PROJECT DELIVERABLE REPORT

NAIADES

Greening the economy in line with the sustainable development goals

D5.5 – Water Demand Prediction Toolkit – Mid-term

A holistic water ecosystem for digitisation of urban water sector

SC5-11-2018

Digital solutions for water: linking the physical and digital world for water solutions

"This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 820985"

Document Information

Grant Agreement Number	820985	Acronym	NAIADES
Full Title	A holistic water ecosystem for digitization of urban water sector		
Topic	SC5-11-2018: Digital solutions for water: linking the physical and digital world for water solutions		
Funding scheme	IA - Innovation action		
Start Date	1stJUNE 2019	**Duration**	36 months
Project URL	www.naiades-project.eu		
EU Project Officer	Alexandre VACHER		
Project Coordinator	CENTER FOR RESEARCH AND TECHNOLOGY HELLAS - CERTH		
Deliverable	D5.5 – Water Demand Prediction Toolkit – Mid-term		
Work Package	WP5 - NAIADES Smart Framework: AI analytics and predictive services		
Date of Delivery	**Contractual**	M18	**Actual** M18
Nature	R - Report	**Dissemination Level**	PU-PUBLIC
Lead Beneficiary	Konnektable Technologies Ltd. (KT)		
Responsible Authors	Klemen Kenda	Email	klemen.kenda@ijs.si
	Georgia Lytra	Email	glytra@konnektable.com
Reviewer(s):	Cédric Crettaz		
Keywords	Data Fusion, Machine Learning, Incremental Learning, Anomaly Detection		

Revision History

Version	Date	Responsible	Description/Remarks/Reason for changes
0	31/10/2020	Klemen Kenda	ToC, initial inputs
1	09/11/2020	Gal Petkovšek, Gregor Kržmanc, Jože Peternelj, Klemen Kenda	Deep learning inputs, architectural inputs, data fusion inputs, general learning inputs
2	10/11/2020	Matej Čerin, Matej Posinkovič, Klemen Kenda	Results from modeling experiments, Use case definitions
3	11/11/2020	Georgia Lytra, Klemen Kenda	Long-term predictions, finalization of the draft version of the deliverable
4	15/11/2020	Georgia Lytra, Matej Čerin, Matej Posinkovič, Klemen	Finalisation of the deliverable and initiation of the internal review process

		Kenda, Nikos Angelopoulos	
5	16/11/2020	Cédric Crettaz	Internal review
6	18/11/2020	AIMEN	Internal review
7	30/11/2020	JSI, KT	Final draft

Contents

Abbreviations

ADWIN	ADaptive sliding WINdow drift detection algorithm
AIC	Akaike information criterion
ANN	Artificial Neural Network
API	Application Programming Interface
AR	AutoRegression
ARIMA	AutoRegressive Integrated Moving Average
CNN	Convolutional Neural Network
D-K Statistics	Dickey – Fuller Statistics
GBM	Gradient Boosting Machine
GUI	Graphical User Interface
JSON	JavaScript Object Notation
kNN	k-Nearest Neighbor
LSTM	Long-Short Term Memory
MA	Moving Average
MAPE	Moving Average Percentage Error
ML	Machine Learning
RF	Random Forest
RMSE	Root Mean Squared Error
SARIMA	Seasonal ARIMA
SVM	Support Vector Machine

1 Summary

This report presents several classical and cutting-edge approaches to water demand prediction. It includes long-term prediction based on statistical approaches, such as ARIMA and SARIMA, it includes short-term forecasting methodologies that exploit traditional machine learning workhorses like linear regression, random forests and gradient boosting and finally it presents approaches based on deep-learning technologies. Among modeling approaches, the classical one outperforms the deep learning since deep learning requires more data or more repetitions.

Report also presents additional machine learning tasks such as real-time heterogeneous sources data fusion, feature selection and hyper-parameter tuning. We will provide the use of methodologies that are the cutting edge and will prepare some improvements.

Finally, report provides conceptual architecture and module design proposals for a real-world implementation of the system. Some of these components have already been partially developed and will be updated until the end of the task and will be reported in the final report.

2 Introduction

The scope of this deliverable is to train different models and make predictions from historical data regarding the amount of water demand in a residential area. The scope of the deliverable is quite broad and addresses several issues in machine learning/statistical modelling of time series. The work is based on the task definition from the Grant Agreement (GA):

> **T5.3 Water demand prediction toolkit (M3-M18 & M20-M30)**
>
> Forecasting the dynamic behaviour of the buildings is an essential part for the overall NAIADES system. This task implements a novel forecasting system incorporating the necessary algorithms and knowledge to provide efficient and robust water demand predictions based on the available profiles and models. As concerns the weather forecast, Web services will be used to retrieve information on-line. During this task, various modelling techniques will be investigated (e. g. clustering analysis, principal component analysis regression, partial least square regression, multivariate analysis of variance, etc.) that could be deployed for optimal predictions. Within this task, the requirement is to obtain the water demand for the next few hours so as to run multiple iterations to ensure the confidence parameters with the minimum cost and/or usage of water. The outcomes of the simulations, this are the forecasted demands, and their gap with the real water needs will be minimized by the application of machine learning techniques. The machine learning techniques implemented will match forecasted and measured values considering boundary conditions as weather forecast, type of day, building demand profiles etc. The application of the mentioned techniques will enable NAIADES as self-tuning approach to buildings demands forecasting objective.

We have not only followed the task description, but have looked at the machine learning approaches broader. We have been thinking about various other problems such as:

- Real-world real-time data fusion from heterogeneous IoT and external data sources
- Feature engineering
- Feature selection
- Hyperparameter tuning
- Usage of deep learning models within water scenarios – pros & cons

Progress of the task is reported in 2 stages (in M18 and in M30). This deliverable therefore presents the following:

- Theoretical introduction into modeling approaches
- Conceptualization of the stream mining framework, with the emphasis on streaming data fusion
- Internal API definitions
- Initial use case analysis regarding prediction

Prototypes have already been implemented in this initial phase of the project. They will be finalized and integrated in the second phase (by M30). The prototype is a loosely coupled component of the NAIADES framework and integrates into the overall system by using specific NAIADES APIs.

The report is structured as follows. Current section presents related work within the field of machine-learning based modeling in the water management domain. Section 3 presents the conceptual architecture of a Big Data water management system, Section 4 presents various modeling approaches from ARIMA to Deep Learning and corresponding algorithmic tools to provide more accurate models. Section 5 is dedicated to the definition of a stream mining pipeline that will implement the methodologies from Section 4. Section 6 provides internal API definitions. Section 7 provides an overall evaluation of modeling approaches in a ground/surface water scenario (this data was available already in the beginning of the project and was big enough to provide a thorough statistical analysis). Section 8 is dedicated to the

laboratory tests of provided methodologies on NAIADES data. Finally, Section 9 concludes the report. Bibliography is given in the end.

Note: This deliverable is based strongly on the findings of 2 scientific papers studying the incremental learning techniques and appropriate architectures within the water management domain [1], [2]. Both papers have been fully or partially prepared within the scope of NAIADES and are disseminating the work accomplished in this task. Therefore, this report can contain longer excerpts from those papers (e. g. in related work, experimental results, etc.) without additional citation.

2.1 Related Work

Note: Related work has been prepared in [1] and is provided in the same form in this subsection.

Water data is becoming increasingly accessible and low-cost. Investments in improvement of data acquisition and data transfers have enabled significant growth of knowledge-intensive economies [3]. However, there is still a lot of room for improvement, especially compared to the energy or transportation sectors, as indicated by the expected future infrastructure costs by sector [4]. On the contrary, water as a resource itself is becoming a more expensive commodity. Water utilities worldwide are incorporating – or have already incorporated - the opportunity costs of capital, operation, maintenance, and environmental impacts to the final price under the Polluter-Pays and the User-Pays principles, commonly accepted by the OECD countries. Digitalization has penetrated most areas of human activity, including major manufacturing facilities, energy markets, health care and even well-being, while various methodologies on improving resources management and optimizing consumption, usage or exploitation systems have been tested in various settings producing positive results. Water management digitalization process is showing great potential for the usage of modern technologies such as the Internet of Things (IoT) and Artificial Intelligence (AI). The latter can operate as a catalyst for investigating, understanding, forecasting, and optimizing water usage, leakage, fraud, and pollutant detection, flooding and damage prevention and protection [3].

AI methodologies, especially statistical modeling techniques from the family of machine learning (ML) algorithms, have proven to successfully complement or even replace the traditional process-based models, usually, requiring much less data preparation and computing time; they are therefore more easily implemented in real-world scenarios [5]–[7]. Machine learning techniques have also proven to effectively catch patterns that involve complex interdependencies and non-linearities such as those found in the interdisciplinary boundaries of aquatic systems and ecological systems [8]. On the other hand, process-based models are more generalized and provide results that can be applied in broader areas, whereas machine learning models usually target a specific point in space. One should note that machine learning models are completely data-driven. This means that expert knowledge is required only to select and transform relevant data and their derivatives into meaningful inputs; and for the conclusive phase of evaluating, validating and interpreting the outputs. Underlying processes are agnostically modeled by the ML methods. This can be perceived as a benefit or as a drawback, since on one hand, it constitutes the modeling process practically easier, on the other hand, it deprives some of the interpreting function of building up the causal-effect relations. The above reveals a great potential (research and practical) and value for combining machine learning with process models in a similar way to "injecting humans-in-the-loop" when modeling with machine learning, either by interactive training or interactive feature selection [9], [10].

During the last decades, scientific literature is overwhelmed by laboratory tests of machine learning methodologies, as the aforementioned benefits of such techniques have attracted researchers (Maier & Dandy, 2000). Often, data preprocessing (including cleaning and data fusion) in such applications is done manually offline. However, transferring the machine learning applications to real-world scenarios would

require automated data processing pipeline from the data source (sensor or web resource, e.g. for weather and weather forecast data) to the final user [11].

IoT and other sensor data itself are useful and can be applied to many scenarios, however – much better results can be obtained when using multiple interconnected data sources. For example: predicting groundwater levels in the future will benefit significantly from weather situation and weather forecast data and perhaps in some special cases from the water withdrawal and population modeling data. This means that multiple heterogeneous data sources have to be combined in real time to achieve the best possible results.

To the best of our knowledge, in the scope of this work another family of machine learning techniques is introduced to water domain: incremental learning (sometimes referred to also as stream mining). This is specifically suitable for learning from continually generated sensor data as the models are updated with each subsequent measurement without extensive use of computer resources as in traditional batch methods, which require re-learning on the whole historical dataset. Savings are obtained in data storage, computing power and time.

Slovenia is a country with a dense hydrographical network, with a great difference in the amount of precipitation between areas in the east and west, with areas of regular or occasional flooding or drought and a positive balance between incoming and outcoming waters. The population density and its related pressure on the aquatic environment also differ. Water landscape is affected by the anticipated climate change, which is causing longer lasting spring and summer droughts as well as less and more imbalanced precipitations. Despite an overall favorable water balance, shortages can be expected in 15% of country's surface area, mostly in the north-eastern part[1], as well as floods on another 15% of the territory. Since 1992, seven summer droughts have hit agriculture. In 2003, 2.4% of population required water to be supplied with a tanker.

Accurate ground and surface water level short term predictions allow to better understand its dynamics and convey information about coming extreme events; identify factors that affect water consumption as well as optimize operating schedules of related infrastructure [5]. This information also allows to better plan in a context of greater water scarcity [12] and manage the resource in new ways (e.g. by establishing dynamic pricing [13]) in order to increase sustainability.

2.2 Context of D5.5 within the NAIADES Technical Architecture

The task T5.3 is strongly coupled with T5.1 (reported in NAIADES D5.1 – Failure and Leakage Prediction Engine – mid-term and NAIADES D5.2 – Failure and Leakage Prediction Engine – final). Both tasks share some of the technology (e. g. data fusion component is used also for modeling and batch/incremental models can also be exploited for anomaly detection). Both deliverables are therefore strongly intertwined and should be considered in combination.

With respect to NAIADES' technical architecture which is described in detail in deliverable D2.9, the water demand prediction services presented in this deliverable are positioned as shown in Figure 1 highlighted inside the red rectangle.

The water demand prediction toolkit makes use of the weather data, the water consumption data and plant-related input data generated by the NAIADES sensors networks through the data management framework and the FI-WARE NGSI context broker in order to deliver predictions on the water demand and consumption for a specific area.

[1] https://www.arso.gov.si/en/soer/freshwater.html

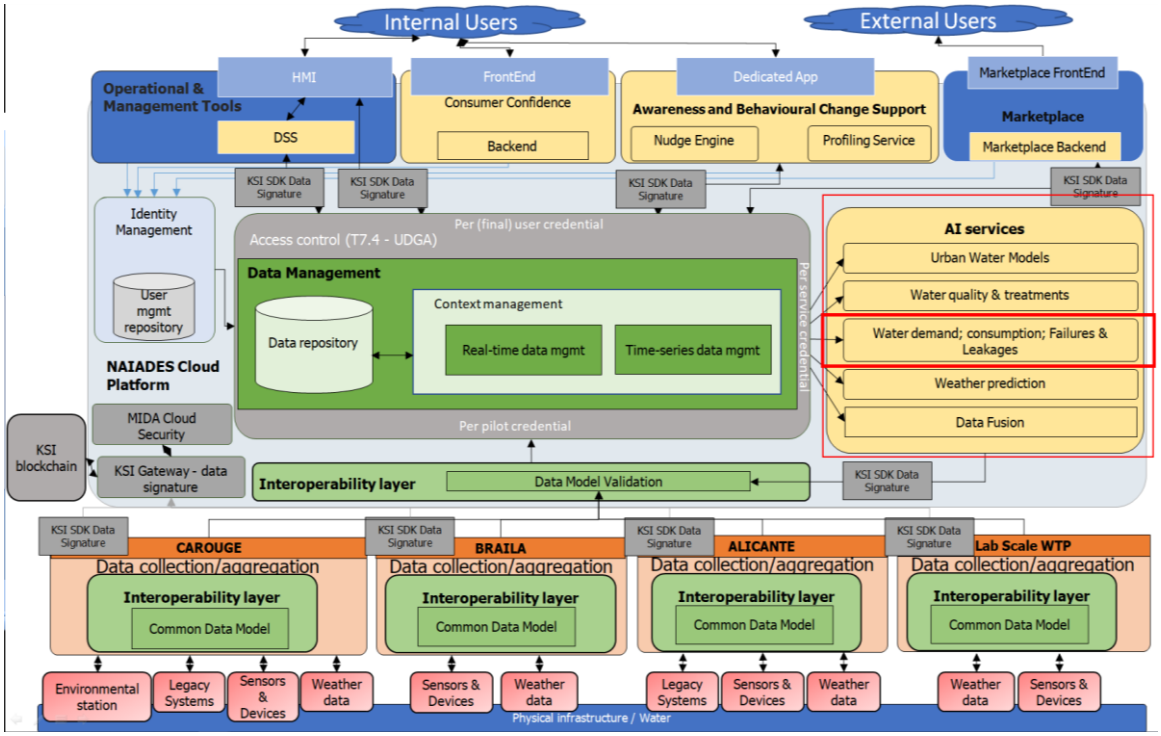

Figure 1: NAIADES' Technical Architecture

The interaction of the toolkit with the database will be done using GET/POST methods and the outputs of the toolkit will be available through a RESTful API with JSON content (examples of the JSON files can be found in Chapter 6.1).

3 Conceptual Architecture for Big Data Processing

We base our architecture on the Big Data *lambda architecture* [14] concept, which was extended with the hut architecture [15]. Both of these architectures consider the single-entry point for the data and then two separate pillars. The first pillar is named "Speed Layer" and the second "Batch Layer". In lambda architecture, both pillars work independently. Speed layer is based on streaming data and in charge of complex event processing while batch layer is based on batch methods and is in charge of modeling. Our contribution to this architecture [2] is in the speed layer, where we construct a pipeline that can handle advanced analytics scenarios in a streaming manner. We bring functionalities of a batch layer into the speed layer and enable fast generation of predictions (and even learning of the incremental models) in the speed layer.

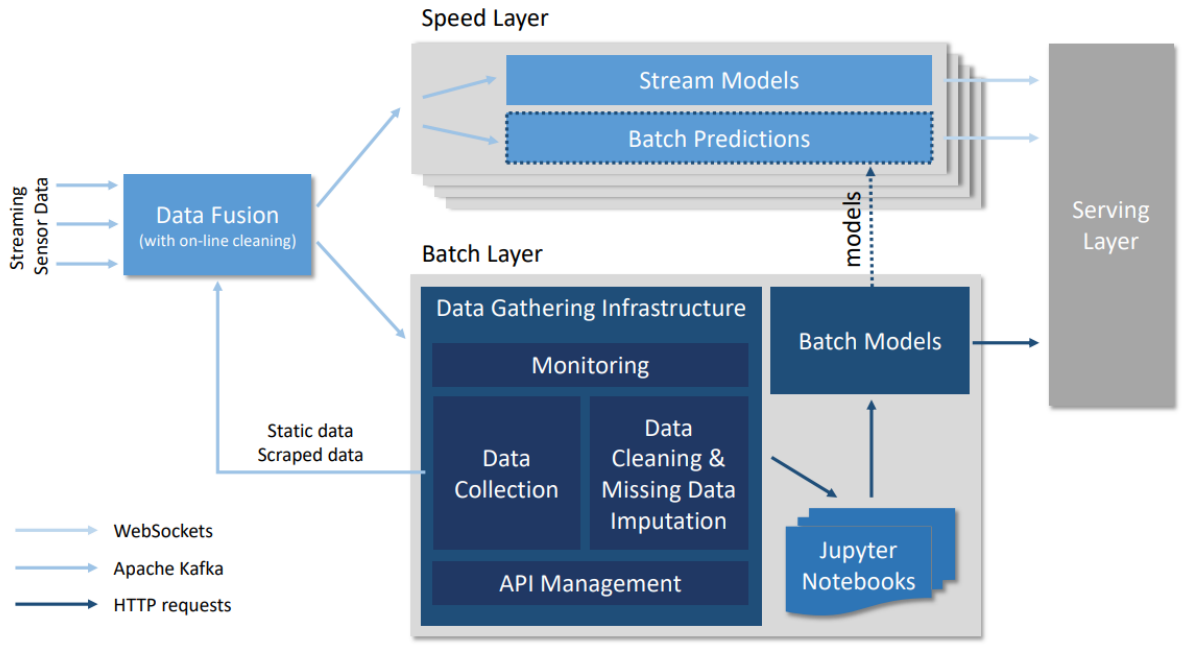

Figure 2: Proposed modification of lambda and hut architectures for usage in water management domain. Speed layer includes types of models - stream and batch. Batch models are updated from the batch layer. Heterogeneous input time series are fused in the Data Fusion component and pushed into both layers.

Section 0 is dedicated to analysis methodologies within the batch layer and Section 0 depicts basics of the speed layer.

4 Modeling Approaches for Batch Layer

In general, data-driven approaches still include basic principles derived by CRISP-DM[2]. The process includes all the stages from data retrieval, data and domain understanding, data preparation (most time-consuming part), model training, evaluation and of course optimization of all these steps. The process should be concluded by deployment of the data-driven statistical model. Our approach in the initial exploratory phase of NAIADES project is described in the figure below.

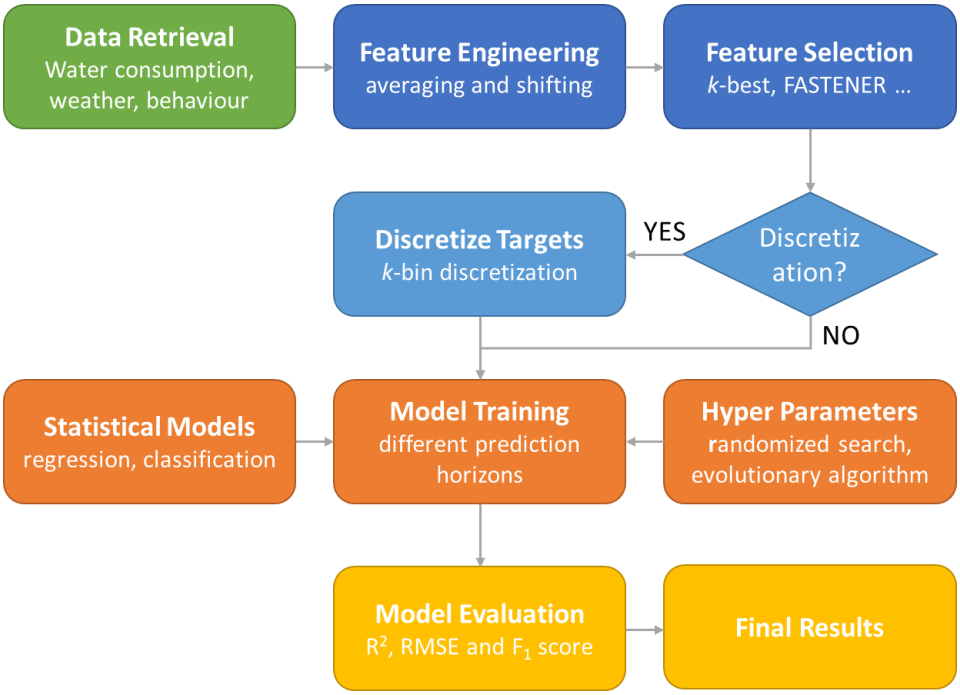

Figure 3: The workflow of the data-driven approach includes data acquisition, feature generation and selection, modeling and evaluation tasks.

Several sources of the data within the project have been acquired and analyzed. As the datasets are still sparse, an additional data set on groundwater (and also on surface water) data from Slovenia has been acquired for extensive experiments. We tried to understand the underlying processes in each of the use cases. That knowledge has driven feature engineering. As we have provided extensive feature sets, we have developed appropriate feature selection algorithms. We still plan to include more sophisticated methods like FASTENER [16]. FASTENER is our in-house developed method for Earth Observation but has shown promising results also on other use cases. Such an evolutionary approach would for sure be much more successful for feature selection than just a simple correlation-based approach like k-best. In our experiments we have also tried to find out whether discretization of the target space (we usually have a continuous target space and we use regression) would benefit us, since many more methods could be used. As expected, this has not proven to be a successful approach, but we have evaluated it rigorously. After all this, model training for different prediction horizons is performed, models are improved based on hyper-parameter tuning (we propose a new evolutionary algorithm to accomplish this task better than current standard method and this algorithm should be realized in the next project period). Finally, we evaluate the models and select the best candidate, suitable for deployment in the real world.

[2] https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining

4.1 ARIMA and SARIMA Approaches for Long Term Prediction

A **time series** is a sequence of numerical data points in successive order. It tracks the movement of the chosen data points, over a specified period and records the data points at regular intervals.

A time series can be broken down into 3 components.

- Trend: Upward and downward movement of the data
- Seasonality: Seasonal variance
- Noise: Spikes and troughs at random intervals

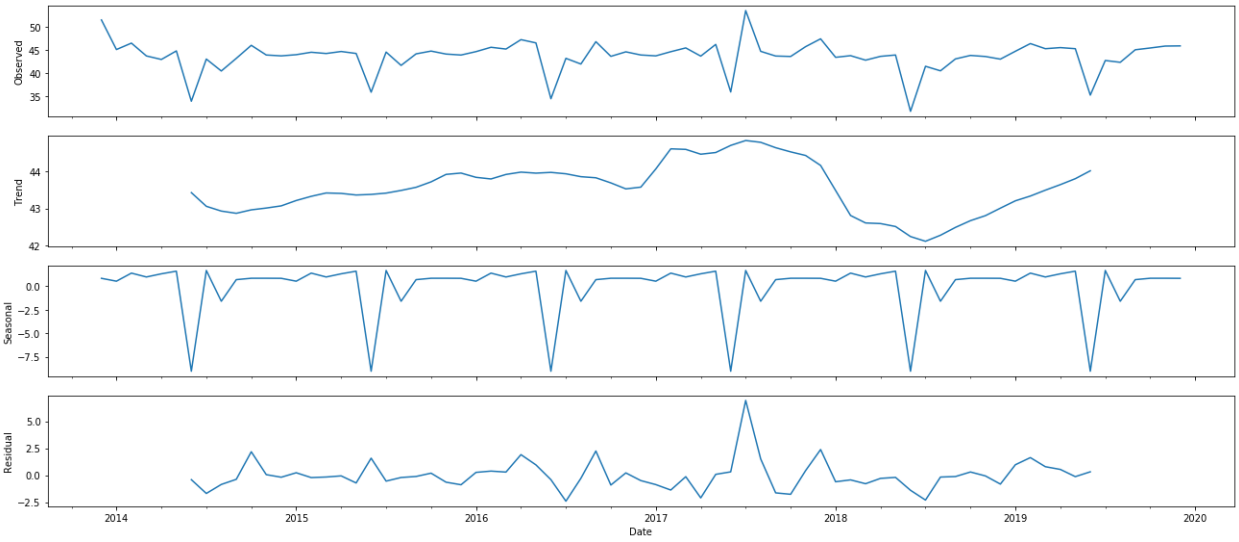

Figure 4: Monthly decomposition of time series.

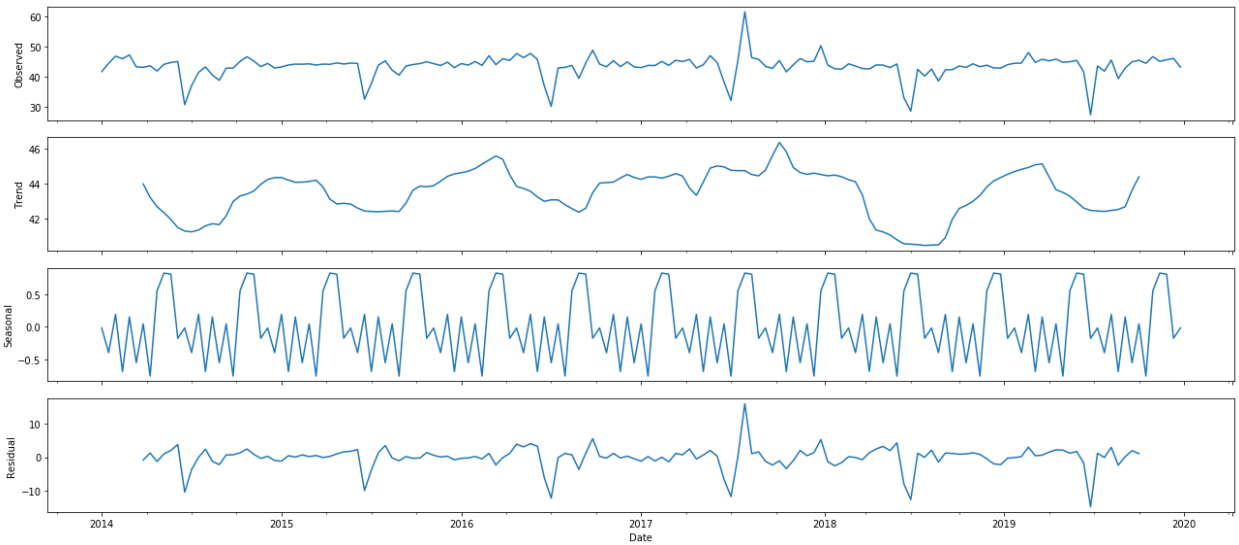

Figure 5: Biweekly decomposition of time series.

As it is observed from Figure 4 and Figure 5, both the time series have a seasonality since there is a pattern showing every year but no clear and visible trend. The residuals also seem to be revolving close to zero and only go off in extreme cases where data have abnormal values.

Time series forecasting is a technique for the prediction of events through a sequence of time. The techniques predict future events by analyzing the trends of the past, on the assumption that future trends will hold similar to historical trends. In the long-term approach, the prediction is based on this very same technique where past measures of water consumption help predict feature values of water demand in this specific region.

Before applying any statistical model on a time series, we want to ensure that it's stationary and doesn't have a time-dependent structure. This will be done with help of the Dickey - Fuller test.

The **Dickey-Fuller test** [17] is a type of statistical test called a unit root test which is a test that determines how strongly a time series is defined by a trend. It uses an autoregressive model and optimizes an information criterion across multiple different lag values. It uses a hypothesis test to determine whether the time series is stationary or not:

- Null Hypothesis (H_0): If failed to be rejected, it suggests the time series is non-stationary. It has some time dependent structure. This is the case when p-value > 0.05.
- Alternate Hypothesis (H_1): If the null hypothesis is rejected, it suggests the time series is stationary. It does not have time-dependent structure. This is the case when p-value < 0.05.

The last step we followed before moving to predictive modeling techniques, is to divide the data into training and test sets to avoid the overfitting of the data and test the final results.

Model: One of the most common methods used in time series forecasting is known as the ARIMA [18]–[21] model, which stands for **AutoRegressive Integrated Moving Average**. ARIMA is a model that can be fitted to time series data to better understand or predict future points in the series.

The ARIMA forecasting for a stationary time series is a linear equation. The predictors depend on the parameters (p, d, q) of the ARIMA model:

- number of AR (Auto-Regressive) terms (p): p is the auto-regressive part of the model. It allows us to incorporate the effect of past values into our model. AR terms are just lagging of dependent variable.
- number of Differences (d): d is the Integrated component of an ARIMA model. This value is concerned with the amount of differencing as it identifies the number of lag values to subtract from the current observation.
- number of MA (Moving Average) terms (q): q is the moving average part of the model which is used to set the error of the model as a linear combination of the error values observed at previous time points in the past. MA terms form lagged forecast errors in prediction equation.

These three distinct integer values, (p, d, q), are used to parametrize ARIMA models. Because of that ARIMA models are denoted with the notation ARIMA(p, d, q). Together these three parameters account for seasonality, trend, and noise in datasets.

Seasonal ARIMA (SARIMA) model is used when the time series exhibits seasonality. This model is similar to ARIMA models and it is written as ARIMA(p, d, q)(P, D, Q)m where:

- (p, d, q) are the non-seasonal parameters described above.
- (P, D, Q) follow the same definition but are applied to the seasonal component of the time series.
- The term m is the periodicity of the time series

Seasonal differencing considers the seasons and differences the current value and its value in the previous season.

The SARIMA model formulation includes four steps which are the same steps that were followed in the prediction and building of our model:

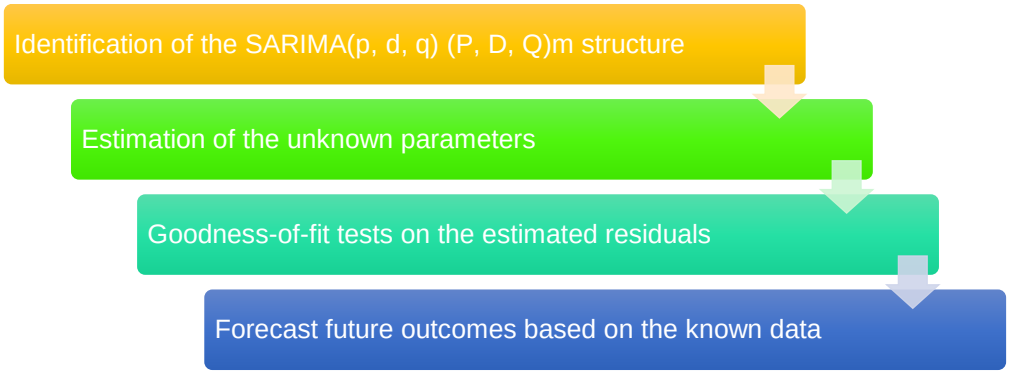

Figure 6: Workflow for making predictions based on SARIMA models.

Since we are interested in making monthly but also biweekly forecasts the periodicity of the models will be m = 12 and m = 24 respectively where it concerns the structure of the model.

Hyperparameters estimation: A critical part in the process of training an ARIMA model is the right selection of the hyperparameters which are the parameters that are determined before the application of the model. To address this problem **hyperparameter optimization** took place to select the best parameters for the proposed model with respect to the given set of data. The method that was used in our case to explore different combinations of these parameters was grid search process.

Grid search is a tuning technique that is widely used for its efficiency and timesaving that attempts to compute the optimum values of hyperparameters of a model. It is an exhaustive search that is performed through a set of possible parameter values for a model, with the scope to find the optimum values with respect to a metric that measures the fitting of the parameter to the model. The metric that we used for the SARIMA model was the Akaike Information Criterion.

The **Akaike Information Criterion (AIC)** is a method for evaluating how well a model fits the data it was generated from and is most often used for model selection. By calculating and comparing the AIC scores of several possible models, we choose the one that is the best fit for the data. AIC is defined as follows:

$$AIC = -2*\ln(\hat{L}) + 2*k$$

where k is the number of the estimated parameters of the model and $\hat{L}$ is the value of the maximum likelihood function for the model. AIC measures both the fit of a model and the unreliability of a model; the former is measured by the first term of the definition, which decreases as the number of parameters within the model is increased and the latter by the second term which increases with the number of parameters.

The **implementation** of the long-term analysis and modelling was made using Python, for its wide variety of capabilities and libraries like Pandas (for data wrangling), scikit-learn (for basic modeling), matplotlib (for visualization), stats models (for statistical modeling).

For the development of the long-term prediction model we used historical data. These have been stored in files on the local file system in a short of csv format. Therefore, the processing and the analysis that has been applied on them follows a batch processing model.

4.2 Batch Learning with Re-training

Note: This work has been prepared in [1] and is provided in the same form in this subsection (main part), we extended this with deep learning approaches additionally.

As opposed to traditionally used process-based models, data-driven models rely solely on data. The underlying dynamics of a water system is modeled latently. The model is being trained from the data itself and usually does not require external domain knowledge. Domain knowledge can lead to a significant increase of the model accuracy; however, it is introduced into the model through the appropriate selection of data sources and through appropriate transformation of the data (e.g. relevant non-linear combinations of available sensor data help significantly when trying to improve linear models).

Machine learning is a subfield of the wider artificial intelligence field. The discipline has been blooming since mid-1970-ies and has provided widely used solutions such as machine language translation, typing assistant, spam mail identification, image recognition and many others.

In water management, machine learning has been used for predicting various key variables in water systems such as groundwater levels, urban water demand in multiple scales from household to residential, urban water consumption behavior [22], anomaly detection such as fraud incidents, leakage in water distribution networks, stratification in reservoirs and many more. Often machine learning research in water management is focused on a particular method, the selection of which is not necessarily justified. There are few research papers, which really investigate a wider variety of algorithms and even among these, very few address the fact, that the usage of a particular modeling algorithm does not influence final results as much as the appropriate use of contextual data [23] and that ensuring proper feature generation and data fusion in real-time (in live, real world systems) is a great challenge until today [24].

The usual machine learning tasks in environmental data analysis include solving regression and classification problems, which are a part of the family of supervised learning, and clustering, which is part of the family of unsupervised learning algorithms. Supervised learning is performed on labelled data (e.g. groundwater level data, where target values to be modeled are known) whereas unsupervised learning can be used in data, where the target values (e.g. data about users, where the stakeholders would like to discover families of users, which behave approximately the same) are not known. The work reported in this paper tackles regression problems (prediction of numerical values, e.g. of groundwater and surface water levels). These problems were also converted into classification problems (e.g. by dividing groundwater level change into different classes and trying to predict those instead of a continuous value), but those did not yield competitive results.

The most relevant and widely used methods in environmental data-driven modeling nowadays are random forest, gradient boosting and deep learning, which are based on simpler building blocks like decision trees and perceptrons. It should be noted that water management modeling often assumes regression problems, for which deep learning does not exhibit as much power as with other problems (e.g. image recognition, text translation and similar). According to the nature of a dynamic water system also linear models like linear regression and support vector machines (SVM classifier or SVC and SVM regressor or SVR) with linear kernel can achieve very good results while preserving low computational cost. Quite often very simple methods like k-Nearest Neighbors can yield good results.

When using batch learning techniques in real world, we need to consider re-training. Re-training must occur frequently and – depending on the concept drift in the data – must contain only the relevant data. For example, if target value distribution does not change over time, then no re-training is needed (only to

get more accurate estimates within the current probability distribution), however, if target value distribution changes, re-training should consider only data that is relevant to the current distribution (if possible and if there is enough data).

4.2.1 Deep Learning Approach – LSTM time series prediction

Artificial neural networks are machine learning models, inspired by biological neural networks (animal brain). The most basic elements of a neural network are neurons, which are arranged into multiple layers. The first layer (input layer) is where the data comes in and the last layer (output layer) is where the network's prediction is outputted. In between there are hidden layers. If a network has multiple hidden layers, we call it a deep neural network.

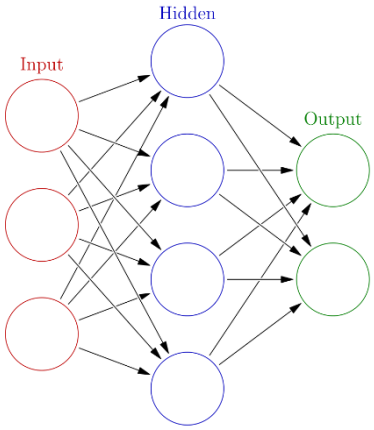

Figure 7: Neural network (source: https://en.wikipedia.org/wiki/Artificial_neural_network)

A subclass of deep neural networks are recurrent neural networks (RNNs). In this case the output layer is fed back to the input layer, enabling the network to learn from the past data. That makes them applicable for sequence learning and prediction. However, common problems with RNNs are the vanishing and the exploding gradient problems. The vanishing gradient problem happens if the gradient becomes very small when it is backpropagated through time, making the network unable to learn from the older data. Long-short term memory (LSTM) models overcome this issue (however they can still suffer from exploding gradient problem) with gates that regulate the flow of information. An LSTM consists of cells. In the cell state information from previous cells is saved and considered in predictions (upper horizontal line in figure 2). With that the LSTM network can remember the relevant older data. The first yellow box from the left in the figure represents the **forgotten gate.** Based on the input (and the output of the previous cell) this gate decides which data the cell state should forget. The **input gate (**second and third yellow box) then updates the cell state and the **output gate** determines the output of the cell. Each cell gets its own input (an element from a sequence) and with it and with the information from the previous cells it makes predictions.

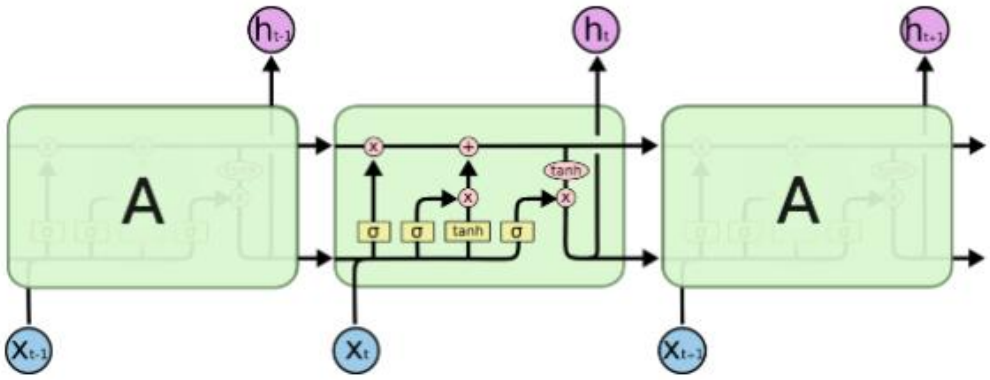

Figure 8: The repeating module in an LSTM contains four interacting layers (source: https://mc.ai/a-crash-course-in-sequential-data-prediction-using-rnn-and-lstm/)

Another example of RNNs are gate recurrent units (but are not considered into this report). A detailed explanation of RNNs and LSTMs can be found in various tutorials[3][4].

Data preparation

In case of classic machine learning models, the training data is a 2-dimensional array [number of samples, number of features] but for LSTM models the data needs to be 3-dimensional array [number of samples, series size (on how long sequence the LSTM is going to predict target value), number of features]. The target vector shape for LSTM depends on the problem. If we are predicting a single value one step in the future it would be a normal vector [number of samples]. If we are predicting multiple values one step in the future or (single value multiple steps in the future) it would be a 2-dimensional [number of samples, number of features to predict (or number of steps in the future)]. Lastly if we are predicting multiple values for multiple time steps into the future it would be a 3-dimensional array [number of samples, number of steps in the future, number of features to predict]. Below is the example of such data preparation for single value for multiple time steps.

The data is taken from Braila dataset, measuring data flow and aggregated water consumption (the dataset is further explained in chapter 8.2.2.1), but includes only a few features for simplicity. For this example, let's say that `series_size=3` (how far back are we looking) and `time_step=2` (how far in the future are we predicting).

timeStamp	analog2	hour	analog2_average_5	analog2_average_1
2018-12-05 07:15:00	0.821333	7	0.828000	0.820667
2018-12-05 07:30:00	0.820000	7	0.824933	0.821333
2018-12-05 07:45:00	0.827429	7	0.823467	0.820000
2018-12-05 08:00:00	0.829952	8	0.822286	0.827429
2018-12-05 08:15:00	0.830381	8	0.823876	0.829952
2018-12-05 08:30:00	0.832333	8	0.825819	0.830381
2018-12-05 08:45:00	0.828048	8	0.828019	0.832333
2018-12-05 09:00:00	0.830476	9	0.829629	0.828048
2018-12-05 09:15:00	0.834667	9	0.830238	0.830476
2018-12-05 09:30:00	0.833667	9	0.831181	0.834667

Table 1: Braila dataset features relevant for water consumption prediction

Below is the 3-dimensional array created from dataset above (analog2 being the target variable). Its construction can be viewed as a sliding window of size `series_size` over the feature columns, each time sliding one row lower.

```
[[[7.         0.828      0.82066667] [7.         0.82493333 0.82133333] [7.         0.82346667 0.82      ]]
[[7.         0.82493333 0.82133333] [7.         0.82346667 0.82      ] [8.         0.82228571 0.82742857]]
[[7.         0.82346667 0.82      ] [8.         0.82228571 0.82742857] [8.         0.82387619 0.82995238]]
[[8.         0.82228571 0.82742857][8.         0.82387619 0.82995238][8.         0.82581905 0.83038095]]
[[8.         0.82387619 0.82995238] [8.         0.82581905 0.83038095] [8.         0.82801905 0.83233333]]]
```

And here is the belonging target vector.

[3] https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[4] https://mc.ai/a-crash-course-in-sequential-data-prediction-using-rnn-and-lstm/

```
[[0.82742857 0.82995238] [0.82995238 0.83038095] [0.83038095 0.83233333] [0.83233333 0.82804762] [0.82804762
0.83047619]]
```

These two arrays are now ready to be fed into the LSTM.

Building a model

In Keras, a Sequential object, to which different layers are added is used for building a LSTM. There are several different designs for building a LSTM depending on how and which layers are stacked. The first layer added to the Sequential has an `input_shape` attribute no matter which type of layer that is. In our case the input shape should be (`series_size`, `number_of_features`).

The most basic design for a LSTM is a vanilla LSTM which only contains a single LSTM layer and a Dense output layer. The number of neurons in the output layer should be the same as the dimensionality of the output, so for normal 1-value, 1-step in the future prediction the number of neurons would be one but for multiple-value or multiple-time step predictions it would be more.

Stacked LSTM is very similar to vanilla LSTM but instead of one it has multiple LSTM layers.

On some time series it is useful to allow LSTM to learn it both forward and backward and for that a wrapper layer is used called Bidirectional (before adding a LSTM layer to Sequential it is put into a Bidirectional).

CNN-LSTM uses a CNN layer (wrapped in a `TimeDistributed` layer) to first interpret subsequences of input which are then provided as an input sequence to the LSTM.

Once the Sequential object is complete it needs to be compiled, specifying the optimizer and a loss function.

In the above process multiple variables can be optimized such as `series_size`, number of LSTM layers stacked, number of neurons in a layer, batch size, etc. When an LSTM is being trained the weights are updated after every batch of samples and the size of these batches is determined by the batch size parameter. Another important parameter is epochs which determines how many times all input vectors are used to update weights and biases. If it is too low the model will not learn from data all it can and if it is too high the model will overfit to training data, preforming worse on test data. To determine the ideal number of epochs validation set is used for calculating validation error after each epoch. As long as the validation error is dropping the model is learning and once it starts rising the model begins to overfit. This can most easily be recognized from a graph where the model is built and trained more than once so it does not depend only on a single run.

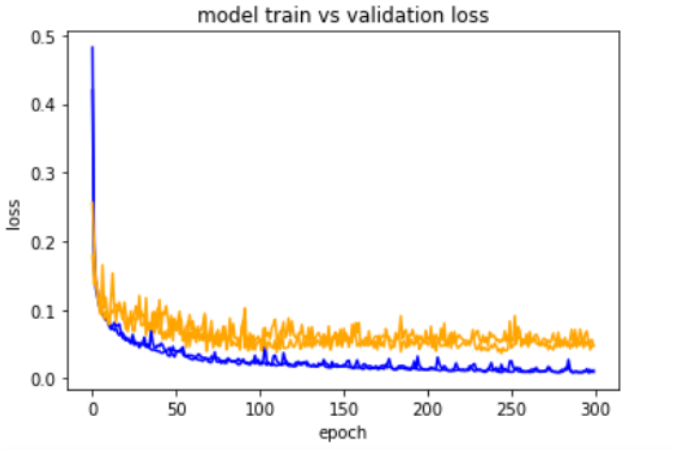

Figure 9: Model train vs. validation loss.

In the example above the model was trained twice with 300 epochs. The orange line represents validation loss and the blue represents train loss. The blue line is dropping during all of the 300 epochs however the orange one seems to stop dropping at about 100th epoch and maybe even slightly rises. Therefore, it is better to stop at about 100th epoch.

4.3 Incremental Learning Models

Note: This work has been prepared in [1] and is provided in the same form in this subsection.

Traditional statistical modeling techniques use batches of data points to learn. If the observed system is prone to concept drift [25], which means that the distribution of the target value is changing through time due to changes in the user behavior or in the environment, frequent re-learning of the models is needed, which is time-consuming (e.g. repeating of model learning step after each day or week). Incremental learning techniques [26] are able to update the existing models. The model, that has been taught on a set of learning examples, can be updated with the next one alone. The update itself is much cheaper than re-learning and often the incremental learning techniques are aware of the concept drift (e.g. when user behavior is changed to a previously unseen mode due to some external reason and this influences the underlying model), which means they are able to adapt to the change in the new systems behavior much faster.

Some of the tested methods were the streaming perceptron, Hoeffding trees [27] and Hoeffding adaptive trees - HAT [28]. Other methods include recursive linear regression, model trees like FIMT-DD [29], incrementally learned neural networks and incrementally learned SVMs based on stochastic gradient descent. Algorithms like decision trees can be used in ensembles (e.g. bagging), where each tree in the ensemble is fed with a subset of input data. For classification problems, which are rarer in the water management domain, there are many more methods available, however there are still not many effective implementations of the state-of-the-art methods.

4.4 Evolutionary Algorithms with Multiparent Recombination for Hyperparameter Tuning

Hyperparameter tuning is an optimization problem, traditionally solved with grid search or randomized search. The first one soon becomes useless with large number of hyperparameters since the time complexity rises exponentially and the second chooses samples totally at random.

Therefore, for NAIADES we will develop an approach using evolutionary algorithms. Multiparent recombination in EAs was already proven to be superior for solving some optimization problems (numerical optimization) and with this algorithm we will be testing how well does it perform at hyperparameter tuning.

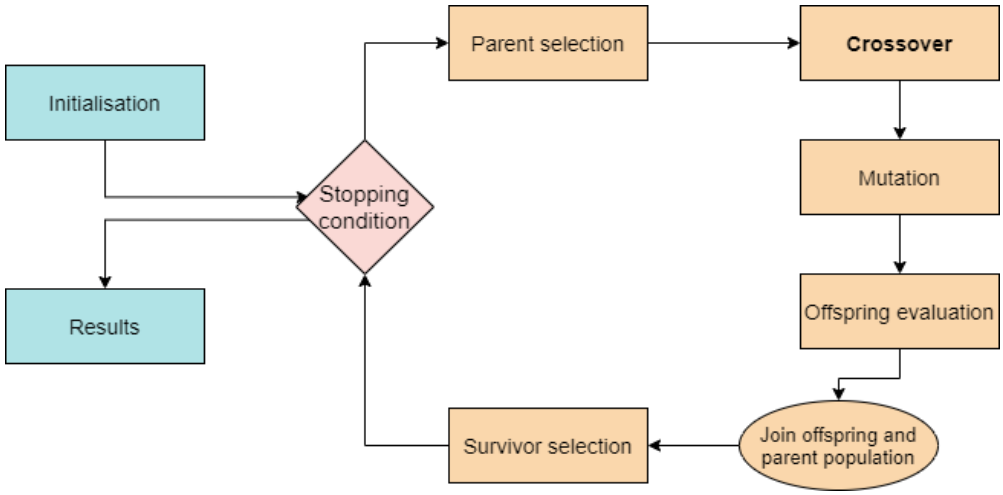

Figure 10: Schema of an evolutionary algorithm with multiparent recombination.

The overall structure of the algorithm is shown in Figure 10. Different parent selection, mutation and survivor selection procedures will be implemented but the focus here will be on the crossover, specifically if the algorithm could be improved by using multiparent crossover. There are many interesting algorithms for multiparent crossover (like simplex crossover, differential evolution, parent-centric crossover, etc.) but not many are suitable for mixed-problem spaces such as the space of possible hyperparameter values (some hyperparameters have discrete and some continuous values).

For comparison the n-point crossover schema will be implemented (shown Figure 11) where n marks the number of cutting points. Each time this algorithm is run it produces (at most) 2 offspring. One of the multiparent crossover algorithms implemented will be the diagonal crossover (shown in Figure 12) with a being the number of parents. It produces (at most) a offspring. This algorithm will be compared to n-point crossover, with $n = a-1$, so the number of cutting points (and therefore the disruption level) remains the same.

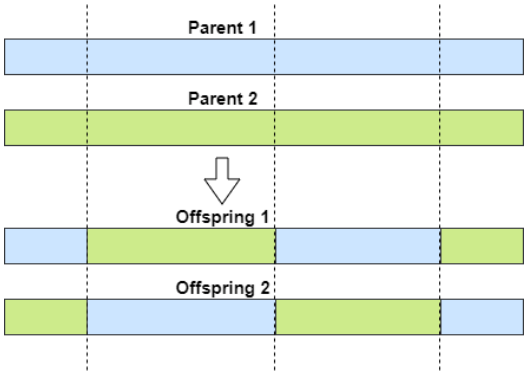

Figure 11: n-point crossover schema for comparison.

The algorithms will be compared based on how good solution it finds in a limited number of evaluations (solution quality) and how many evaluations does it take to find a solution that scores above some threshold (convergence rate). Since the EAs are stochastic, the experiment will be repeated multiple times and the average scores will be taken into consideration.

In addition, a variation of global recombination (a multiparent recombination algorithm) will be implemented and tested with the same criteria as the previous two. This one builds the offspring one gene at a time, for each choosing new parents (2 parents). If the gene has discrete values the value of the offspring is the value of the first parent with probability p or the value of the second parent with probability 1-p. Probability p is determined based on $s1$ (score of the first parent) and $s2$ (score of the second parent) by equation $p = s1/(s2+s1)$. The genes that have continuous values are calculated with the equation: $x = x2 + c(x1 - x2)$, where $c = p$ from the previous case.

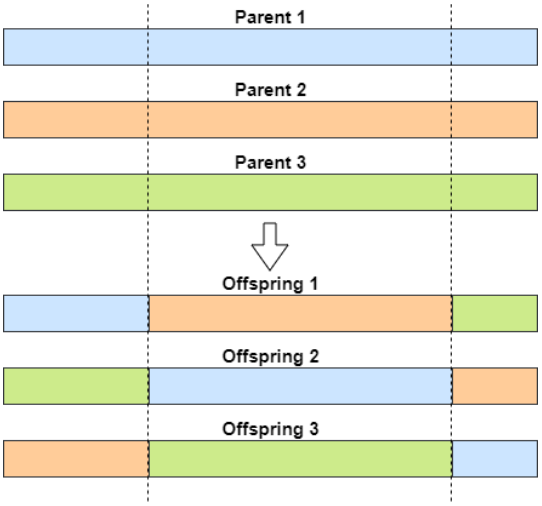

Figure 12: Diagonal crossover algorithm.

If the algorithm would be successful, it could be used for hyperparameter tuning for time series prediction models.

The algorithm has been implemented and will be tested on various dataset in the second phase of WP5.

4.5 Feature Engineering and Feature Selection

Note: This work has been prepared in [1] and is provided in similar form in this subsection. The work has been expanded with FASTENER feature selection.

Statistical models require not only the raw data, but also derived features, which reflect certain physical processes that are influencing the modeled phenomena (e. g. water consumption). When building additional features, the raw hourly weather forecast data can be used, consisting of precipitation probability, precipitation intensity, precipitation type, temperature, cloud cover, dew point, humidity, pressure and daytime. From these, daily averages, minima and maxima can be calculated, producing a plethora distinct features which were then analyzed with a correlation matrix (see Figure 13). In water consumption, time features are also important, such as hour of the day, day in a week, holiday status, day before or after holiday, working hours and similar. Most of the time-series that are directly influenced by human behavior are strongly dependent on all these features and those should be considered very carefully. Quite often accuracy scores can be boosted significantly (more than with other features like weather), when these features are included.

The following illustration is derived from groundwater level modeling scenario.

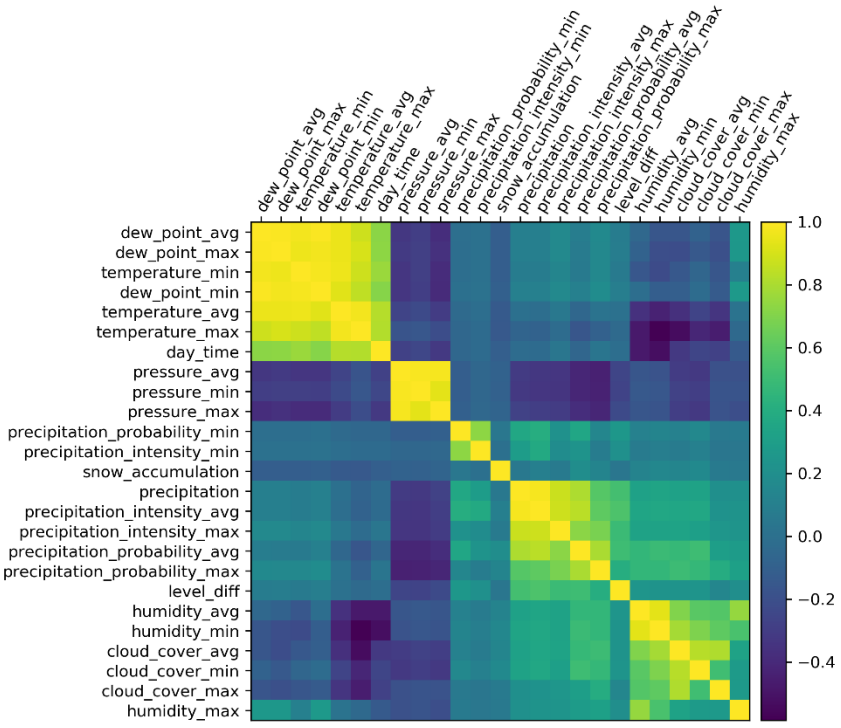

Figure 13: Correlation matrix of the target value (groundwater level difference) and 24 base features calculated from hourly weather forecasts.

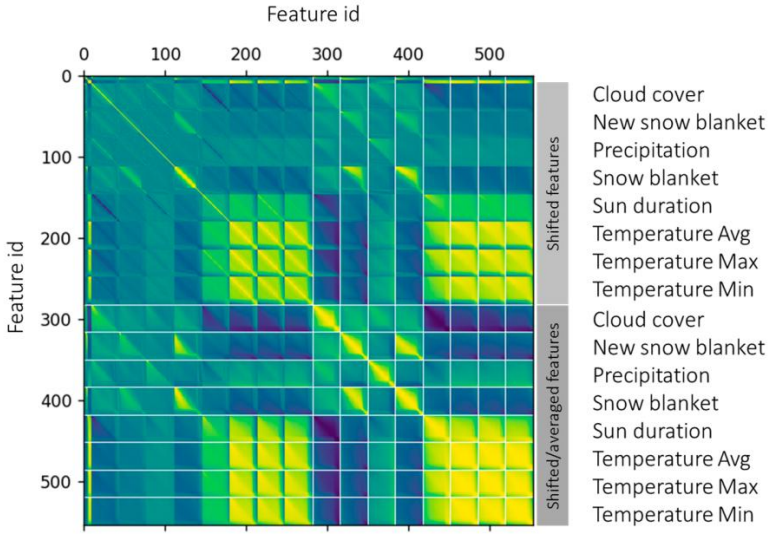

Figure 14: Correlation matrix of 544 features, including derived ones. The features are the original ones represented in the dataset (like weather data and current and historic groundwater levels) and derived ones (like averaged and shifted by different time intervals). Our platform enables on-line generation of all these features. Positive correlation is depicted in yellow and negative in dark blue shades.

A correlation matrix is a good tool when performing manual feature selection and can be used in two ways. Firstly, the correlations of particular attributes with the target variable can be read and the most correlated attributes can be selected to be used in the models. Secondly, it can be used for filtering of highly correlated attributes. Highly correlated attributes will not bring additional knowledge to the model and might worsen model accuracy.

For example, in this case, pressure turned out to be uncorrelated to the target value and has therefore been removed from the initial set of features. Dew point, on the other hand, has shown a very strong positive correlation to temperature and has therefore also been removed.

Remaining 18 initial features can be used to construct additional derivatives by introducing time delays (shifts) and averages over multiple past days. The idea behind this comes from the intuition that ground and surface water level responses to the weather changes have some hysteresis.

Since weather forecasts are always limited to finite prediction horizon the models would always be limited to the same prediction horizon in real-life circumstances. Therefore, it seemed appropriate to introduce the same limitation to the experiments. Specifically, in the majority of the tests, the prediction horizon was set to 3 days. With this limitation in place, it was possible to include another set of features derived from past water level values, which were constructed with a combination of delays and averages in a similar way as the aforementioned weather feature derivatives.

After the addition of all the feature derivatives, a total of 3890 features were available. With approximately 3000 samples of water level measurements in the datasets and the number of features in the same order of magnitude, further steps to reduce the size of the feature vectors were necessary to avoid overfitting. Feature selection can significantly reduce the number of features without sacrificing the expressivity of the model. In water related scenarios one often encounters problems, where the number of measurements of a time-series that is modeled is not very high (e.g. one measurement per day in a period of a couple of years). Many methods exist that provide efficient feature selection.

4.5.1 Top-ranked Features Approach to Feature Selection

The feature selection approach is based on the selection of a fixed number of top-ranked features, where *F-value* between features and target values is used as a ranking score. The whole procedure is done in three steps. Firstly, the correlation between each feature and target value is calculated for all training samples as defined in equation

$$c_i = \frac{\sum_{j=1}^{n}(x_{i,j} - x_{i,mean})(y_i - y_{mean})}{\sum_{j=1}^{n} x_{i,std}\, y_{std}}$$

where c_i is correlation of *i-th* feature to target value, *n* is the number of training samples, $x_{i,j}$ is the value of *i-th* feature in the *j-th* sample, $x_{i,mean}$ is the mean value of *i-th* feature over all samples as defined in equation

$$x_{i,mean} = \frac{\sum_{j=1}^{n} x_{i,j}}{n}$$

y_{mean} is the mean value of target value over all samples as defined in equation

$$y_{mean} = \frac{\sum_{j=1}^{n} y_j}{n}$$

$x_{i,std}$ is the standard deviation of *i-th* feature over all samples as defined in equation

$$x_{i,std} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{i,j} - x_{i,mean})^2}$$

and y_{std} is the standard deviation of target value over all samples as defined in equation

$$y_{std} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - y_{mean})^2}$$

Secondly, correlation is converted to *F score* as defined in equation

$$f_i = (n-2)\frac{c_i^2}{1-c_i^2}$$

where f_i is *F-value* of *i-th* feature. And finally features are sorted by their corresponding *F-values* and only first *k* with the highest score are included in the final selection. In this case, *k* was experimentally determined to be k=30, which yielded the best results on average.

This whole process of automatic feature extraction and selection was repeated for each dataset used in the experiments.

4.5.2 FASTENER – Genetic Approach to Feature Selection

The previous approach (and similar ones) relies on a sort of heuristics to find the optimal feature set. Based on correlation only we are not guaranteed to find the best possible dataset. Actually, if we consider the correlation matrix, it can happen that our model will only receive the useful information from on feature and the rest would only bring in additional noise. Also, the approach that was presented in the previous subsection considers similar shortcomings.

To be find the optimal set, one should try all the possible feature combinations. This however means, that many options should be considered. Based on the learning algorithm and dataset size, this process can be quite slow as we know that model training takes the most of the time. There are approximations that do this incrementally, for example "forward selection" which first finds the single best feature to model the target variable and then adds another one to it and repeats this process until the end. There is the opposite process of backward elimination. Still, although these searches introduce much less options, the search is still exhaustive and takes a lot of time.

Recently, we have proposed a FASTENER algorithm [16], which takes advantage of genetic algorithms and evolutionary approach. The algorithm has been proved state of the art in the Earth Observation domain, however, there are strong indications that it could give fast and superior results also in other domains.

FASTENER architecture is depicted in Figure 15. In the initialization phase individual feature importances are calculated based on entropy and Pareto front is created based on the initial random population (= feature sets). Pareto-front (=nearly optimal) solutions are then used in the mating and cross-over and mutation procedures in order to obtain better candidates for the next generation of solutions. If next generation candidates perform better than the old ones, they are introduced into population. Also, the population includes non-optimal solutions, which can still contribute positively in the evolutionary process. The front is, however, regularly purged in order to improve fitness of the candidates.

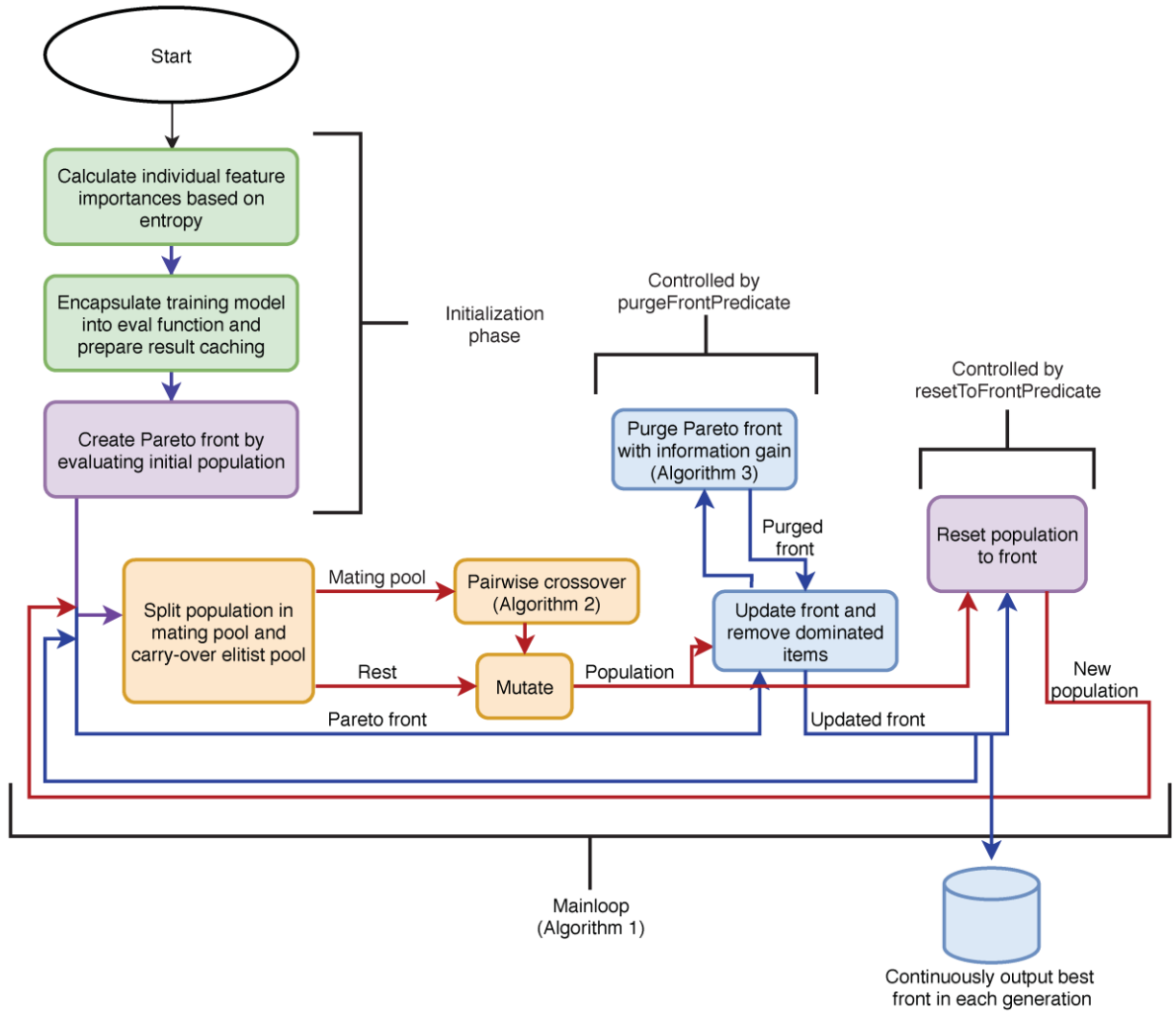

Figure 15: FASTENER architecture [16].

The final result of the FASTENER is a set of feature sets, from 2 to N features (where N is usually around 20). These feature sets can yield better modelling results than the original datasets with hundreds or thousands of features, because FASTENER eliminates the noisy features and maximizes the performance.

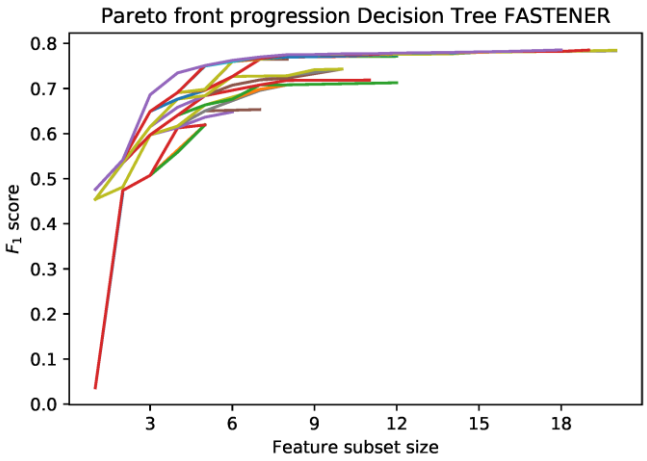

Figure 16: An example of Pareto front progression with FASTENER [16].

Figure 16 depicts progression of the model accuracy through different generations of the feature set candidates. The graph depicts F1 scores on the *y* axis and number of features on the *x* axis. The scores improve with time (each consecutive curve is above the previous one). Pareto front only presents the candidates that behave better than the candidates with less features. We can observe that the best candidates after some initial progressions are all below the size of 12. Also, we can observe that accuracy naturally is improved with a larger number of features, however – even though in this particular example hundreds of features have been used, already with ~7 features the algorithm achieves nearly optimal performance.

Also, with less features the data preparation and data acquisition steps take much less time and training/inference times are reduced. Within the scope of the NAIADES project FASTENER has been improved and documentation has been created. Everything is available in GitHub (https://github.com/JozefStefanInstitute/FASTENER).

4.6 Model Evaluation

Note: This work has been prepared in [1] and is provided in the similar form in this subsection. It has been extended with some further measure definitions and references.

When evaluating statistical models, the data set is usually split into 3 different parts. The first part (training set) is used to train the models, the second part (validation set) to fine-tune the hyper parameters and the last part (test set) for testing. This is known as the "hold-out" approach. This approach is dependent on a single split. This limit can be overcome with **cross-validation**. In this method, the data set is divided into *N* parts, each part being used once for testing, once for validation and $N-2$ times for training. With such an approach the random effects of arbitrary train-validation-test splitting are reduced. However, cross-validation assumes that the samples are independent, which is not the case for time series such as water levels due to the causality and autocorrelation of nearby samples. Therefore, an evaluation of the model is only possible on the basis of "future" observations. A ***k*-fold time series split methodology** was used, which is a variant of the *k*-fold cross-validation, but without including future data in the training set.

With this evaluation methodology several evaluation criteria are eligible for use when evaluating regression models. The most common are coefficient of determination (R^2), root mean squared error (RMSE) and mean absolute percentage error (MAPE)[30]. In the experiments no major differences between the different scores for the models and data were observed, however, in order to ensure comparability with other experiments and datasets a measure that is invariant to data offset and amplitude was chosen. R^2 preserves both (and has therefore been chosen), while RMSE is sensitive to amplitude and MAPE to the offset of data. R^2 has been chosen as the most suitable evaluation metrics for the experiments.

R^2 is defined as $\mathrm{R}^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}$, where y_i is the i-th target value, $\bar{y}$ is the average target value and f_i is the predicted value.

RMSE is defined with:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (\mathrm{a_i} - f_i)^2}{n}}$$

where a_i represents the actual observations of the model, f_i represents the estimated values of the model and n is the size of the sample. It ranges from 0 to ∞, where RMSE = 0 indicates a perfect fit.

MAPE is defined with:

$$\mathrm{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{a_i - f_i}{a_i} \right|$$

5 Stream Mining Pipeline for Speed Layer

Stream mining pipeline is described in depth in D5.1 (the reader should check additional chapters there), since the data preparation phases in anomaly detection and prediction are intertwined. The same components are used in both cases. Furthermore, models from demand prediction can be used as baseline estimates in the anomaly detection process.

Figure 17: Example of speed layer implementation and interconnection of particular components. Message distribution component is depicted in grey. illustrates the data flow within the envisioned platform. The basic building block of the platform is a messaging system, which we base on Apache Kafka. Apache Kafka is able to ingest messages from different producers and deliver them to different consumer. Within the IoT scenario such an approach provides some nice effects, mainly due to distribution of components within the system.

In the Figure 17, sensor data sources are depicted in blue, forecasts (e. g. weather) in green and additional metadata (static anthropogenic data) in yellow. Each data is ingested in a corresponding topic in Kafka. To these topics different data fusion components are listening. For example, **Fusion 3** component, depicted in magenta, is listening to sensor topic S3, forecast topic F1 and metadata topic M2. Data fusion merges all this data into viable feature vectors and broadcasts them to topic Fu3. There are 3 modeling components listening to this topic Fu3: Stream Model 1, Stream Model 2 and Batch Model 1. This means that feature vectors, generated with Fusion 3 will be involved in 3 predictions. Results of the predictions are then distributed further via Kafka topic P3. Any consumer listening to this topic, either directly connected to GUI or just an agent listening to the data and writing it to the database, can access this feed.

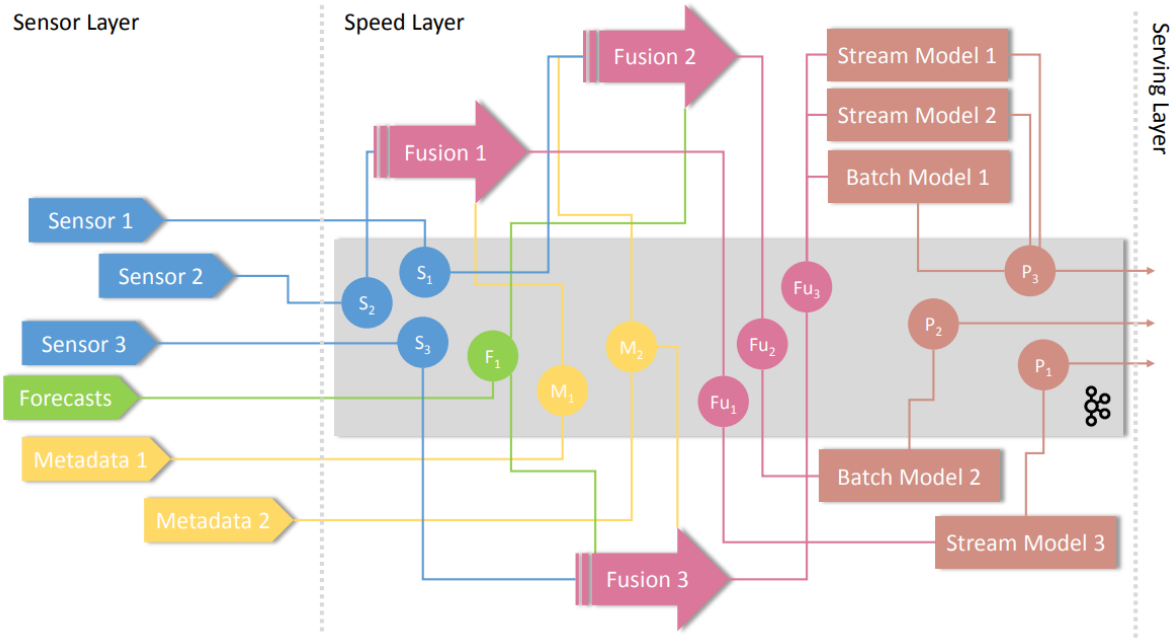

Figure 17: Example of speed layer implementation and interconnection of particular components. Message distribution component is depicted in grey.

Streaming data fusion component [24] component is the core of such a system and is technically particularly difficult to develop. Our prototype is based solely on streaming data, the prototype provides various in-memory buffers to compensate for different frequencies and delays of the incoming data. This means that the engine is fast (the bottle neck is in the modelling algorithms), however – if any instabilities happen in the system, this can mean loss of data.

Within the scope of the task we intend to provide an extension to this system, which would be based on dedicated streaming databases such as InfluxDB and would provide more reliable predictions.

5.1 Incremental Learning Module (within Data Fusion Framework)

One of the technical components that are needed to realize modeling demands within NAIADES project include incremental learning module. Incremental learning module is able to ingest the streaming real-world heterogeneous data sources and convert them into viable feature vectors. These vectors are then used for incremental training of the incremental models and for making the real-time predictions.

5.1.1 Challenges and Objectives of the Incremental Learning Component

The main challenges of this work are:

- Find proper and robust implementations of incremental learning methods
- If they do not exist, update a custom library (ml-rapids C++ library, exposed to Python will be extended within NAIADES; see Section 5.3)
- Explore competitiveness of incremental learning models in real-world scenarios (are they good enough?)

The goal of this component is to be seamlessly included into the data pipeline and will produce predictions on the output, which will be shared over Kafka.

5.1.2 Architecture

Architecture of the component is quite straight forward. The component is a simple single-box unit, where the inputs are in the form of feature vectors and outputs in the form of predictions. The unit only has to keep the model in place, take care of its initialization and updating.

5.1.3 Implementation

The module will be implemented within the data fusion component (see D5.1), because this way there is the least of overhead in the resources; simplified architecture is also less prone to errors. Implementation will be done using NodeJS, Qminer library and ml-rapids library.

5.2 External Batch Modeling Component

5.2.1 Challenges and Objectives of the Modeling Component

Most of the modelling in academia (but often also in industry) is happening in the ecosystem which is most popular among machine learning engineers. And without a doubt this system includes Python with additional libraries like Pandas (for data wrangling), scikit-learn (for basic modeling),

5.2.2 Architecture

Architecture is similar to the one of the previous components, except that with a batch modeling component it is important also to take care of re-training. For this, a special local storage for the feature vectors will be realized and an interface provided for triggering the re-training.

Additional plan is to prepare a GUI for managing the ecosystem of data fusion and modeling components. The ecosystem would be able to distribute a plethora of any components involved in modeling and connect them into an efficient distributed processing network.

5.2.3 Implementation

Due to availability of modeling libraries, this component will be developed in Python.

5.3 Incremental Learning Library (ml-rapids)

Incremental learning is a promising field of machine learning. Many academic endeavors have tried to extend the efficiency of the methods. The simplest modeling approaches include stochastic gradient descent-based methods, like recursive linear regression and perceptron. A more complex approach would include an implementation of incremental decision trees (like Hoeffding trees or FIMT-DD). The next level would include ensemble learners, such as streaming random forest of Hoeffding trees or bagging of Hoeffding trees. Theoretically, stochastic gradient descent could also be used for learning linear SVM and neural networks.

5.3.1 Challenge and Objectives of Incremental Learning Library

There are several libraries, that already include incremental learning methods. The most well-known are MOA and SAMOA. Those are, however, written in Java and mostly support academic community and laboratory experiments. There is also a `scikit-multiflow` library written in Python. It is however in its early stage of development and the implementations are unstable and rather slow.

No good alternative is available for development in NodeJS.

5.3.2 Current State and Development Plans

We have decided to build upon a port of MOA into C++ (streamDM) library, which we have extended for the classification purposes within the H2020 PerceptiveSentinel project. The library includes the fastest implementation of the Hoeffding trees and is exposed to Python.

In NAIADES we plan to extend the library with regression algorithms (stochastic gradient descent for linear regression, perceptron, linear SVM and simple neural networks), regression Hoeffding trees and its derivatives (bagging and streaming random forest).

We plan to expose the library via NodeJS using SWIG.

The library is available in GitHub (https://github.com/JozefStefanInstitute/ml-rapids).

6 API Definitions

API definitions are presented in D5.1; only prediction definitions are given here. The reader should refer to NAIADES D5.1, Chapter 6, for additional information regarding ingestion of data and intermediate APIs.

6.1 North-Bound Interface

6.1.1 Predictions from modeling component (batch)

Kafka topic: `predictions_[nodeid]`

Predictions are generated as scheduled (after the arrival of last relevant data) and then pushed to Kafka with the following format.

Field name	Description	Data type
sensor_id	Unique sensor identification	string
value	Forecasted value of the sensor	float
stampm	Measurement time stamp in UNIX date format	long
horizon	Prediction horizon (in hours)	integer
predictability	R2 evaluation measure (multiplied by factor 100), calculated on last 20% of the training dataset	float
rmse	RMSE for a particular model for last n measurements (configurable by evaluation_periode in config.json); optional.	float
mae	MAE for a particular model for last n measurements (configurable by evaluation_periode in config.json)	float
mape	MAPE for a particular model for last n measurements (configurable by evaluation_periode in config.json); optional.	float
mse	MSE for a particular model for last n measurements (configurable by evaluation_periode in config.json); optional.	float
r2	R2 for a particular model for last n measurements (configurable by evaluation_periode in config.json); optional.	float

Table 2: NAIADES forecast prediction NBI schema.

Example of sensor JSON for (batch) modeling forecast prediction:

```
{
    "stampm":1479049200000,
    "value": 821,
    "sensor_id": "N1",
    "horizon": 24,
    "predictability": 0.82,
    "rmse": 425.7499286389971,
    "mae": 358.06944444444446,
    "mape": 60.692180851294097,
    "mse": 181263.00173611112,
    "r2": 0.89
}
```

6.1.2 Predictions from data fusion component (incremental learning)

Kafka topic: `predictions_[nodeid]`

Predictions are generated as scheduled (after the arrival of last relevant data) and then pushed to Kafka with the following format.

Field name	Description	Data type
sensor_id	Unique sensor identification	string
value	Forecasted value of the sensor	float
stampm	Measurement time stamp in UNIX date format	long
method	Method, used for prediction (EMA, Structured EMA or RecLinReg)	string
horizon	Prediction horizon (in hours)	integer

Table 3: IAPE forecast prediction NBI schema.

Example JSON of data fusion incremental forecast prediction:

```
{
    "stampm":1479049200000,
    "value": 821,
    "sensor_id": "N1",
    "horizon": 24,
    "method": "EMA",
}
```

6.1.3 Predictions from modeling component (long-term / batch)

Kafka topic: `predictions_[nodeid]`

Predictions shall be generated upon the user's request and then pushed to Kafka with the following format.

Field name	Description	Data type
location	Location for which the forecast is made (Alicante or Braila)	string
timestamp	Time of the prediction generation in UNIX date format	long
monthlyValues	Array of predicted monthly values	array
biweeklyValues	Array of predicted biweekly values	array
month	Month for which the prediction is made (January, February, etc.)	string
fortnight	Fortnight for which the prediction is made (January 1-15, January 16-31, etc.)	string
prediction	Prediction value for water demand	float
rmse	RMSE for a particular prediction.	float
mape	MAPE for a particular prediction.	float
horizon	Prediction horizon (in months)	integer

Table 4: NAIADES long term forecast schema

Example JSON of monthly forecast:

```
{
	"timestamp":1479049200000,
	"monthlyValues":[
	{
		“month”:”January”,
		"prediction":345,
		"rmse": 425.7499286389971,
		“mape": 60.692180851294097
	},
	{
		“month”:”February”,
		"prediction":255,
		"rmse": 425.7499286389971,
		“mape": 60.692180851294097
	}],
	"location": "Alicante",
	“horizon": 2,
}
```


Example JSON of biweekly forecast:


```
{
	"timestamp":1479049200000,
	"biweeklyValues":[
	{
		"fortnight":"January 1-15",
		"prediction":375,
		"rmse": 425.7499286389971,
		“mape": 60.692180851294097
	},
	{
		"fortnight": "January 16-31",
		"prediction":315,
		"rmse": 425.7499286389971,
		“mape": 60.692180851294097
	},
	{
		"fortnight": "February 1-14",
		"prediction":250,
		"rmse": 425.7499286389971,
		“mape": 60.692180851294097
	},
	{
		"fortnight":"February 15-28",
		"prediction":260,
		"rmse": 425.7499286389971,
		“mape": 60.692180851294097
	}],
	"location": "Alicante",
	“horizon": 2,
}
```

7 Evaluation of Modeling Methods on Groundwater Data for Ljubljana Aquifer

As data from the use cases was not available from the beginning of the project and since the available data is quite sparse (only a few available sensors), the initial evaluation of the used methods has been developed on an open dataset of groundwater levels in the Ljubljana aquifer. The dynamics of this system is not dependent on the human behavior, which is a big difference with NAIADES use cases, but we have extensive experience with modeling with anthropomorphic variables from other domains (smart buildings, energy consumption, etc.).

The goal of these experiments has been twofold: (1) test efficacy of the statistical modeling methods in water scenarios (Are they good enough in general? What approaches work best?) and (2) how do incremental learning techniques perform in comparison to batch learning (Are incremental learning techniques good enough?).

Note: The findings have been published in [1] and the following subsections are mostly a direct citation from this paper. We have extended this work with deep learning approach (in the end of this section).

7.1 Data

Groundwater and surface water data (see Table 5) have been acquired from an online repository[5] at the Slovenian environment agency (ARSO). Weather data has been retrieved from DarkSky[6] web service and ARSO historical weather data repository[7]. Sensor data has been thoroughly inspected and only the sensors with data in the period from 2010 to (including) 2017 were selected, for which accurate weather could be retrieved as well. Weather data related to underground water modeling has been retrieved for numerous spatial points in the aquifer, based on the locations of the target sensors, while the weather data for surface water modeling has been retrieved from a single location in the vicinity of the water spring for each selected watercourse.

Id	Name	Selected sensors	Availability	Frequency
1	Groundwater levels	2	2010-2017	1/day
2	Surface water levels	22	2010-2017	1/day

Table 5: Experimental datasets include 24 time series. 2 for groundwater levels in Ljubljana region and 22 for surface waters in Slovenia.

Although available datasets are larger, only the sensors with clean data that is available throughout the selected period have been chosen for the experiments. Sensors with missing data and with missing or corrupted contextual data (weather) were excluded from the experiments.

The used groundwater levels dataset has been studied extensively (Kenda, 2018b). The groundwater levels in the aquifer were modeled with linear regression of the values of nearby sensors. The study exposed that majority of the sensors are highly correlated and could be modeled with extremely high accuracy ($R^2 > 0.995$), while the minority of much less correlated sensors could still be modeled with $R^2 > 0.83$. Due to transitive properties of the operations the presented methodology could be extended to other sensors and comparable results could be expected.

[5] http://vode.arso.gov.si/hidarhiv/pov_arhiv_tab.php
[6] https://darksky.net/
[7] http://meteo.arso.gov.si/met/sl/archive/

7.2 Experiments with Groundwater (classical methods)

In this subsection, we provide the following results: illustrative prediction results for water level differences and for water levels with multiple prediction horizons, comparison of the accuracy of different methods, and qualitative results on the importance of model features that can be further analyzed by domain experts.

The experiments were conducted with a set of 11 regression and 10 classification modeling methods. When using multi-class classification methods, the discretization of the target space (daily water level differences) into 8 separate classes was used. The feature space normalization was used where necessary (for support vector machines, multi-layer perceptron neural network - MLP, k-nearest neighbors and perceptron). The implementation of the statistical models from scikit-learn (batch learning) and scikit-multiflow (incremental learning) libraries for Python was used.

Illustrative results for the prediction of level differences are shown in Figure 18, cumulative results for water levels are shown in Figure 19. The overall experimental results are listed in Table 5 and depicted in Figure 20.

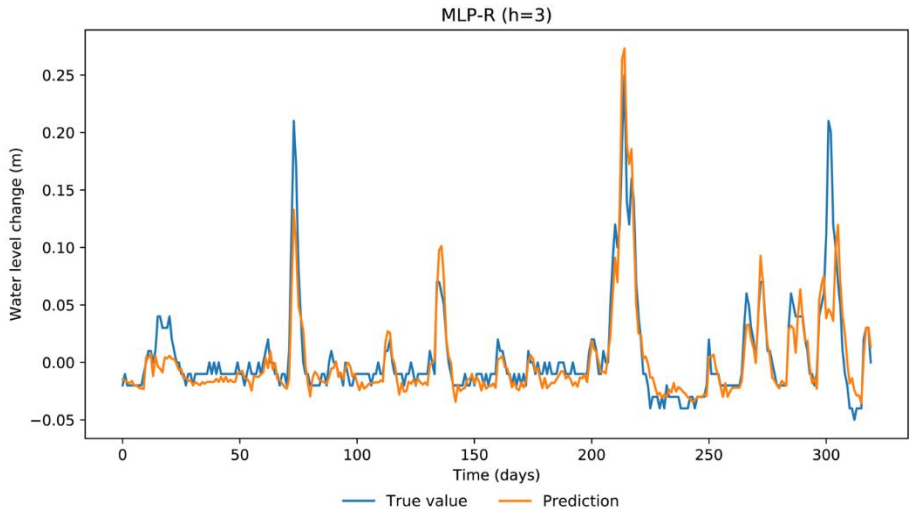

Figure 18: Prediction of groundwater level difference (change) for MLP (multi-layer perceptron neural network) Regressor for 3-day prediction horizon.

Figure 18 depicts the actual results of the prediction experiments. Level differences are calculated for 3 days in advance. The statistical model can predict major changes in groundwater levels with fairly good accuracy (in terms of start, duration and amplitude). The modelling results are converted into water level predictions in Figure 19.

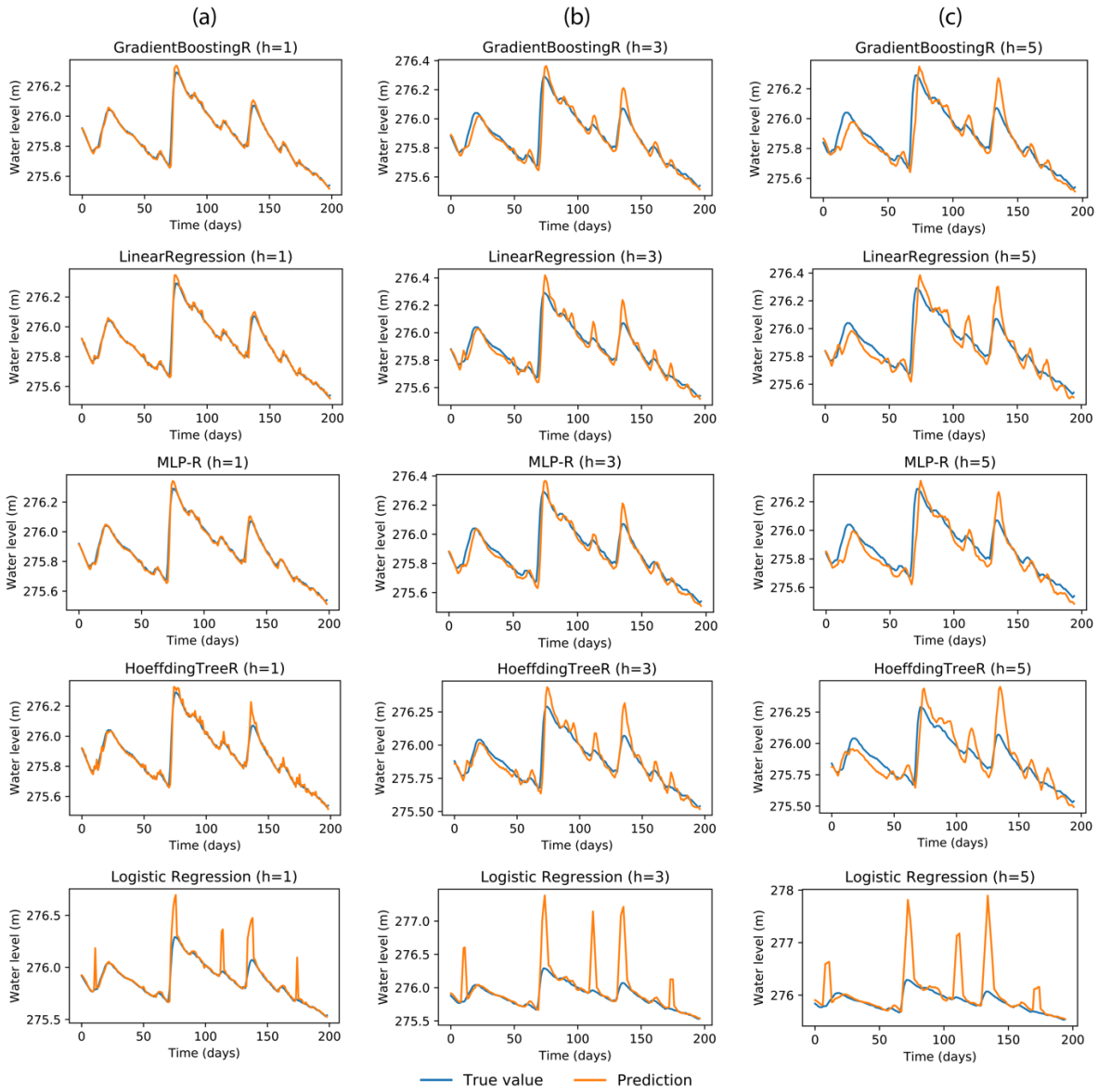

Figure 19: Illustrative results of different prediction models for groundwater levels. Different algorithms are depicted in rows, different prediction horizons: (a) 1 day ahead, (b) 3 days ahead and (c) 5 days ahead are shown in columns.

Figure 19 illustrates the characteristics of different types of statistical models for groundwater level prediction. The best regression models, the best streaming regression model and the most illustrative classification-based models are included in the figure. The majority of models overshoots larger level changes. This means that in historical data in some cases includes bigger level changes under similar conditions. This could, for example, indicate higher water withdrawal or a change in the dynamics of the groundwater system. Prediction accuracy decreases with the prediction horizon and that the models sometimes miss the beginning of a change (with longer prediction horizons). The only model that correctly records the beginning of the water level change properly even for a 5-day prediction horizon is the classification model. This means that although the accuracy of the model is poor, the classification could be a good choice for estimating the start of level changes. In addition, the reason for poor classification results can be seen from the figure. During the discretization, the rarely occurring high values of the level changes were grouped into a single bin (which produces a fixed amplitude), although these values are distributed over a bigger interval. For extreme values, finer discretization would lead to better results.

	1 day ahead		2 days ahead		3 days ahead		4 days ahead		5 days ahead			
Method	R^2	σ	R^2	σ	R^2	σ	R^2	σ	R^2	σ	t_t [ms]	t_p[ms]
LinearRegression	0.834	0.037	0.708	0.063	0.661	0.066	0.638	0.063	**0.641**	0.07	1.9	0.2
DecisionTreeR	0.677	0.146	0.61	0.111	0.545	0.093	0.541	0.061	0.543	0.059	3.3	0.1
RandomForestR	**0.842**	0.031	**0.726**	0.108	**0.669**	0.08	0.644	0.067	0.600	0.054	1047	8.5
GradientBoostingR	**0.849**	0.037	**0.775**	0.052	**0.732**	0.071	**0.690**	0.081	**0.655**	0.09	322.6	0.5
PLSRegression	0.726	0.052	0.65	0.073	0.639	0.076	0.639	0.076	0.639	0.077	2.9	<0.1
ExtraTreeR	0.677	0.146	0.61	0.111	0.545	0.093	0.541	0.061	0.543	0.059	3.3	<0.1
SVR	-0.137	0.41	-0.16	0.356	-0.113	0.325	-0.069	0.317	-0.086	0.324	2.5	0.4
MLP-R	0.825	0.045	0.691	0.084	0.659	0.09	**0.646**	0.092	**0.641**	0.082	210.2	1.3
KNeighborsR	0.747	0.053	0.661	0.089	0.631	0.102	0.618	0.112	0.610	0.114	5.9	8.0
HoeffdingTreeR	0.495	0.164	0.513	0.127	0.518	0.144	0.493	0.139	0.482	0.13	3347.1	28.1
HAT-R	0.506	0.176	0.513	0.127	0.518	0.144	0.493	0.139	0.482	0.13	3722.2	28.5
LogisticRegression	0.485	0.153	0.408	0.131	0.395	0.146	0.376	0.179	0.370	0.16	83.3	0.2
DecisionTreeC	0.506	0.141	0.395	0.116	0.342	0.183	0.365	0.203	0.377	0.183	4.9	<0.1
ExtraTreeC	0.306	0.09	0.336	0.113	0.24	0.162	0.332	0.058	0.256	0.258	2.4	0.1
RandomForestC	0.554	0.108	0.489	0.178	0.481	0.19	0.498	0.176	0.470	0.186	279.8	9.8
SVC	0.522	0.088	0.413	0.131	0.375	0.158	0.391	0.193	0.400	0.191	135.4	16.9
KNeighborsC	0.530	0.071	0.378	0.143	0.387	0.201	0.357	0.157	0.353	0.16	7.8	15.8
Perceptron	0.435	0.206	0.102	0.388	-0.007	0.788	0.325	0.291	0.266	0.268	10.5	0.2
GaussianNB	0.472	0.112	0.378	0.123	0.374	0.138	0.362	0.144	0.358	0.148	1.2	0.3
HoeffdingTreeC	0.472	0.112	0.378	0.123	0.374	0.138	0.362	0.144	0.358	0.148	1054.2	119.5
HAT-C	0.453	0.108	0.371	0.14	0.364	0.153	0.346	0.147	0.353	0.148	1912.2	120.4

Table 6: Averaged modeling results for groundwater levels for 5 different prediction horizons (from 1 to 5 days ahead). Best results by prediction horizon are bolded, best classification-based results are underlined.

Note: R^2 is calculated for level differences, which is a rapidly changing time series. This means that the R^2 for the actual water level is much higher. For example, the highest coefficient of determination for 5-day prediction horizon was 0.655 for level differences, while for actual groundwater levels it was above 0.95.

The modeling methods perform as expected. Traditional workhorses perform best (with R^2 scores between 0.6 and 0.85), incremental learning methods and classifiers yield substantially worse results. Gradient Boosting is significantly better than competing methods. Linear regression is also a good choice in this setup. However, it is important to note that extensive feature engineering has helped it catch the latent dynamics of the aquifer, while Gradient Boosting could perform reasonably well even without so many features. The discretization of daily level differences in classes and the use of multi-class classification did not perform well. During discretization some information is lost, which is reflected in the results.

Hoeffding Tree regressor performed best among incremental learning methods. It maintains its performance even with longer prediction horizons and is more competitive in scenarios with a prediction horizon of more than 2 days. The use of incremental learners is most effective in scenarios where the

distribution of target values changes over time, which is not the case for groundwater and surface water levels in Slovenia between 2010 and 2017. The methods could be much more effective in modeling the behavior of water consumers.

The main advantage of the usage of incremental learning methods is that they do not have to re-learn from all the data in the training phase. They can only be updated with the latest value and are thus computationally much more efficient. This could be taken advantage of in scenarios where many sensors with fast updates are used (e.g. modeling consumers' behavior in a large city). Batch models should be retrained regularly. The training (t_l) and prediction (t_p) times listed in

Table 6 show that linear regression could be the most effective method in practice, since its training and prediction times are among the smallest, while the competitive methods like random forest, gradient boosting and MLP require significantly more time to (re-)train.

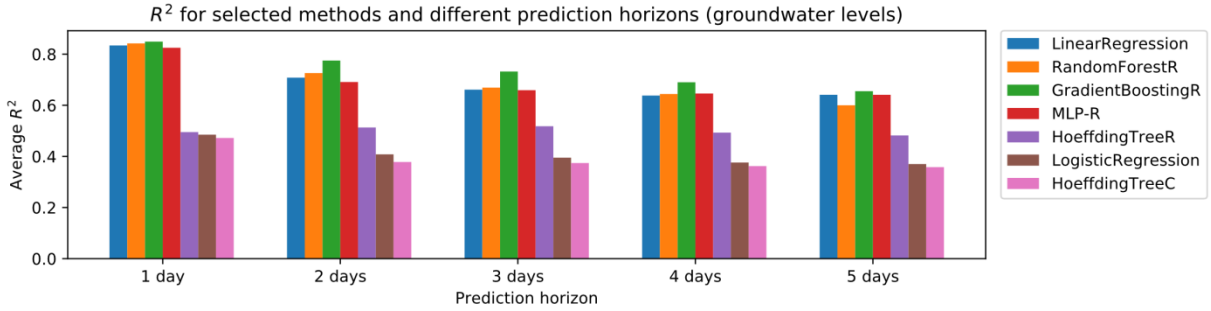

Figure 20: R^2 of selected statistical models for different prediction horizons (groundwater levels).

R^2 scores are used as a comparative measure to select the best possible methodology. Comparison of the R^2 scores with other studies is only possible on the illustrative level, since different datasets are based on the aquifers with different internal processes. Currently, no standardised dataset to test different approaches exists for groundwater (and surface water) levels. Illustrative comparison to the recent state of the art (Chen et al., 2020) shows that the presented methodology could achieve superior results (R^2 scores for 1 day ahead are by 0.1 larger than the compared results in a "now-casting" scenario). This could be attributed to extensive feature engineering, higher number of tested methods and intelligent feature selection. The study (Chen et al., 2020) also shows that data-driven models give superior results to the traditional process-based models by a high margin (R^2 scores of data-driven models are higher by approximately 0.2).

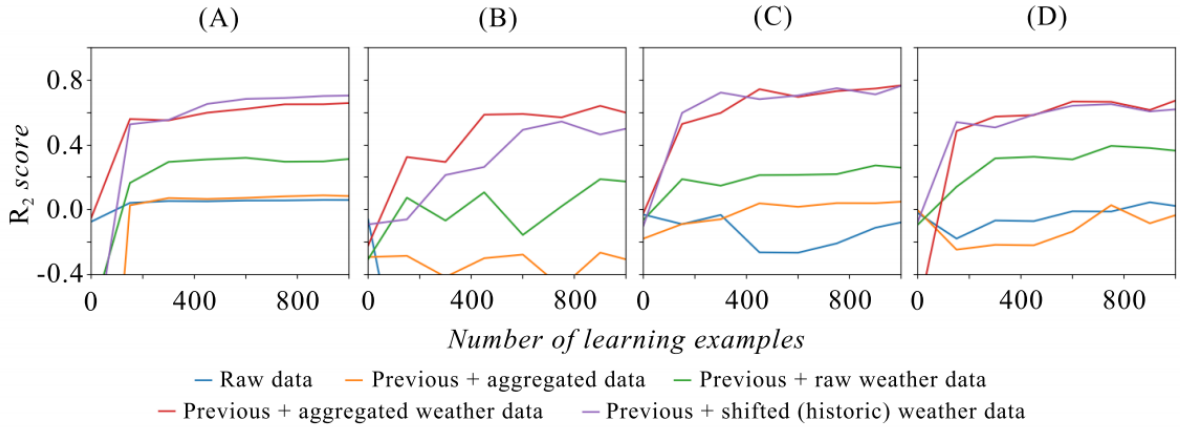

Figure 21: Learning curves of plain and enriched datasets with multiple learning methods: (a) linear regression, (b) decision trees, (c) gradient boosting regression, (d) random forest regression. The models are estimating level rise/fall on a particular day for groundwater.

Progression of the R2 scores (learning curves) with different models according to the number of learned examples is depicted in Figure 21. We can observe that with raw data only the models can not calculate anything that would make much sense (R2 scores are around 0 or less). Also, with aggregated historic data, they have quite some problems. Only when we introduce weather data the accuracy of the models improves, however – really good estimates can only be achieved with aggregated and shifted historic weather data and weather forecasts. This is important to consider, because it illustrates that modelling is strongly dependent on the generated features.

Most machine learning algorithms provide a kind of score on the importance of a feature in the statistical models. In linear regression these are coefficients, in ensemble tree-based methods these are importance weights. An example importance weights for a 3-day prediction horizon for gradient boosting is shown in Figure 22. The groundwater level change is influenced by the current trend (last level difference) and different values related to precipitation and its history. Among other weather phenomena, none was selected using the automatic method. The current average precipitation values are the most important ones. Some shifted features also play an important role.

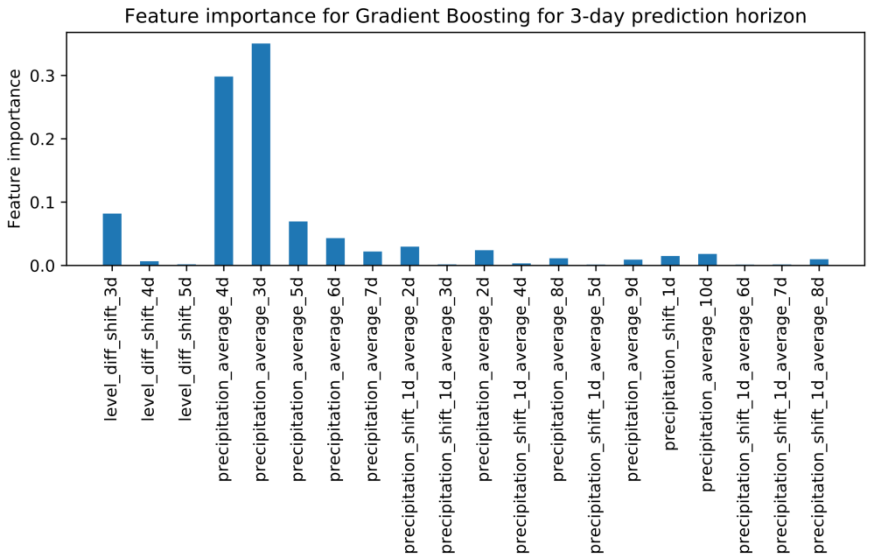

Figure 22: Feature importance scores (relative) as given by gradient boosting regression model for 3-day prediction horizon for ground water level modelling.

This analysis shows that the automatic feature selection algorithm overlooked some of the seasonally important features such as data related to snow/snow melting, cloud cover and similar, which is a consequence of the correlation-based approach. These features could significantly improve the accuracy of the algorithms in the respective season.

Finally, it is worth mentioning that an automatic methodology to produce the reported statistical models was developed. The performance of the automatic methodology could be improved by using a better feature selection algorithm (some have been tested) that would select most informative features based on their modelling performance. A genetic search algorithm across the modelling feature space could be the most efficient. Of course, modelling the water level in a particular well can benefit significantly from the input of the domain expert (what features to use and what additional data is required).

7.3 Experiments with surface water (Deep Learning)

Some experiments have been made also with the usage of deep learning approaches. Surface water of Ljubljana aquifer has been extensively tested. The report is available in [1].

All experiments were performed on historical ARSO surface water level data for sensor 8565. A 10-fold time series split was used for data validation with first 3 splits discarded each time for their small size. The model was trained to predict the water level change for the next day (prediction horizon = 1) based on weather data and water level changes for the past 6 days.

The following Keras model was used in the experiments:

```
model = Sequential()
input_shape = (self.look_back, self.data[0][0][0].shape[2])
# shape: (look_back, number_of_features)
conv_layer = Conv1D(filters=self.CNN_FILTERS, kernel_size=self.CNN_KERNEL,
  input_shape=input_shape)
model.add(conv_layer)
lstm_layer = tf.keras.layers.LSTM(activation="tanh", recurrent_activation="sigmoid",
  units=20, dropout=self.LSTM_DROPOUT, recurrent_dropout=0, unroll=False,
  use_bias=True,
input_shape=(conv_layer.output_shape))
model.add(lstm_layer)
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer=Adam(), metrics=[])
```

Some of the underlined hyperparameters were modified and the models repeatedly evaluated to estimate their effect on the model accuracy. The models were trained with batch size of 1 as better generalization was observed with smaller batch sizes and smaller numbers of full passes (epochs).

Filters \ Kernel size =>	2	3	4
1	0.39 / 0.35	0.39 / 0.39	0.37 / 0.35
4	0.42 / 0.43	0.41 / 0.43	
8	0.44 / 0.45	0.43 / 0.43	
16	0.43 / 0.45		0.36 / 0.44
64	0.39 / 0.43		0.38 / 0.38

Table 7: Averaged R-squared scores for models compiled with given number of filters and kernel size. The first score was obtained with dropout=0, the second score with dropout=0.1. Not all scores shown. Batch size = 1, 7 epochs.

The highlighted result (filters=8, dropout=0.1, kernel_size=2) was used in the following experiments. In all experiments, the average R^2 score of the training data was ~20% or more above the score of the testing data. This may be explained by the small number of data samples (less than 5000).

Another experiment was performed by injecting zero-centred Gaussian noise with standard deviation 0.1 into the training data, and artificially generating three times the original amount of data:

```
model.add(tf.keras.layers.GaussianNoise(stddev=0.1, input_shape=input_shape))
```

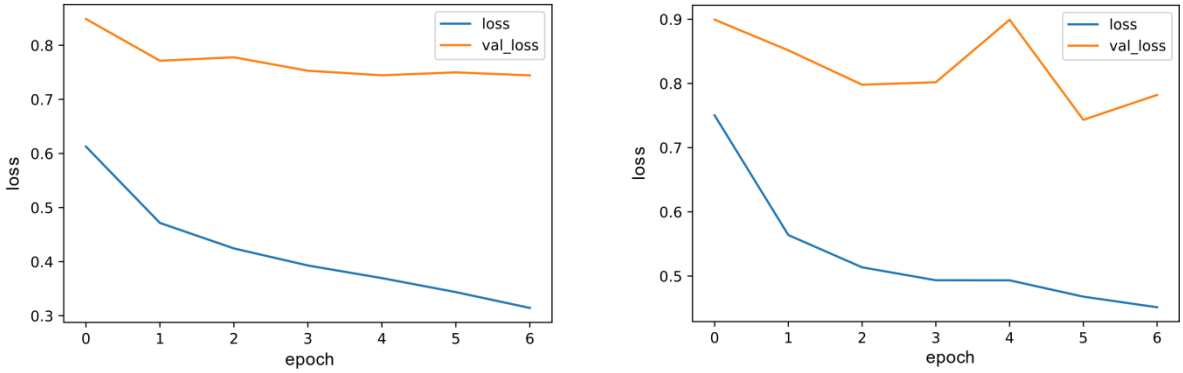

Figure 23: Training and validation loss vs. epoch plotted for fitting the model with data with Gaussian noise (left) and raw data (right). An 80% train – 20% validation split is used with these two experiments

From **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** it is seen that validation loss convergence is easier to detect when using more data with inserted Gaussian noise.

Models trained on both sets of data were evaluated with a 10-fold time series split at 5 and 7 epochs.

Epochs	**Using Gaussian noise, σ=0.1, 3 times more training data**	**No Gaussian noise**
5	R^2(test) = 0.43 / R^2(train) = 0.67	R^2(test) = 0.40 / R^2(train) = 0.51
7	R^2(test) = 0.39 / R^2(train) = 0.77	R^2(test) = 0.44 / R^2(train) = 0.56

Table 8: R^2 scores for models using Gaussian noise and not using Gaussian noise, at 5 and 7 epochs.

The performance of the model fitted on the extended dataset with gaussian noise at 5 epochs is similar to the performance of the original model at 7 epochs. However, when more data with added noise is used, detecting convergence of the validation loss is significantly easier.

A linear regression model fitted with selected k-best features (k=15) and evaluated using the same validation scores resulted in R^2 = 0.55. Because of the heavy computational cost of neural networks and their smaller performance linear regression or other models should be preferred for our use case.

8 Use-Case Definitions

8.1 Carouge

8.1.1 Objectives

The Carouge use case is divided into several subcases, of which UCC2 (Use Case Carouge 2) subcase is identified as a case that matches objectives of task 5.3.

From a broader perspective, the idea of UCC2 is to investigate and identify a more optimal scenarios for the process of public gardens watering. Due to fact, that majority of cities are taking care of their public gardens, the UCC2 is highly relevant for a number of cities in Europe and broader.

A quest for a more optimal scenarios is not only focused on water savings, but also on savings in terms of employee's time that is spent for gardens watering.

The concrete objective of the Carouge UCC2 are:

- Identifying watering deadlines for observed plants.
- Identifying amounts of water needed for a given plant watering.
- Identifying process optimization in terms of water savings as well as employee's time savings.

8.1.2 Available Data

For the Carouge use case we have failed to obtain any relevant data that would enable us to run the experiments. The sensors were installed late (just before M18) and we have not been able to obtain any data substitutes. Fortunately, the sensors now are installed and data have started to aggregate, which is a base premise for our work.

8.2 Alicante and Braila

The reports for Alicante and Braila are merged into one single report as both subcases, UCA1 (Use Case Alicante 1) and UCB1 (Use Case Braila 1), are identical.

8.2.1 General Objectives

From a broader perspective, the idea of UCA1 and UCB1 is to predict water consumption of a given city district. The subcases are divided into subtasks:

- Short-term prediction, which is predicting water consumption up to 10 days in advance;
- Long-term prediction, which is predicting water consumption from two weeks and up in advance.

8.2.2 Short-term Forecasting Objectives

The importance (and hence the objectives) of the water prediction task, can be explained through the process of how water is supplied to Alicante residents.

Alicante, due to water scarcity in the region, is unable to abstract enough water for its residents. The problem has been resolved by building long pipes (50 kilometers plus), through which water is supplied from other Spanish regions to the city of Alicante. Not only water is abstracted from very deep wells (up to 400m in depth), which requires lots of energy, abstracted water needs to be transported to the city as well. Alicante, in order to reduce water supply costs, executes abstraction and transportation by night, when energy has a low tariff. The water is then stored in a city water tanks, ready to be supplied to the city residents. Which bring us to the problem Alicante has:

- If water utility fails to store enough water in water tanks, the utility will run out of water still during a day time, which will force the company to start water abstraction within a high energy tariff, which results in high water abstraction costs.
- If water utility stores too much water in water tanks, water gets eventually contaminated and utility needs to dispose water, which, again, leads to the unnecessary costs.

The task objective is, by prediction water consumption of a given district, to minimize water utility expenses.

8.2.2.1 Available data

In terms of data availability, Alicante and Braila differs:

- Alicante: we were able to obtain all relevant data from Alicante to run the experiments.
- Braila: Braila has just recently installed sensors into their pipeline system and we were not able to obtain data for the district we will be observing. Nevertheless, that did not present an obstacle as: (1) Braila has sent us data for an adjacent district and (2) due to a fact, Alicante and Braila subcases are identical, all the experiments, we have run on Alicante data are fully pertinent to Braila as well.

Alicante has started installing water sensors approximately 10 years ago. Which means, Alicante is able to provide a long history of water consumption, which is of great value for water prediction task. On the other hand, sensors for the district have been installed in Braila just recently and lack of historical data may affect results of the experiments.

In the case of Alicante, despite its long history of data collection, Alicante has been installing sensors in batches and unifying data collections over time. The data coherency is the reason we have decided to predict water consumption for 7 selected Alicante districts – as data for those 7 districts are the most complete and unified. The districts are: Rambla, Montañeta, Mercado, Diputación, Benalua, Autobuses, Alipark.

In the case of Braila, the subcase is limited to one district (Radunegru).

The data structure is currently provided in .cvs. Clearly, that will change as data will be transmitted in appropriate data models. However, just for the sake of clarity, let's describe data, relevant for the consumption prediction.

The data structure's header is:

```
CA009-52;;CT005-52;;
```

and a typical row is:

```
31/12/2018 14:03:36;57,871;31/12/2018 14:30:00;9163255,24;
```

Obviously, we have 4 columns, two for each sensor ID, CA009-52 and CT005-52. The important pair are the first two columns:

```
31/12/2018 14:03:36;57,871
```

The first number before the semicolon (31/12/2018 14:03:36) is a date time stamp. The second one (57,871) is describing a water flow (with unit m3/h) on the time of measure.

The two last numbers from the example are:

```
31/12/2018 14:30:00;9163255,24;
```

Again, the first number before the semicolon (31/12/2018 14:30:00) is a date time stamp. The second number (9163255,24) denotes aggregated water consumption. The difference between two consecutive

rows will give the amount of water that has been consumed within the given period between two date timestamps.

8.2.3 Long-term Modeling with ARIMA and SARIMA

1. Data

The model is implemented and tested with real historical data from the city of Alicante (Spain) and more specifically the Benalua district. The measures of the data come almost at two times per minute on daily water consumption and the duration of these data spans to seven years (January 2013 to December 2019).

The (aggregated) weekly and monthly water consumption time-series used for the long-term prediction in unit of cubic meters per hour (m_3/h) are presented below in Figure 24 and Figure 25:

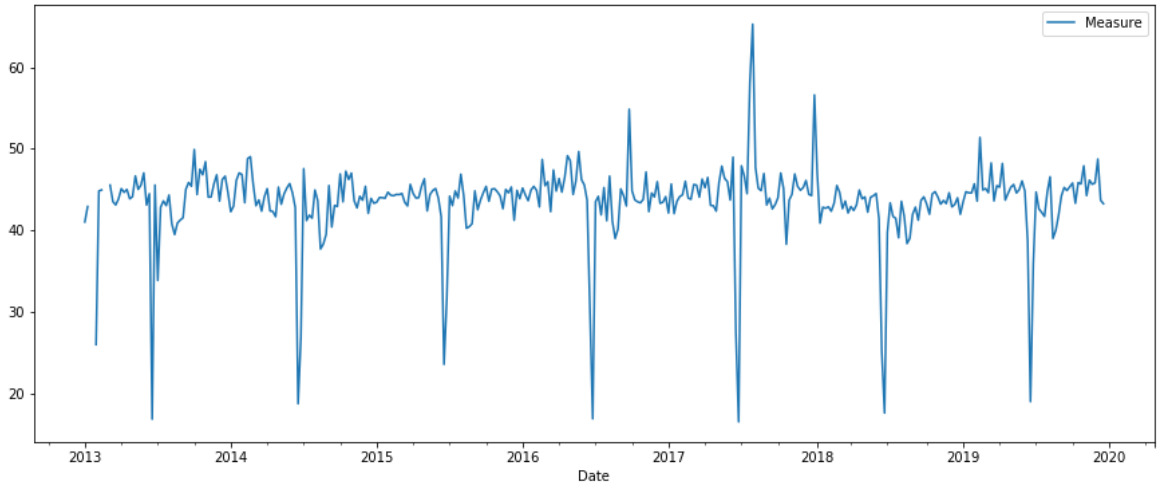

Figure 24: Biweekly water consumption 2013 - 2019 in m^3/h.

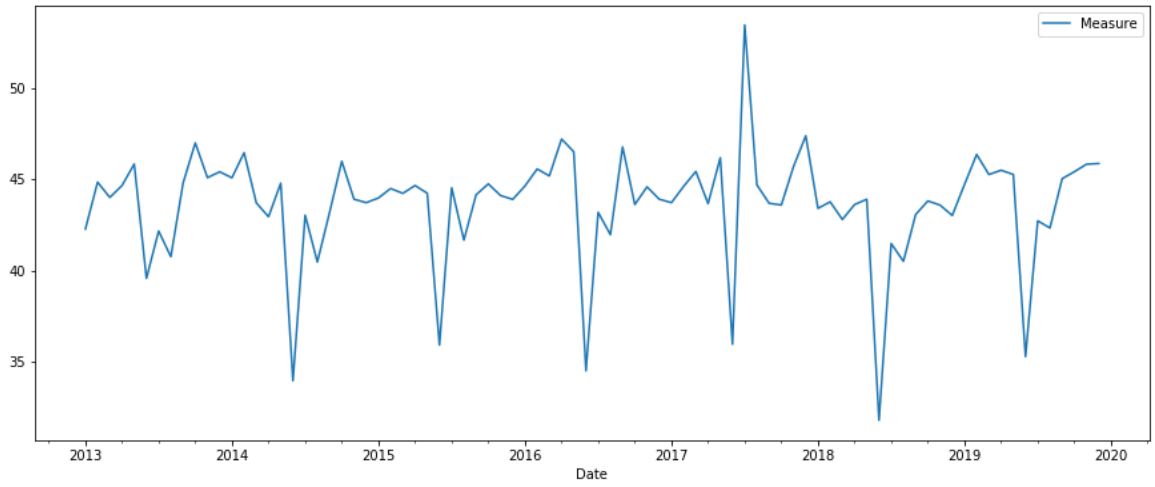

Figure 25: Monthly water consumption 2013 - 2019 in m^3/h.

During the pre-processing face of the data, missing data and invalid values discovered throughout the entire dataset were excluded. Since a large portion of the data during the 2013 were found to be null, the

entire year was excluded from the final dataset that was used during the training and testing process in order to be able to achieve better results. The number of data after the final pre-processing ended up being at a little over 6 million entries of water consumption recordings.

D-F Statistic:	-9.843328
p-value:	0.000000
Critical Values:	
1%:	-3.433
5%:	-2.863
10%:	-2.567

Table 9: Statistical tests for D-K test as described in the Section 4.1.

In Table 8 the results of the D–K test are presented. All values indicate that the time series is stationary since first of all the p-value falls under the 0.05 threshold but also because the result of D-F statistic remains under the critical values of 1%, 5% and 10%.

2. Model Selection

The model with the lower AIC score (check definition in Section 4.1) is expected to strike a superior balance between its ability to fit the data set and its ability to avoid overfitting the data set and therefore is the one that will be selected. In our case the output that I produced suggests that SARIMA(1, 1, 1)x(1, 1, 1, 12)12 gives the lowest AIC value of 204.7 for the monthly prediction and SARIMA(0, 0, 1)x(0, 1, 1, 12)24 gives the lowest AIC value of 583.6 for the biweekly prediction. Therefore, we should consider these to be the optimal choices among all the models we have considered for both cases respectively.

3. Results

By combining and synthesizing all the interim results that have been presented in the previous sections we hereby present the final results of our methodology that are concluded with the biweekly and monthly prediction of the water demand from the start of the year of 2020 until the end of year 2022:

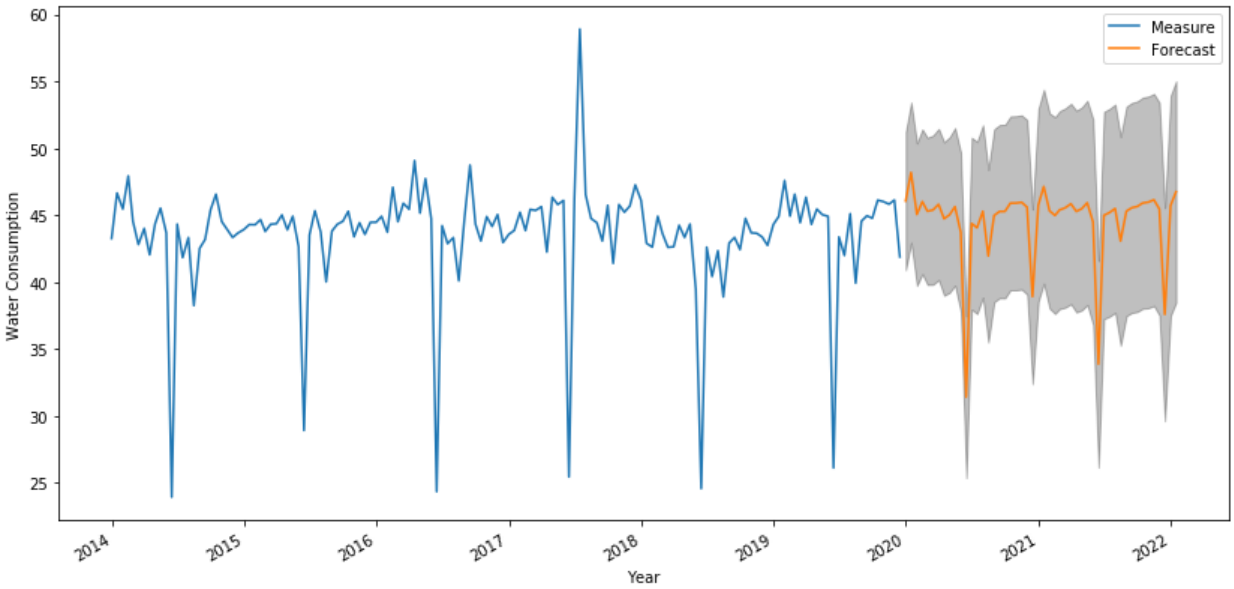

Figure 26: Biweekly prediction of water demand in m^3/h for the next 2 years

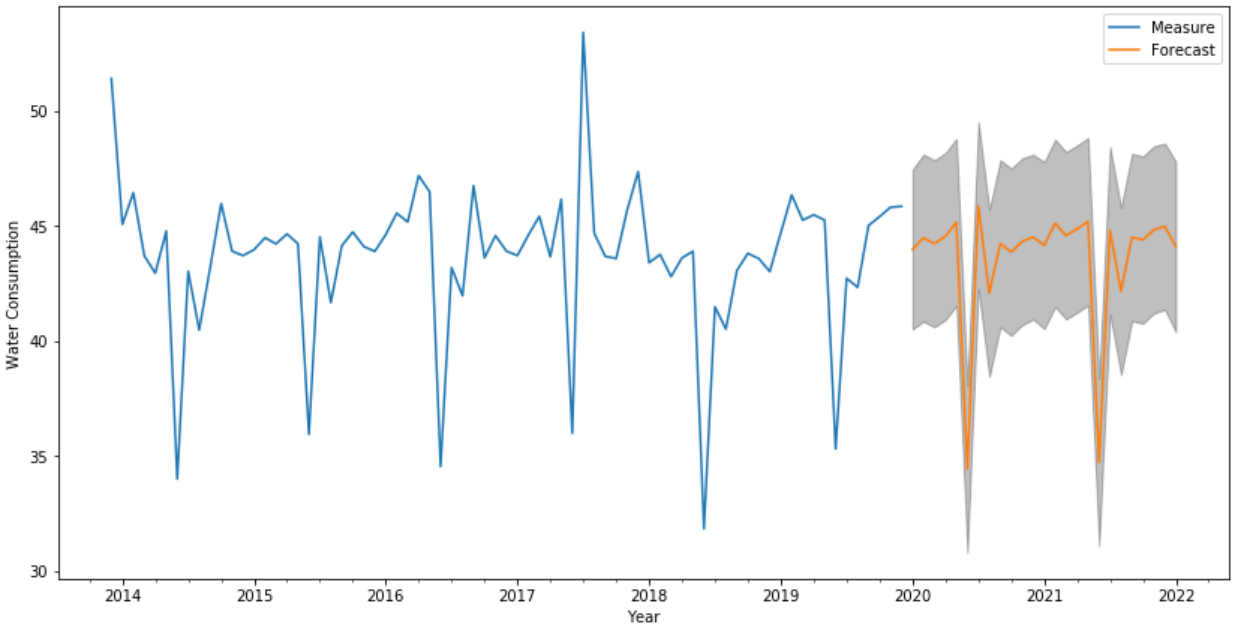

Figure 27: Monthly prediction of water demand in m^3/h for the next 2 years

The blue line indicates the observed values from the dataset that was used whereas the orange line shows the predicted values that our model estimated. The grey area in the part of the predicted values is the 95% confidence interval which indicates the potential values that the prediction could take. As it is visible, on average the value of water consumption falls close to 45 m^3/h in most of the time in both cases with the exception of the dive that it takes once every year towards the end of June. This has occurred because this dive is met in each year of the data that we processed and used. More specifically it is encountered during the second or third week of June where the measures show a sudden downfall around this time of the year.

The small void that is visible between the prediction and the observed values is due to the fact that some small portion of data during the end of December 2019 are missing. Since the prediction starts from January 2020 and we have combined the visualization of the two plots in the same graph, the graph does not contain any values in this specific time.

In the following table (Table 9) there is a representation of the summary of the outcomes that were produced during the methodology process that was followed on this approach:

	Number of observations	**RMSE**	**MAPE**	**Model structure**	**AIC**
Monthly	73	4.475	6.615436265883	(0, 1, 1)x(1,1,1,12)12	204.7501137442638
Biweekly	157	5.690	8.438892526082	(0, 0, 1)x(1,1,1,12)24	583.6048783507828

Table 10: Overview of results.

8.2.4 Classical Machine Learning Approach

8.2.4.1 Data

For Alicante water consumption prediction, we used data for Benaula district for the year 2013 - 2019. Data available to us contained the water flow time series with a frequency of 30 seconds.

The data was split 2/3 for training the models, and 1/3 for testing. For weather data, we expect to get data later from other partners in the project, but for initial experiments, we downloaded data for Alicante airport from the web[8]. This date does not contain all the features that we would like to explore, but it contained temperature, pressure, and humidity. In this data, we miss the precipitation and wind power, which intuitively would be good predictors of water consumption especially precipitation.

8.2.4.2 Feature engineering and feature selection

In our experiment, our goal was to predict the daily water consumption. Our data contained water flow data therefore we just tried to predict average daily water flow. To do that we transformed water flow data to daily average water flow.

To test different learning algorithms, we had to also engineer some features. We used two kinds of features. The first one is the shift of data. That way we used past data to predict the future. But only raw past data are typically not enough, therefore we also calculated the mean water flow in a time window. After the talk to the experts and explanation of current models used for prediction of consumption, we decided that we will calculate these features for up to one year. We added as a feature also the day of the weak, because we expect that the same days have similar events.

That way we get a lot of new features, but using all of them would be inefficient, therefore we have to use some sort of feature selection. The selected features were selected with the help of a correlation matrix. The important aspect of choosing features in machine learning is a good correlation with the predicted feature. But it is also important that each feature brings as much new information to the table. Thus, we decided that we will choose features most correlated with water flow that is not too correlated (<0.75) between each other.

When we see what kind of features are chosen, we see that that kind of approach has a lot of sense because most of the features picked up are similar to previous expert chosen features. Some features are from last week like average water flow for 1 day, 3 days, and 5 days. That makes sense because the most recent data points tell us most about the state of the system. The other data are shifts for 7 days, 14 days, and one year. This also makes sense because it considers data from the same day one and two weak ago and data from the same time one year ago.

8.2.4.3 Results

The experiment was conducted in stages first we used only past water flow data, then we added the day of the weak and finally, we added weather features. The models are from python library sci-kit learn. To evaluate the results, we use the R2 score.

Data set	Past water flow data	Past water flow data + day of weak	Past water flow data + day of weak + weather
LinearRegression	0.639	0.668	0.708
DecisionTreeRegressor	0.484	0.589	0.405
RandomForestRegressor	0.651	0.658	0.695

[8] https://rp5.ru

GradientBoostingRegressor	**0.665**	**0.704**	**0.715**
SVR	0.065	0.072	-0.053
KNeighborsRegressor	0.639	0.646	0.632

Table 10: R2 score for various regressors.

The results in the table 10 show that the best regressor in all cases is GradientBoostingRegressor. LinearRegression and RandomForestRegressor are slightly worse but still competitive. We also notice that each addition of new data improved the prediction strength of algorithms. The data in the table are not normalized, often in data mining tasks, the normalization is also an important preprocessing step. To test that, we have used normalized data to train the model. We have found that a normalized data set improves the regression power of most algorithms. The highest R2 score was with GradientBoostingRegressor and reached 0.728.

The other thing that we investigated was how much in the future we could predict water consumption. In the experiment described above, we did it one day in advance. For water management planning it would be useful to predict water consumption also more than one day in advance. Thus, we did the same experiment for more than one day in advance. The table below shows the R2 score for GradientBoostingRegressor for 1 to 5 days.

Days	R2 score
1	0.715
2	0.507
3	0.428
4	0.416
5	0.454

Table 11: R2 score for GradientBoostingRegressor for several days in advance.

We see that it falls significantly already on the first day and after the third-day settles at a score of around 0.42.

8.2.5 Deep Learning Approach

8.2.5.1 Data preparation

Data was read from a file and then averaged by the day. Additional features were then created (consumption difference (since previous day), day of the week and month). Water consumption and water consumption difference were then also shifted and averaged (with respect to the prediction horizon).

8.2.5.2 Feature selection

Several feature selection techniques were tested, and (according to R^2 score) the best was SelectKBest (with k being 13) and with weekday and month as preselected features.

8.2.5.3 LSTM specific data preparation

After data was split to train and test sets it was normalized (with mean and standard deviation of train set). This data (both train and test sets) was then used to generate subsets (windows) of consecutive records (rows in data frame) of some given size n. This was done by sliding a window of size n over the data frame row by row each time generating new subset and also the corresponding target values vector.

These subsets were then used as samples to train the LSTM and the n parameter determined how far back the LSTM is going to look (`series_size` mentioned below).

8.2.5.4 Building and testing the LSTM

The LSTM was built using Keras and Tensorflow. Different variations of LSTM-s were tested including vanilla LSTM, stacked LSTM, bidirectional LSTM and CNN LSTM.

For testing cross validation was used considering the data is a time series so no data in the testing set should be chronologically before any data in training set. Consequently, when the data was split into folds and fold 2 was used for testing only fold 1 was used for training, fold 3 had fold 1 and 2 for testing and so on. Since LSTM requires a lot of data to be trained on, predictions from two folds gave very bad and unstable scores so they were not considered into the final score.

For scoring R^2 score was used.

8.2.5.5 Results

From the above LSTM variations Vanilla LSTM and Bidirectional LSTM gave the best results.

After some hyperparameter optimization (series_size (n from above), batch size, neuron number and epoch number), the best results from cross validation (for prediction horizon 1) were:

Method	Mean	Variance
Bidirectional LSTM	0.5350	0.2288
Vanilla LSTM	0.5261	0.2496

Table 11: Bidirectional LSTM and Vanilla LSTM results

LSTM with an additional gaussian layer was also tested but the score did not improve.

9 Conclusions and Future Work

This is the mid-term report on water demand prediction toolkit. The report is dedicated to the analysis of the possible approaches to the modeling in the water management scenarios. Several aspects of the modeling have been explored and focus of the work is not solely on the methodologies but also on the supporting components that are needed to successfully deploy the models in the real world.

Architectural issues have also been explored; an extension of a lambda big-data architecture has been proposed for the NAIDES. We propose to use machine-learning based analytics also in the speed (real-time) layer of the system. Furthermore, a couple of solutions for real-time heterogeneous sources data fusion have been proposed (also extending the state of the art in the field).

APIs have been defined and can be used for internal communication between several data analytics components. The APIs are to be considered with APIs defined in NAIADES D5.1 (on anomaly detection) since the underlying machinery should be the same in both cases.

Several algorithms have been studied and several use cases have been tackled with them.

We have studied long-term predictions as well as short-term predictions. Classical approaches for long-term predictions gave satisfactory results. The application of an ARIMA model in time series data is a very common practice that has shown great results in many cases. In the water demand prediction, the results are quite satisfactory but since there are many other techniques and models that could be tested and evaluated there is still room for improvement.

The next steps that could be followed for the prediction model are mainly focused on two parts. The first part is the data that we have for the training of the model. These could be enhanced either with the following measures of the current data from the Alicante district during the year 2020 or we could also utilize measures on water consumption from other use cases inside the project. The second part is the application of other ML models, such as Recurrent Neural Networks (RNN's), that fit the need of this task and explore other potential approaches to water demand prediction.

For short term predictions we have made an extensive analysis of classical batch learning methods, incremental methods and deep learning approaches. Traditional workhorses work the best and they perform better than deep learning. We have also specified two software components that will be able to realize short-term predictions within the NAIADES platform.

Further work will be dedicated to obtain more data from the use cases, analysis of those data and to software engineering part. We need to provide a data pipeline from the sources directly into the analytics platform and then the pipeline to insert the predictions back into the platform. This conceptually simple step is usually the hardest due to (un)availability of the data, various legacy systems at the stakeholders, etc. Another important step for the NAIADES will be integration of analytics platform, because the prediction and anomaly detection services are context unaware.

Bibliography

[1] K. Kenda, J. Peternelj, N. Mellios, D. Kofinas, M. Čerin, and J. Rožanec, "Usage of statistical modeling techniques in surface and groundwater level prediction," *J. Water Supply Res. Technol. - AQUA*, vol. 69, no. 3, pp. 248–265, 2020, doi: 10.2166/aqua.2020.143.

[2] K. Kenda, N. K. Mellios, M. Senožetnik, and P. Pergar, "Computer Architectures for Incremental Learning in Water Management," *Sustainability*, vol. under cons, 2020.

[3] A. Di Nardo, M. Di Natale, D. Musmarra, G. F. Santonastaso, V. Tzatchkov, and V. H. Alcocer-Yamanaka, "Dual-use value of network partitioning for water system management and protection from malicious contamination," *J. Hydroinformatics*, vol. 17, no. 3, pp. 361–376, 2015, doi: 10.2166/hydro.2014.014.

[4] C. Laspidou, "ICT and stakeholder participation for improved urban water management in the cities of the future," *Water Util. J.*, vol. 8, pp. 79–85, 2014.

[5] A. J., F. C. H., P. S. O., O.-Z. B., and S. A., "Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in Montreal, Canada," *Water Resour. Res.*, vol. 48, no. 1, p. W01528.

[6] M. K. Tiwari and J. F. Adamowski, "Medium-Term Urban Water Demand Forecasting with Limited Data Using an Ensemble Wavelet–Bootstrap Machine-Learning Approach," *J. Water Resour. Plan. Manag.*, vol. 141, no. 2, p. 04014053, Feb. 2015, doi: 10.1061/(asce)wr.1943-5452.0000454.

[7] D. Kofinas, N. Mellios, E. Papageorgiou, and C. Laspidou, "Urban water demand forecasting for the island of Skiathos," *Procedia Eng.*, vol. 89, pp. 1023–1030, 2014.

[8] F. Recknagel, "Applications of machine learning to ecological modelling," *Ecol. Modell.*, vol. 146, no. 1–3, pp. 303–310, Dec. 2001, doi: 10.1016/S0304-3800(01)00316-7.

[9] J. Krause, A. Perer, and E. Bertini, "INFUSE: Interactive feature selection for predictive modeling of high dimensional data," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 1614–1623, Dec. 2014, doi: 10.1109/TVCG.2014.2346482.

[10] A. S., C. M., D. S. M., L. B., S. P., and S. J., "Modeltracker: redesigning performance analysis tools for machine learning." ACM, New York, NY, pp. 337–346.

[11] K. Kenda and D. Mladenić, "Autonomous sensor data cleaning in stream mining setting," *Bus. Syst. Res.*, vol. 9, no. 2, pp. 69–79, 2018, doi: 10.2478/bsrj-2018-0020.

[12] R. C. Griffin and C. Chang, "Pretest analyses of water demand in thirty communities," *Water Resour. Res.*, vol. 26, no. 10, pp. 2251–2255, 1990, doi: 10.1029/WR026i010p02251.

[13] A. F., G.-V. M. Á., and M.-E. R., "Estimation of residential water demand: a state-of-the-art review," *J. Socio. Econ.*, vol. 32, no. 1, pp. 81–102.

[14] N. Marz and J. Warren, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2015.

[15] P. Ta-Shma, A. Akbar, G. Gerson-Golan, G. Hadash, F. Carrez, and K. Moessner, "An Ingestion and Analytics Architecture for IoT Applied to Smart City Use Cases," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 765–774, 2018.

[16] F. Koprivec, K. Kenda, and B. Šircelj, "FASTENER feature selection for inference from earth observation data," *Entropy*, vol. 22, no. 11, 2020, doi: 10.3390/e22111198.

[17] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *J. Am. Stat. Assoc.*, vol. 74, no. 366a, pp. 427–431, 1979.

[18] M. Valipour, "Long-term runoff study using SARIMA and ARIMA models in the United States," *Meteorol. Appl.*, vol. 22, no. 3, pp. 592–598, Jul. 2015, doi: 10.1002/met.1491.

[19] S. Mohan and S. Vedula, "Multiplicative seasonal ARIMA model for longterm forecasting of inflows," *Water Resour. Manag.*, vol. 9, no. 2, pp. 115–126, 1995.

[20] S. N. A M Razali, M. S. Rusiman, N. I. Zawawi, N. Arbin, and T. Hussein Onn Malaysia, "Forecasting of Water Consumptions Expenditure Using Holt-Winter's and ARIMA," *J. Phys*, p. 12041, 2018, doi: 10.1088/1742-6596/995/1/012041.

[21] T. Ozaki, "On the Order Determination of ARIMA Models," *Appl. Stat.*, vol. 26, no. 3, p. 290, Nov. 1977, doi: 10.2307/2346970.

[22] A. E. Ioannou and C. S. Laspidou, "The Water-Energy Nexus at City Level: The Case Study of Skiathos," in *Multidisciplinary Digital Publishing Institute Proceedings*, 2018, vol. 2, no. 11, p. 694, [Online]. Available: http://www.mdpi.com/2504-3900/2/11/694.

[23] K. Kenda, M. Škrjanc, and A. Borštnik, "Modelling of the complex data space: Architecture and use cases from NRG4CAST project," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Jul. 2015, no. 0, pp. 1–4, doi: 10.1109/IISA.2015.7388056.

[24] K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming Data Fusion for the Internet of Things," *Sensors*, vol. 19, no. 8, 2019, doi: 10.3390/s19081955.

[25] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4. Association for Computing Machinery, 2014, doi: 10.1145/2523813.

[26] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "{MOA:} Massive Online Analysis," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, 2010, [Online]. Available: http://portal.acm.org/citation.cfm?id=1859903.

[27] G. Hulten, L. Spencer, and P. Domingos, "Mining Time-Changing Data Streams lau ries @ innovation- next.corn," pp. 97–106, 2001, doi: 10.1145/502512.502529.

[28] W. Fan and A. Bifet, "Mining Big Data : Current Status , and Forecast to the Future," *ACM SIGKDD Explor. Newsl.*, vol. 14, no. 2, pp. 1–5, 2013, doi: 10.1145/2481244.2481246.

[29] E. Ikonomovska, J. Gama, and S. Džeroski, "Online tree-based ensembles and option trees for regression on evolving data streams," *Neurocomputing*, vol. 150, pp. 458–470, 2015, doi: 10.1016/j.neucom.2014.04.076.

[30] A. Ritter and R. Muñoz-Carpena, "Performance evaluation of hydrological models: Statistical significance for reducing subjectivity in goodness-of-fit assessments," *J. Hydrol.*, vol. 480, pp. 33–45, 2013.