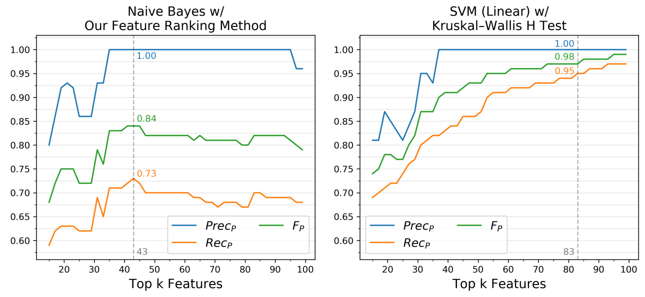

Figure 6: Classifiers' performance in the synthetic setting along with the top k features selected.

While the Kruskal–Wallis H test worked better on SVMs, ours managed to improve the performance of NB. Note that NB's performance using 43 features is better than with 325, meaning that some could be misleading and uninformative. Regarding SVM (linear), it never reaches it using the top 100 alone, but Kruskal-Wallis still managed to deliver competitive results using 83 features. Regarding the next sections, since an RBF kernel could not match the results with a linear one, we will only use the latter. Moreover, NB will use the top 43 features selected by our method, and SVM will use the top 83 by Kruskal–Wallis.

6.1.3 Time Window Size. Since we have been using a window of 40 time points, it is important to understand the impact of other sizes on the results. Fig. 7 shows that increasing the size of the time window does not always enhance the classifiers' performance. As

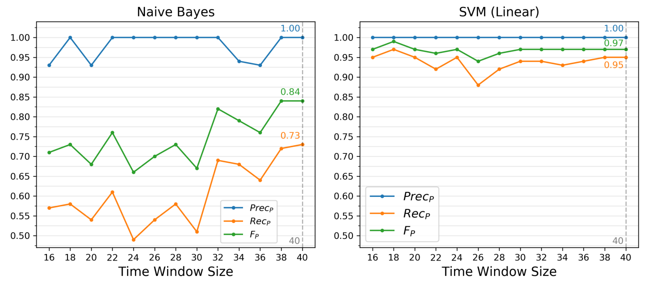

Figure 7: Classifiers' performance in the synthetic setting along with different time window sizes.

the experiments in section 5.4 indicated, we see an improvement around 30 time points. Unlike NB, SVM seems to be able to adapt well to different time windows. Since a window of 40 time points obtained the best results for both classifiers, we will continue to use it in the next sections.

6.1.4 Threshold Adjustment. Here we assess (1) if the default threshold zero ($\rho = 0$) prevents NB and SVM from achieving optimal results and (2) its impact on small leakages.

Fig. 8 shows that, with an AUC of 0.99, SVM's performance is almost optimal. Note that SVM's $recall_P$ increases when we lower

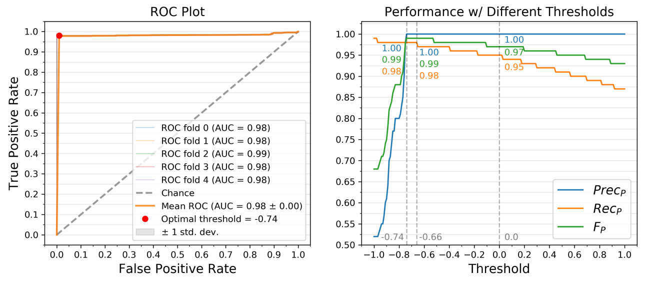

Figure 8: SVM's (linear) ROC plot and its performance in the synthetic setting along with different thresholds.

the threshold. Then, after peaking at the optimal threshold $\rho = -0.74$, the $precision_P$ rapidly drops, indicating that some positive instances' scores are close to the majority of the negative ones. We can also see that $\rho = -0.66$ achieves the same results as the optimal threshold $\rho = -0.70$. Therefore, we can obtain the same performance without lowering the threshold as much. Regarding NB, it has a AUC of 0.94 and its optimal threshold is $\rho = -0.99$ ($Prec_P = 0.94$, $Rec_P = 0.82$, $F_P = 0.88$). Since $\rho = -0.99$ increases the number of FP, a modest alternative would be to use $\rho = -0.67$ ($Prec_P = 1.00$, $Rec_P = 0.75$, $F_P = 0.86$). Although the results are not as good, it still increases the $recall_P$ and prevents instances from being classified as negative if their negative class score is low.

Fig. 9 clearly shows that the lower the coefficient of a positive instance, the closer its score is to zero. Note that in SVMs, an instance's score corresponds to its distance from the hyperplane. Therefore, the closer it is to zero, the less sure the SVM is about its class. As

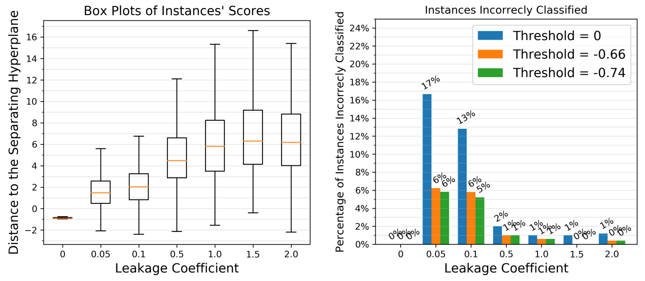

Figure 9: SVM's (linear) box plots of the instances' scores and the percentage of instances incorrectly classified.

the experiments in section 5.4 indicated, the percentage of FN is larger for instances with coefficients of 0.05 and 0.1 than for others. NB is also not as confident about the true class of these instances as it is for the others. Once again, the majority of negative instances have similar scores. However, some positive instances were able to get even lower scores, proving that these are hard to differentiate from the negative ones. Note that it might not be exclusively due to their size but also because of their location. Considering these results, we decided to move on to the next section using a threshold of -0.67 for NB and one of -0.66 for the SVM.

6.1.5 *Test Set Results.*

Since all iterations were coherent and the experiments were successful, we can now proceed to the final evaluation. As Table 2 shows, the results obtained using the test set are similar to the mean results of the cross-validation on the training set, proving that our classifiers are not overfitting and are generic enough to accommodate small changes in the data.

Table 2: Test set results for the synthetic setting.

Classifier	Correlation	TWS	#F	Threshold	$Prec_P$	$Recall_P$	F_P
NB	DCCA ($n = 1$)	40	43	-0.67	1.00	0.75	0.86
SVM (Linear)	DCCA ($n = 1$)	40	83	-0.53	1.00	0.98	0.99

To get a detailed view of the results, we plotted the percentage of misclassified instances by their leakage coefficient, as depicted in Fig. 10. Overall, the test set confirms that (1) some positive instances are difficult to differentiate from negative ones, and (2) although SVM is much better at identifying leakages, NB still manages to identify more than 80% of the positive instances with coefficients larger than 0.1.

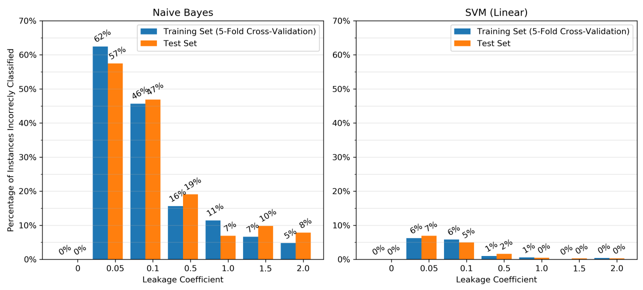

Figure 10: Percentage of instances incorrectly classified on the training and test set.

6.2 Real Water Distribution Network

Contrary to section 6, here we focus on understanding how a classifier performs under real conditions. Note that the real dataset obtained through feature construction has 33529 instances, where 33401 are negative (99.6%) and the remaining 128 are positive (0.4%).

6.2.1 *Initial Results.*

As in section 6.1.1, we started by gathering the results obtained with three different classifiers using DCCA and PCC as the correlation methods. Accordingly, Table 3 presents the mean results for an 11-fold cross-validation on the training set. Note

Table 3: Initial training set results for the real setting.

Classifier	Correlation	TWS	#F	$Recall_N$	$Prec_P$	$Recall_P$	F_P	L
NB	DCCA (n=29)	120	36	0.99	0.00	0.00	0.00	0.00
	PCC	120	36	0.98	0.01	0.04	0.01	0.27
SVM (Linear)	DCCA (n=29)	120	36	1.00	0.00	0.00	0.00	0.00
	PCC	120	36	1.00	0.00	0.00	0.00	0.00
RBF (RBF)	DCCA (n=29)	120	36	0.80	0.00	0.06	0.00	0.36
	PCC	120	36	0.68	0.00	0.33	0.01	0.55

that L represents the portion of leakages detected. For example, $L = 1$ indicates that all iterations of the 11-fold cross-validation correctly classified at least one positive instance. Although the goal is to correctly classify all positive instances, we only need to identify one per iteration to detect a leakage.

We can see that when the $recall_P$ increases, the $recall_N$ drops, meaning that the classifiers have to misclassify negative instances to identify positive ones. Although it is difficult for all three to differentiate them, the SVM (RBF) looks promising. However, and as expected, its cross-validation results are not consistent through all iterations. Several problems may cause this, including leakages with different profiles and only one leakage to validate per iteration.

6.2.2 *Reduction of Negative Instances.*

Since the class imbalance might be preventing our classifiers to learn, here we assess how their performance changes with the reduction of negative instances. We started our experiments with a 10% random sample of negative instances from our training set (3062 instances) until we reached an almost balanced training set, i.e., a 0.5% random sample (153 instances). Note that the chosen samples may not be representative of the class, causing the loss of crucial information. Fig. 11 shows the performance of NB and SVM (linear) along with the reduction of negative instances.

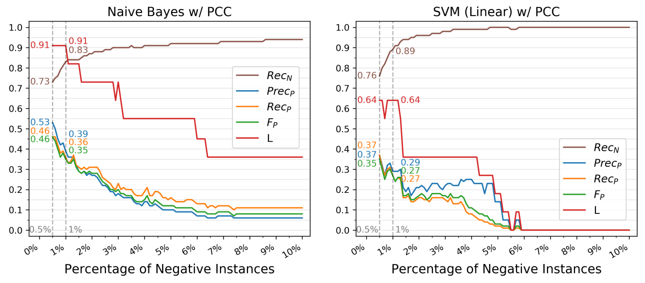

Figure 11: Classifiers' performance in the real setting along with the reduction of negative instances.

Although an SVM (RBF) did not benefit a lot from dataset reduction, NB and SVM (linear) performed better with a more balanced dataset. Additionally, contrary to the results obtained for the synthetic setting in section 6.1, these experiments confirmed that PCC results are better and more stable than DCCA's in this setting. One possible explanation is that DCCA could be more sensitive to variations, as shown in section 5.2, which is an advantage in a controlled scenario since most changes are leakages. However, in a real unstable scenario, this sensitivity could mislead the classifiers into thinking that negative instances are positive. Lastly, a sample of 0.5% (153 instances) produced better results across our three classifiers, but it was not generic enough to be consistent in all iterations of cross-validation. Thus, a sample of 1% (306 instances) is a safer choice even though its results are not as good.

Regarding the next sections' experiments, we decided to use PCC as the correlation method. Moreover, we will compare the results obtained when training with a complete set versus one with a 1% sample of negative instances.

6.2.3 *Feature Selection.*

Although this dataset does not have as many features as the synthetic one, it still could be useful to understand if any features are causing a performance decrease. Accordingly, Fig. 12 shows the performance of SVM (RBF) for a complete training set and NB for a 1% sample.

With a complete training set, the results dependent a lot on the features selected. Although this disparity does not allow us to be

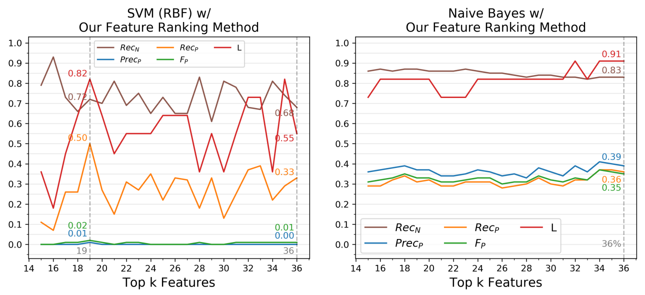

Figure 12: Classifiers' performance in the real setting along with the top k features selected.

sure about which features to choose, the 19 selected by our method got better and slightly more consistent results than with all features.

Regarding NB and SVM (linear) with a 1% sample, the results are very similar but it seems that including all features is a safer choice. As detailed in section 6.1.2, the classifiers' performance in the synthetic setting improves considerably after reaching a minimum of 30 features. Since we only have 36, we believe that our classifiers would benefit from having more features/sensors.

6.2.4 Time Window Size. Since we have been using a time window of 120 minutes, we prepared three additional ones of 60, 180, and 240 minutes, as shown in Table 4 to understand how different sizes affect the performance. Note that NI refers to the percentage of negative instances used in the training set.

Table 4: Classifiers' performance in the real setting using four different time window sizes.

NI	Classifier	Correlation Method	TWS	#F	$Recall_N$	$Prec_P$	$Recall_P$	F_P	L
1%	NB	PCC	60	36	0.74	0.36	0.43	0.37	0.91
			120	36	0.83	0.39	0.36	0.35	0.91
			180	36	0.83	0.34	0.39	0.36	0.64
			240	36	0.83	0.40	0.36	0.36	0.73
1%	SVM (Linear)	PCC	60	36	0.88	0.25	0.11	0.16	0.55
			120	36	0.89	0.29	0.27	0.27	0.64
			180	36	0.87	0.20	0.20	0.19	0.45
			240	36	0.90	0.31	0.21	0.23	0.55
100%	SVM (RBF)	PCC	60	19	0.79	0.01	0.25	0.01	0.91
			120	19	0.72	0.01	0.50	0.02	0.82
			180	19	0.88	0.00	0.01	0.00	0.18
			240	19	0.90	0.02	0.01	0.01	0.09

Overall, a 60 and a 120 minute window provided better results than larger windows. As we have seen in section 5.4, the correlation over time for larger windows looks more stable, possibly camouflaging the leakages. Since a 120 minute window obtained better results for the SVMs, we will keep using it. Note that the granularity of the synthetic dataset is only 10 minutes, whereas this one is 1 minute. So, although the synthetic window is larger in time, it contains fewer points. Therefore, the correlation methods need to use a larger window in the synthetic setting to capture the true correlation value between time series. Consequentially, we also do not need to use a window that large since the granularity of the real dataset is high enough.

6.2.5 Threshold Adjustment. As the last step of hyperparameter tuning, we assess (1) what the best performance our classifiers can achieve is, and (2) if the default threshold is preventing them from achieving optimal results. Accordingly, Fig. 13 presents NB's ROC plot and its performance along with different thresholds.

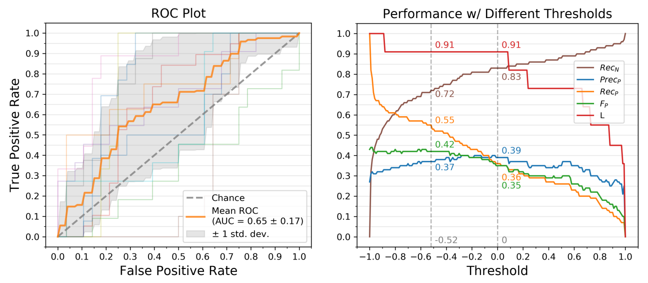

Figure 13: NB's ROC plot and its performance in the real setting (1% sample) along with different thresholds.

All iterations of our classifiers generated distinct ROC curves, some even worse than random, reflecting the disparity of the results between iterations and proving that the dataset is not generic enough. Moreover, all three mean curves are very close to the chance line, confirming that the classifiers show a lot of difficulty in differentiating classes. Consequently, we can not improve the $recall_P$ without misclassifying a large number of negative instances.

When assessing the threshold evolution, if we were to chose a different threshold for NB and the SVM (linear), we would technically improve their performance. However, since the iteration results are very distinct, we could be overfitting the data even more.

6.2.6 Test Set Results. During section 6.2 , we used an 11-fold cross-validation on the training set to assess the performance of our classifiers and tune its hyperparameters. The mean results obtained after the hyperparameterization and the ones obtained with the test set are summarized in Tables 5 and 6, respectively.

Table 5: Final training set results for the real setting.

NI	Classifier	Correlation	TWS	#F	$Recall_N$	$Prec_P$	$Recall_P$	F_P	L
1%	NB	PCC	120	36	0.83	0.39	0.36	0.36	0.91
1%	SVM (Linear)	PCC	120	36	0.89	0.29	0.27	0.27	0.64
100%	SVM (RBF)	PCC	120	19	0.72	0.01	0.50	0.02	0.82

Table 6: Test set results for the real setting.

NI	Classifier	Correlation	TWS	#F	$Recall_N$	$Prec_P$	$Recall_P$	F_P	L
1%	NB	PCC	120	36	0.57	0.00	0.22	0.00	1
1%	SVM (Linear)	PCC	120	36	0.59	0.00	0.11	0.00	1
100%	SVM (RBF)	PCC	120	19	0.54	0.00	0.44	0.01	1

As expected, the test set results differ from the training set ones. Additionally, a higher $recall_N$ results in a lower number of TP, meaning that we have to misclassify negative instances to identify the positive ones. Nonetheless, all classifiers identified at least one positive instance, so they all successfully detected the leakage.

Regarding the dataset imbalance, the results obtained with a 1% sample of negative instances are similar to the ones obtained with the SVM (RBF). However, it is important to note that both NB and the SVM (linear) performed better with fewer negative instances. Therefore, we believe that reducing the negative instances is promising, but the inconsistency of the positive instances is preventing the classifiers from achieving satisfactory results.

Overall, the test set confirms that it is very difficult to learn the difference between positive and negative instances in a real WDN. We identified several problems that may contribute to these results:

- *Low number of leakages* – Since we do not have many examples of what a leakage is, our classifiers can not generalize that information and end up overfitting the data;

- *Lack of information regarding the size and actual beginning of the leakage* – The results in section 5.3 show that it is easier to detect larger leakages, especially when the time window covers its beginning. For example, our leakages could be small, thus preventing our classifiers from learning and detecting them;
- *Location of leakages* – Leakages occur in different WDN points, affecting the correlations between downstream and upstream sensors, which hampers the generalization of leakage dynamics;
- *Poor sensor coverage* – Section 6.1.2 showed that the performance of the classifiers improved considerably with more than 30 features. Additionally, the synthetic dataset is also composed of data from sensors placed throughout the network. Therefore, we believe that our classifiers would benefit a lot from sensor expansion and relocation;
- *Network changes and interventions* – They can disrupt the natural behavior of the WDN, making the learning process more difficult.

6.3 Conclusions

Our work explored and presented a solution to detect burst leakages in WDNs through the classification of time series data collected by their sensors. The solution used the correlation values between pairs of sensors along time as descriptors of network dynamics for later predict the leakage dynamics, which can be easily generalized to other WDNs. Our experiments were conducted in a controlled setting without noise and network changes, and in a real challenging setting with highly irregular consumption patterns.

As expected, the real setting results are more inconsistent than the synthetic ones since the leakages we used have different profiles and the cross-validation only has one leakage to validate per iteration. However, we were still able to draw interesting conclusions. First, the maximum disruption of the correlation happens when the middle of the time window aligns with the beginning of the leakage. Moreover, DCCA seems to be more sensitive to variations than PCC. The classifiers needed a minimum of 30 features to achieve desirable results, and increasing the size of the window does not always enhance their performance. Both our feature relevance ranking method and Kruskal–Wallis H test provided good results depending on the setting. We also believe that the reduction of negative instances is promising, but the inconsistency of the positive ones is preventing the classifiers from achieving desirable results. Overall, the SVMs obtained better results than NB, but the latter still proved to be a very competitive classifier in a real setting and benefited the most from dataset reduction. Lastly, we believe that our classifiers would benefit a lot from sensor expansion and relocation.

Concluding, it is very difficult for classifiers to learn the difference between positive and negative instances. We identified several problems that may contribute to these results, namely, (1) the low number of leakages to learn from and consequent class imbalance, which would eventually disappear with time since the classifiers can be updated, (2) the lack of information regarding the size and actual beginning of the leakage, (3) the location of leakages, (4) poor sensor coverage, and (5) network changes and interventions that disrupt the natural behavior of the network.

In the future, we expect to apply our solution in other Portuguese WDNs and develop a classifier that is better prepared to handle the nature of the gathered descriptors (correlation-based features) and their inherent spatiotemporal dependencies. Additionally, we also intend to comprehensively analyze the correlation disruptions and predictive behavior to identify the location and size of leakages, detect background leakages, support the placement of new sensors and the detection of status' changes in active hydraulic elements.

REFERENCES

[1] K Aksela, M Aksela, and R Vahala. 2009. Leakage detection in a real distribution network using a SOM. *Urban Water Journal* 6, 4 (2009), 279–289.

[2] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 3 (2017), 606–660.

[3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control.* John Wiley & Sons.

[4] Nelson Carriço, Maria Amado, Dídia Covas, Jacinto Estima, José Borbinha, José Figueira, Laura Monteiro, Maria Barreira, Rui Henriques, Sérgio Fernandes, and Susana Vinga. 2018. Water Intelligence System Data Project. Active period: 2018-present. Lisbon, Portugal: Fundação para a Ciência e Tecnologia (FCT). Public information available in https://www.fct.pt.

[5] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28.

[6] Andrew Colombo and Bryan Karney. 2002. Energy and Costs of Leaky Pipes: Toward Comprehensive Picture. *Journal of Water Resources Planning and Management* 128 (11 2002), 441–450. https://doi.org/10.1061/(ASCE)0733-9496(2002)128:6(441)

[7] Joost CF de Winter, Samuel D Gosling, and Jeff Potter. 2016. Comparing the Pearson and Spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological methods* 21, 3 (2016), 273.

[8] EPA. 2019. Drinking Water Distribution Systems. https://www.epa.gov/dwsixyearreview/drinking-water-distribution-systems Accessed in: 18 Oct 2019.

[9] Malcolm Farley and Stuart Trow. 2003. *Losses in water distribution networks.* IWA publishing.

[10] Malcolm Farley, Sanitation Water, Water Supply, Sanitation Collaborative Council, World Health Organization, et al. 2001. *Leakage management and control: A best practice training manual.* Technical Report. Geneva: World Health Organization.

[11] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. 2008. *Feature extraction: foundations and applications.* Vol. 207. Springer.

[12] Gary Heiman. 2013. *Basic Statistics for the Behavioral Sciences.* Cengage Learning.

[13] Martin Hofmann. 2006. Support vector machines-kernels and the kernel trick. *Notes* 26, 3 (2006).

[14] Yih-Wenn Laih. 2014. Measuring rank correlation coefficients between financial time series: A GARCH-copula based sequence alignment algorithm. *European Journal of Operational Research* 232, 2 (2014), 375–382.

[15] Ellen J Lee and Kellogg J Schwab. 2005. Deficiencies in drinking water distribution systems in developing countries. *Journal of water and health* 3, 2 (2005), 109–127.

[16] Stef Lhermitte, Jan Verbesselt, Willem W Verstraeten, and Pol Coppin. 2011. A comparison of time series similarity measures for classification and change detection of ecosystem dynamics. *Remote sensing of environment* 115, 12 (2011), 3129–3152.

[17] Rui Li, Haidong Huang, Kunlun Xin, and Tao Tao. 2015. A review of methods for burst/leakage detection and location in water distribution systems. *Water Science and Technology: Water Supply* 15, 3 (2015), 429–441.

[18] Patrick E McKight and Julius Najab. 2010. Kruskal-wallis test. *The corsini encyclopedia of psychology* (2010), 1–1.

[19] Ofwat. 2019. Leakage. https://www.ofwat.gov.uk/households/supply-and-standards/leakage Accessed in: 9 Oct 2019.

[20] Boris Podobnik and H Eugene Stanley. 2008. Detrended cross-correlation analysis: a new method for analyzing two nonstationary time series. *Physical review letters* 100, 8 (2008), 084102.

[21] Irina Rish et al. 2001. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, Vol. 3. 41–46.

[22] Sima Valizadeh, Behzad Moshiri, and Karim Salahshoor. 2009. Leak detection in transportation pipelines using feature extraction and KNN classification. In *Pipelines 2009: Infrastructure's Hidden Assets.* American Society of Civil Engineers, 580–589.

[23] Andrew J Whittle, Michael Allen, Ami Preis, and Mudasser Iqbal. 2013. *Sensor networks for monitoring and control of water distribution systems.* International Society for Structural Health Monitoring of Intelligent.

[24] Donald F Young, Bruce R Munson, Theodore H Okiishi, and Wade W Huebsch. 2010. *A brief introduction to fluid mechanics.* John Wiley & Sons.

[25] Mohammed J Zaki, Wagner Meira Jr, and Wagner Meira. 2014. *Data mining and analysis: fundamental concepts and algorithms.* Cambridge University Press.