REPORT ON DELIVERABLE 4.1.1

Consumer Data Analysis Components

PROJECT NUMBER: 619186
START DATE OF PROJECT: 01/03/2014
DURATION: 42 months

DAIAD is a research project funded by European Commission's 7th Framework Programme.

Dissemination Level	Public
Due Date of Deliverable	Month 18, 01/09/2015
Actual Submission Date	08/07/2016
Work Package	WP4 Consumer Data Analysis
Tasks	4.1 Data management 4.2 Consumption patterns and event detection 4.3 Personalization and recommendations
Type	Prototype
Approval Status	Submitted for approval
Version	1.2
Number of Pages	58
Filename	D4.1.1_DAIAD_Consumer_Data_Analysis_Components.pdf

Abstract

This report provides an overview of the Prototype Deliverable D4.1.1 "Consumer Data Analysis Components", which includes all current versions of software components developed in WP4 for supporting consumer data analysis of water consumption data from a single household. First, we describe the data management engine (T4.1) for storing water consumption data, as well as contextual data. In the following, we present our work on developing the pattern recognition & event detection (T4.2), as well as the personalization framework (T4.3).

History

version	date	reason	revised by
0.1	16/07/2015	Initial Version	Yannis Kouvaras
0.2	03/08/2015	Added new sections	Giorgos Giannopoulos
0.3	12/08/2015	Review	Spiros Athanasiou
0.4	21/08/2015	Various edits in several sections	Giorgos Giannopoulos
0.5	24/08/2015	Various edits in several sections	Yannis Kouvaras
1.0	25/08/2015	Final version	Spiros Athanasiou
1.1	01/06/2016	Corrected Figure references – updated report template for MS Office 365/PDF style incompatibility	Spiros Athanasiou
1.2	08/07/2016	Updated all sections and introduced additional sub-sections following reviewers' feedback received during the Y2 review meeting; document re-authored as a Report deliverable	Giorgos Giannopoulos, Pantelis Chronis, Nikos Karagiannakis

Author list

organization	name	contact information
ATHENA RC	Spiros Athanasiou	spathan@imis.athena-innovation.gr
ATHENA RC	Giorgos Giannopoulos	giann@imis.athena-innovation.gr
ATHENA RC	Nikos Georgomanolis	ngeorgomanolis@imis.athena-innovation.gr
ATHENA RC	Yannis Kouvaras	jkouvar@imis.athena-innovation.gr
ATHENA RC	Pantelis Chronis	pchronis@imis.athena-innovation.gr
ATHENA RC	Nikos Karagiannakis	nkaragiannakis@imis.athena-innovation.gr

Executive Summary

This report provides an overview of the Prototype Deliverable D4.1.1 "*Consumer Data Analysis Components*" that includes the implementation of the following software artifacts: (a) a data management engine for water consumption data, (b) a pattern and event detection framework and (c) a personalization framework. The development of software components of T4.2 and T4.3 follows the planning of the entire project, and thus the availability of highly granular water consumption data. Consequently, the current versions for pattern recognition, forecasting and event detection have been developed and evaluated against early water consumption data of low granularity (*at best 1h aggregates*). Our work towards improving and finalizing these software components will continue throughout the course of WP4, based on highly granular water consumption data generated during the pilot (*fine-grained, labeled*), as well as user feedback to evaluate the pattern recognition and personalization facilities.

The database engine and the algorithm on joint pattern forecasting have already been integrated into DAIAD@home prototypes. As development continues, we are going to fine tune the database engine and enhance the algorithms for pattern detection, forecasting and personalization. Our goal is to have all the components implemented, fully tested and integrated into DAIAD@home application by M23 in order to begin the user trials (Prototype Deliverable D4.2.1 "DAIAD@home software for trials").

The remainder of this document is structured as follows.

Section 1 provides an overview of the Prototype Deliverable D4.1.1.

In Section 2, we present our progress on the implementation of the data management engine of T4.1 which supports the acquisition, storage, querying and archiving of real-time water consumption data, as well as contextual data (*e.g. user profile, weather, demographics)*. The data engine manages data generated from a single household by one or more DAIAD@feel devices, as well as smart water meters (if available).

In Section 3, we present our work on researching and implementing novel methods for performing pattern recognition, consumption forecasting, event detection (T4.2) and personalized recommendations (T4.3) on the household level. Given their strongly intertwined nature and the fact that personalization is *dependent* on pattern recognition, during this period we have emphasized the development of pattern recognition and forecasting facilities. First, we present our analysis on three water consumption datasets, followed by an evaluation of state of the art approaches for time-series forecasting on these datasets. Next, based on our findings on the specificities of water consumption data and on the discovered shortcomings of current time-series forecasting algorithms, we describe our development of two novel algorithms: (a) a Pattern Forecasting algorithm that jointly handles pattern recognition and forecasting tasks, and (b) a Model Change Detection algorithm that optimizes the training, and, thus, the precision of forecasting algorithms, by identifying points in time where the model/behavior of the time-series changes. Finally, we present our advancements on exploiting the currently implemented pattern recognition and forecasting methods for event detection, alerting and personalized recommendation functionalities.

Abbreviations and Acronyms

ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	Autoregressive Integrated Moving Average
BLE	Bluetooth Low Energy Bluetooth
CLR	Classed Linear Regression
ENET	Elastic-net regularized Linear regression
ITM	Iterative Training Method
JSON	JavaScript Object Notation
MAPE	Mean Average Percentage Error
MAE	Mean Absolute Error
NMAE	Normalized Mean Absolute Error
LAR	Linear (Auto-)Regression
REST	Representational State Transfer
SR	Sequential Regression
SVM	Support Vector Machines
SVR	Support Vector Regression
SWM	Smart Water Meter

Table of Contents

1. Overview

The Prototype Deliverable D4.1.1 'Consumer Data Analysis Components' comprises all software delivered in the context of Tasks 4.1, 4.2, and 4.3, which along with the appropriate interventions of D3.2.1 "Sustainability Dashboard", are packaged to deliver the DAIAD@home applications (D4.2.1 'DAIAD@home software for trials'). In summary:

- T4.1 provides the data management engine for efficiently storing, managing, and querying real-time water consumption data and metadata (SWMs and amphiro b1). The engine supports two modes of operation (local and cloud-based) and is also responsible for feeding all other components with water consumption data (T4.2, T4.3, D3.2.1).
- T4.2 and T4.3 receive water consumption data from the data engine of T4.1 and deliver analytics and insights for individual households. Their output is conveyed to consumers using the interventions of D3.2.1.
- D4.2.1, i.e. the DAIAD@home mobile and web applications, combine all of the above software components and UI elements into two self-sustained applications for consumers.

In the following we present the individual components of Prototype Deliverable D4.1.1, how they interact with each other, and how they are assembled to deliver the DAIAD@home mobile application. An overview of the architecture in the context of the DAIAD@home mobile application is shown in Figure 1.

- At the bottom level, the Data Engine is located. The Data Engine is powered by the SQLite[1] database and is responsible for storing, indexing and querying application data. Application data consists of the household member user profiles, amphiro b1 device registration and configuration options, and amphiro b1/SWM measurements. The detailed schema of the database is available in Annex: Data Schema.
- Stored data is analyzed by the Data Analysis Components which implement algorithms for detecting consumption patterns and generating recommendations relative to the user's water consumption behavior. Moreover, the Data Analysis Components access the DAIAD server and download SWM consumption data and analysis results generated at the server. The latter provide better insights for the user's water consumption behavior since they take into account smart water meter data as well as data for all the users of the utility.
- The execution results are optionally persisted by the Data Engine and visualized using the user interface components presented in Deliverable D3.2.1 'Sustainability Workbench'.

[1] https://www.sqlite.org/

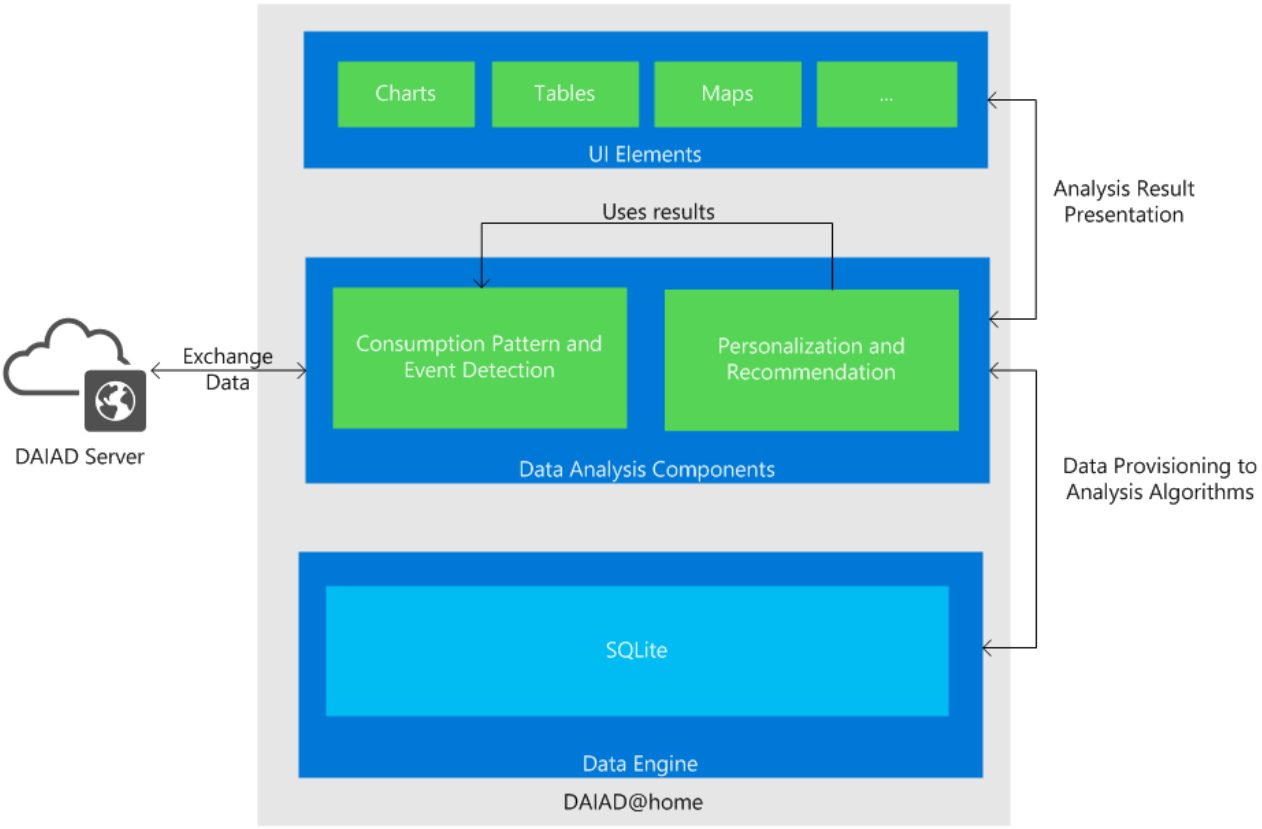

Figure 1: Consumer Data Analysis Components

2. Data engine

In this section, we present our progress on the implementation of the data management engine of T4.1 which supports the acquisition, storage, querying and archiving of real-time water consumption data and contextual metadata. The data engine manages data generated from a single household by one or more DAIAD@feel devices, as well as smart water meters (if available).

2.1. Data lifecycle

The data engine has been developed to efficiently store, manage and query real time water consumption data, as well as contextual metadata (*e.g. user demographic profiles, weather data*) of a single household. As such, it accommodates the needs of data generated by the DAIAD@feel water sensor developed in WP2 (*Amphiro b1*) and a smart water meter. According to Deliverable D1.2 "*DAIAD Requirements and Architecture*", *more than one* DAIAD@feel devices can be installed in a single household. Further, the total water consumption of the household can be *optionally* monitored by a smart water meter at arbitrary time intervals. Consequently, the engine is responsible for serving the data management, analysis, and knowledge extraction needs of all *possible permutations* (*e.g. only amphiro b1 installed, only smart water meter available, multiple amphiro b1 and smart water meter*).

In addition, the data engine is integrated in DAIAD@home, the software providing analysis, knowledge discovery services, and interventions to members of *a single household*. According to Deliverable D1.2 "*DAIAD Requirements and Architecture*", DAIAD@home is available in two versions (*mobile and web*). As a result, data managed by the engine of T4.1 may also reside in the DAIAD cloud infrastructure of WP5, and thus the Big Water Data Engine of T5.1 (*cf. Deliverable D5.1.1 "Big Water Data Management Engine"*) to enable the web-based operation of DAIAD@home.

An end-to-end representation of the lifecycle of water consumption data according to this model is depicted in Figure 2. Specifically, the data flow includes the following possible steps.

1. One or more DAIAD@feel devices are paired to DAIAD@home mobile applications using the Bluetooth Low Energy (BLE) specification. Water consumption data is emitted to the paired application either in *real time* or in a *batch mode*, depending whether the mobile device is in the vicinity of a DAIAD@feel device. The new version of the DAIAD@home mobile application has already implemented support for the new, BLE enabled, amphiro b1 device. The implementation supports discovery, pairing and data transfer operations.
2. The water consumption data is stored locally in the mobile version of DAIAD@home. The stored data is processed locally using the algorithms developed in Tasks 4.2 and 4.3. Measurements, analysis results, and interventions are visualized by the DAIAD@home mobile application. An example of such visualizations is illustrated in Figure 3.

3. When Internet connectivity is available, data from DAIAD@home is uploaded periodically to the Big Water Data Engine of WP5 via the HTTP API (see section 2.2.2). The existing API has been extended in order to support all currently available DAIAD@feel devices, i.e. Amphiro a1 (*used mostly for prototyping*) and b1 devices. Further, it is future-proof, as the API is inherently and easily extensible to support the interconnectivity needs of future DAIAD@feel devices. In addition to detailed measurements, the API supports the transfer of aggregated data, device and user profiles, as well as a deep-level device-control capabilities (*e.g. alter LCD display*).

4. Data is processed by the algorithms developed by Tasks 4.2 and 4.3. The stored data and analytical results are visualized by DAIAD@home. The DAIAD@home web application and provides full support for user profiles, device management and smart water meter integration.

5. Data analysis results, recommendations and contextual data can be optionally uploaded (*i.e. if the user has opted-in to this functionality*) by the DAIAD cloud infrastructure of WP5 to DAIAD@home in order to augment the application functionality and user experience.

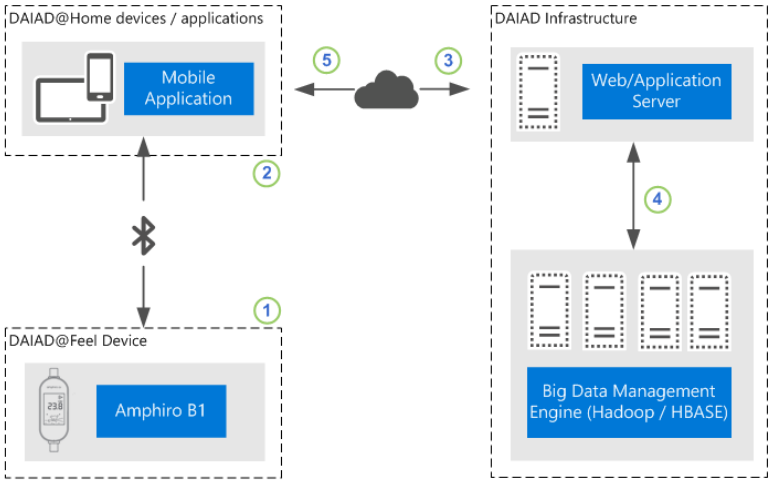

Figure 2: Data Lifecycle

2.2. Implementation

In this section we present the data management engine of T4.1 in more detail. First, we describe the database schema and the HTTP APIs used for exchanging data with the DAIAD Server. Then, we enumerate the data management and analysis features offered by the data engine.

2.2.1. Local data management

At the level of the mobile device, data storage is handled by the SQLite database. The data management engine is responsible of storing and managing data about users, devices and measurements. More specifically, user data is stored as relational tuples that contain information about the user account such as authentication and identification data, as well as demographic characteristics (*e.g. gender, residence, age*).

For each user there can be *one or more* paired DAIAD@feel devices. Information for each paired device is also stored as a relational tuple. Since information about devices is highly volatile and the corresponding data schema is prone to change very often (*e.g. support new devices, firmware updates*), instead of storing

each device property in a single table column, device information is formatted as a JavaScript Object Notation (JSON) document, serialized and stored in a single text column. As the number of devices is always small, there are no performance implications from serialization and deserialization of device information.

Device measurement data is also stored as relational tuples, with the corresponding table having *a fixed schema*. As measurement data is eventually determined by the DAIAD@feel devices, the table schema is not likely to change too often. Detailed information about the data schema is available in Annex: Data Schema.

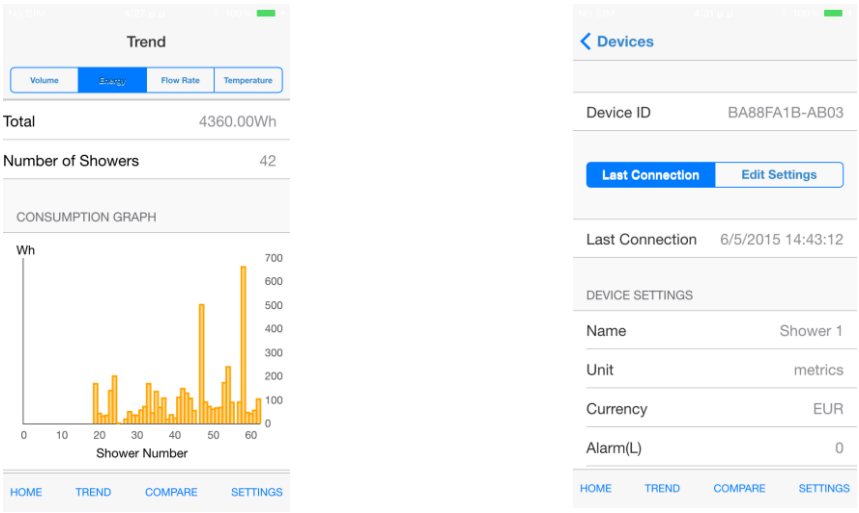

Figure 3: DAIAD@home mobile application

As the DAIAD@home application evolves, it is expected that the user profile schema will be continuously enhanced by analysis results; hence more flexibility is required for storing user profiles. We currently examine the option of discarding the user and device tables and storing the user and device data to an external file on the mobile device local storage. Once more, performance is not a problem since mobile devices in a household have a single or few users (i.e. household members).

2.2.2. HTTP API

DAIAD@home mobile application requires internet access only for user registration. After a user is registered, the DAIAD@home mobile application can operate offline by persisting all data locally using the Data Engine.

However, the user experience can be greatly improved if the application exchanges data with the DAIAD server. DAIAD@home mobile application has access only to amphiro b1 data which is not accessible to DAIAD server. Simultaneously, the DAIAD server has access to smart water meter which is not available to the DAIAD@home mobile application. Further, DAIAD server performance greatly surpasses any mobile device when it comes to data analysis.

In order to improve the functionality of DAIAD@home mobile application the following HTTP APIs are provided for exchanging data between the application and the server:

- Data API: Uploads amphiro b1 data to the server and supports querying historical data for both amphiro b1 and smart water meter devices. Smart water meter data can also be aggregated on demand per daily, weekly, monthly or yearly basis. Moreover, detailed time series for real-time amphiro b1 water extraction sessions can be fetched.

Another important feature offered by the Data API is that the users can reinstall the application without losing any of their data. The application will load data on demand automatically from the server.

- Profile API: Allows users to save and load their profiles to and from the remote server. This feature allows device registration persistence even when a user installs the application to a new device. Already paired amphiro b1 devices are automatically registered without any user intervention.
- Device API: Allows the application to automatically discover amphiro devices registered by the same user but from a difference mobile device. For instance, after a device is registered by a tablet, the user can access the newly registered device from her smartphone.
- Message API: Provides user with alerts, recommendations and tips that are generated at the server after the execution of data analysis algorithms. Moreover, it allows users to receive important announcements from the utility such as scheduled water supply disruptions.

All APIs send and receive JSON encoded messages. The documentation of the APIs is available at https://app.dev.daiad.eu/docs/api/index.html. All data received by the server is also persisted locally by the Data Engine and is available even when the application is offline.

2.2.3. Simple Data Analytics

The Data Engine supports the execution of simple data analytics that allow users to gain valuable insights relative to their water consumption behavior. The main data analysis operations supported are:

- Aggregation: Computes simple aggregates for smart water meter and amphiro b1 data. For smart water meter data, aggregation is performed over variable time intervals e.g. daily total consumption over the last two weeks. For amphiro b1 data, aggregates are computed on scalar properties such as volume, energy and temperature over a list of consecutive extraction sessions e.g. average energy consumption for the last ten showers.
- Comparisons: Based on aggregation results, the Data Engine enables simple data exploration by calculating comparisons of smart water meter data over different time intervals (e.g., compare the daily total consumption for month January of the years 2015 and 2016). Similar comparisons are also computed for amphiro b1 data over different sequences of extraction sessions (e.g. average shower temperature for the last 10 sessions and the 10 sessions before them).
- Alerts: The Data Engine implements simple rules for pattern detection that are validated at regular intervals. Whenever a match is detected appropriate alerts are generated. For instance, if a continuous stream of amphiro b1 measurements is stored that spans a long interval, an alert is displayed.
- Recommendations: The Data Engine augments the local user data with remote data downloaded from the DAIAD cloud infrastructure using the HTTP APIs. Using this data, the Data Engine is able to compare the user water consumption behavior to that of other users with similar demographic characteristics. If the user water consumption is much higher than the average water consumption, useful recommendations are generated to help the user improve her water usage efficiency.

3. Pattern recognition and forecasting

In this section, we describe our work on implementing the currently available algorithms and facilities for the pattern recognition and event detection framework (T4.2) and the personalization framework (T4.3). The goal of T4.2 is to offer user-oriented consumption analysis facilities on the household level. Specifically, it revolves around three directions: *consumption forecasting, consumption pattern recognition and alerting mechanisms on detected consumption events*. Time-series forecasting algorithms constitute the cornerstone of T4.2, since two out of the three aforementioned directions are based on them: consumption forecasting and event detection/alerting. The goal of T4.3 is to exploit the outcomes of T4.2 for producing *personalized recommendations* on users w.r.t. their consumption behavior. Essentially, by being able to identify fine-grained consumption types (patterns), we aim at mapping specific consumption events or types to specific household users.

During this period, our work focused on identifying the specificities of the water consumption analysis problem, w.r.t. both datasets and current state of the art algorithms, and on developing novel algorithms for performing pattern recognition and consumption forecasting on water consumption time-series. In parallel, we have designed and implemented the basic functionality for event detection, alerting mechanisms, and personalized recommendations. The results of our work will be made gradually available to the DAIAD trials participants, according to the treatment phases of each trial.

3.1. Overview of work

Our course of work is divided into three phases. The first phase consisted in evaluating early water consumption data we obtained, with respect to specificities of the water consumption analysis problems and shortcomings of existing methods for forecasting and pattern recognition. The other two phases follow an iterative process that lasts until the end of the project and include: (a) proposing, implementing and assessing novel algorithms and models for consumption forecasting and pattern recognition and (b) integrating this output into DAIAD system. Our work in these two phases is *highly data-driven*, in the sense that the algorithms and models we build are *continuously revisited and enhanced*, based on the granularity, richness and volume of the data that we obtain at each period of the project. Further, we require that the implemented models for consumption time-series analysis are made available to the end users (e.g. the DAIAD trials users) as features of the DAIAD system, after they have been assessed on as many, diverse and complex water consumption datasets as possible. Next, we provide a brief overview of the work and the current results produced by each phase.

During the first phase, we focused on studying the data at hand, i.e. consumption time-series produced either by Smart Water Meters or by Amphiro metering devices. These comprised an initial seed of data, obtained from previous works (SWM data gathered by AMAEM, see Sections 3.2.2 and 3.2.3), Amphiro shower events from previous studies, see Section 3.2.1) that helped us perform an evaluation of state of the

art forecasting algorithms. Our evaluation focused on short-term consumption forecasting on individual households, since this is the main focus of WP4. This study provided three major findings:

(i) **Poor performance on individual forecasting.** Most state of the art time-series forecasting algorithms perform *poorly* on the task of short-term forecasting on individual household consumption time-series, mainly due to the low canonicity of the time-series.

(ii) **Time/volume shifting of consumption events.** Consumption patterns that correspond to the same real world events (i.e. taking a shower) often demonstrate variations on their typical mean time of occurrence, duration and volume.

(iii) **Consumption model/behavior changes.** It is often the case that the consumption behavior changes irrespectively of seasonal changes, most probably due to exogenous factors, such as vacation, change in household members, financial reasons, etc. (see also the analysis in Section 4 of deliverable D1.1 "State of the art Report" [Ath14]).

These findings guided our work in the next two phases.

During the second phase of our work, we proposed two novel algorithms towards handling issues (ii) and (iii) identified above. Specifically, we defined and implemented a model that considers *generalized consumption patterns* and *activity zones* within each day and uses them to jointly solve the problems of pattern recognition and consumption forecasting. The proposed algorithm increases the forecasting precision of the underlying machine learning model (Support Vector Regression) and achieves high accuracy in predicting whether a specific consumption activity will take place the next day (Section 3.4).

Further, for handling (iii), we proposed an orthogonal method that can be applied on top of several machine learning algorithms for time-series forecasting. Our algorithm optimizes the training of a forecasting model by identifying the most suitable historical point in the time-series that represents a *change in the behavior/model* of the time-series. The novelty (and major difference compared to existing change point detection methods) lies in that the algorithm takes into account the *effectiveness* of the trained machine learning model in order to identify the point of change. The proposed method significantly improves the forecasting precision of three different machine learning algorithms (Support Vector Regression, Elastic-net regularized Linear regression and Artificial Neural Network) (Section 3.5).

As mentioned before, the development and assessment of the implemented algorithms follows an iterative process, according to which the algorithms are re-evaluated and enhanced, upon the provision of new water consumption datasets. Given that, the aforementioned algorithms will be re-assessed on highly granular water consumption data that will be obtained from the trials, which will help us further improve their accuracy.

The third phase consists in integrating the outcomes of our research-oriented work into DAIAD@home. As stated above, this is an iterative process that involves the thorough and recurring assessment of the algorithms on real world data and users.

In the following sub-sections, we describe in more detail the datasets at hand; the study we performed and the insights we gained; the two novel algorithms we have implemented and evaluated for (a) joint pattern recognition and consumption forecasting, and (b) consumption model change identification and forecasting optimization; and our current work on implementing event detection and personalization facilities for the

DAIAD system. We note that implementation details on the algorithms that are described next are provided in the Annex (Section 8.2).

3.2. Datasets Description and Analysis

For the evaluation of the current work we used three (3) different datasets on water consumption:

- **Amphiro historical shower data**. Shower water consumption events, measured by Amphiro a1 monitoring device.
- **Individual historical SWM data**. Individual household water consumption, measured by Smart Water Meters (SWM), with hourly granularity.
- **Aggregate historical SWM data**. Aggregate water consumption on multiple households, measured by SWM, with hourly granularity.

Each of these datasets represents a distinct type of water consumption data that has different properties and requirements. Next we describe each of those datasets in detail.

3.2.1. Amphiro historical shower data

The first dataset consists of (a) shower extraction events, measuring the total volume of water used per shower, from Amphiro a1 devices, and (b) detailed demographic information, produced in the context of an external study performed by Amphiro on 77 Swiss households. Each measurement does not include the timestamp of the consumption. However, it is accompanied by metadata, such as the duration of the shower, the water temperature, the flow-rate of the water, the number of stops throughout the shower (e.g. while soaping) and the duration of the stops. The complete list of included metadata for each shower extraction is provided in Annex: Evaluation datasets.

The forecasting problem in this dataset consists in predicting the consumption of the next shower, given the information of the previous showers. In Figure 4 we can see an example of a sequence of consumption events (showers) from the Amphiro historical dataset.

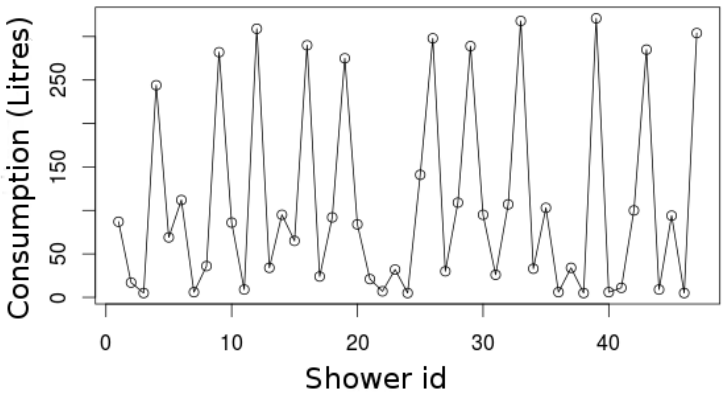

Figure 4: Sequence of shower events measured by Amphiro a1

For this dataset, it is obvious that periodicities or other time-related effects cannot be identified due to the lack of timestamps. By observing Figure 4, there seem to be some patterns in the sequence of consumptions

(e.g. a single high consumption is usually followed by a few smaller ones). Another observation is the appearance of discrete classes of very high, medium and low water consumption events. In Figure 5, we can see a histogram of the consumption values corresponding to these showers, which verifies the aforementioned observation: the clusters of values that seem to form in the histogram suggest the existence of discrete classes of showers of low, medium and high water consumption.

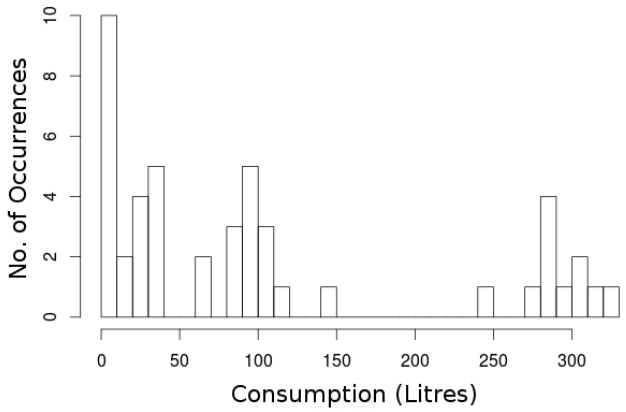

Figure 5: Histogram of Amphiro a1 shower consumptions

The auto-correlogram in Figure 6 depicts the linear correlation of the consumption of each shower with its previous ones. For example, value "0" of the *Lag* axis provides the correlation of showers with themselves and value "5" respectively the correlation of the showers with the showers that happened 5 events ago. The blue horizontal lines, that represent the level of statistical significance, are relatively high because the small length of the time-series does not provide enough samples for confident estimation. The linear auto-correlation does not pass the level of statistical significance for any previous values.

The above observations suggest that in order to model this kind of data, the algorithm requires to be able to identify the different classes of water consumption and model patterns between discrete values. Also, because the sample of observed showers grows relatively slow (e.g. 1-2 showers per day) the algorithm needs to be able to recognize patterns from small samples.

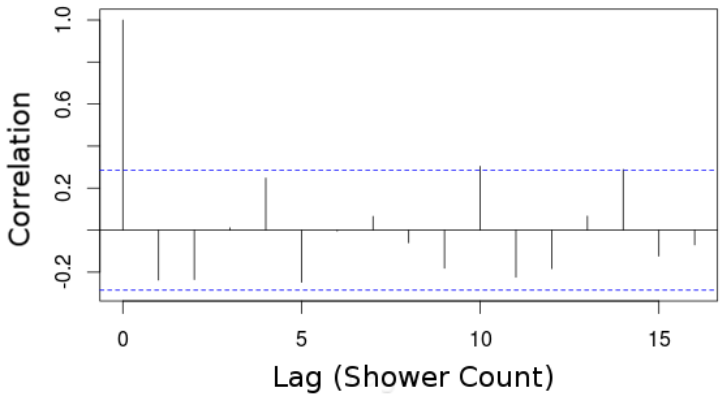

Figure 6: Auto-correlogram of Amphiro a1 shower consumptions

We note that, in order to be able to graphically demonstrate our findings, we provided figures concerning a single household, out of the 77 available. However, the aforementioned analysis was performed on the total of the available data, with similar findings. Indicatively, in Annex: Shower sequences we provide sets of

figures representing consumption events (consumption sequences, histograms and correlograms) for households of different demographics. The observations for these consumption events align with the ones made in our example above.

3.2.2. Individual historical SWM data

The second dataset consists of hourly SWM time-series for 1,000 households in Alicante (selected randomly) covering a time period of a whole year, i.e., from 1st of July 2013 to 30th of June 2014 (8.6M records). The value that each meter outputs corresponds to the total water consumption of the household since the meter has been activated. Each measurement includes the id of the SWM, the timestamp of the measurement and the measured consumption. The exact schema of each record-measurement is provided in Annex: Evaluation datasets.

In order for the data to become more suitable for the algorithms we use, we performed a set of required transformation and cleaning processes. In total, this resulted to the removal of 200 out of the initial 1,000 time-series.

- **Hourly Consumption difference**. Our dataset includes consumption Difference (i.e. current minus previous measurement value) as an optional field. However, most time-series analysis algorithms require as input the difference in consumption between two consecutive measurements. For this reason, we added the Difference field where it was missing in the original dataset.
- **Missing and time-shifted hourly data**. In some cases, measurements are not always provided within exact 1-hour intervals. Specifically, a measurement might be completely skipped-missed or might be provided earlier or later than exactly 1 hour. This can be attributed to several factors: a malfunctioning SWM, temporal RF connectivity issues, third party SWM data software issues, etc. However, (a) most time-series analysis algorithms require as input fixed time intervals between consecutive measurements, and (b) time-series aggregation (see Section 3.2.3) requires that all time-series are *aligned* to each other, containing measurements for the exact same time intervals.

  To address these issues we performed linear interpolation [Mei02], i.e. we divided the time axis in minutes, distributed the consumption of each measurement equally to the minutes of the interval between the current measurement and the previous one, and then added up the minutes of each hour to restore the dataset to the standard time resolution of one hour. This process is based on the assumption that the consumption is performed uniformly in each interval.

  Figure 7 demonstrates the aforementioned process. Let the black circles at time 1.0, 3.0 and 4.5 be the measurements we have obtained from the SWM. The measurement for time 2.0 is missing, while the measurement for time 4.0 is shifted to time 4.5. However, the measurement for time 3.0 includes the aggregate consumption value until this time. Thus, considering a uniform consumption in the interval [1.0, 3.0], we obtain the consumption value for time 2.0 (red circle). Similarly, having the measurements of time 3.0 and time 4.5, we can obtain the value for time 4.0.

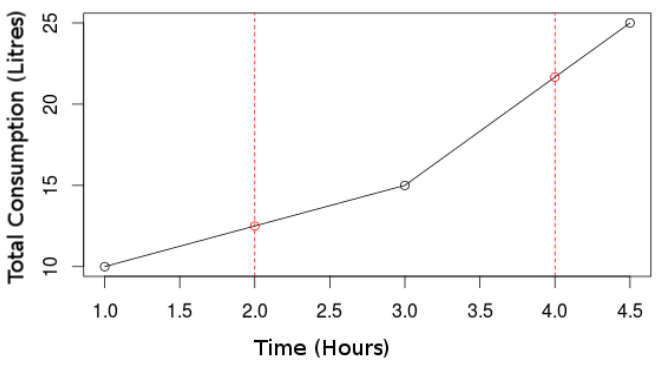

Figure 7: Time-series interpolation

- Negative values. In this dataset there were also time-series that contained negative measurement values, caused by malfunctions of the SWMs. We performed the following cleaning operations.
    - We removed time-series that contained measurements of large negative values (very small negative values also appeared but were considered as benign measurement noise).
    - We removed time-series (10 in total) that had more than 12,000 measurements, which would correspond to more than 500 days included in only one year. The threshold was put by observing the distribution of the number of measurements for the total of time-series and identifying a concentration of outliers for >12,000 measurements.
- Outliers. Finally, we removed the following outliers from the dataset:
    - Time-series that had very large positive values (>1,000 liters/hour). These may correspond to water leak incidents or users with excessive consumption behaviors.
    - We removed households that had zero consumption for more than half of the weeks of the year (e.g. country houses, unrented apartments).

Figure 8 presents a day and a week of a sample time-series of one household after the preprocessing and cleaning of the dataset. We note that the insights we provide have been extracted from examining the complete set of available time-series and can be generalized. In the following, we present them using the time-series of one household as an example. In this example, there appear to be some coarse-grained patterns. The consumption is low at night and is followed by some spikes in consumption, usually two or three, during the day. There is also significant noise that affects both the height and the position of the spikes in time.

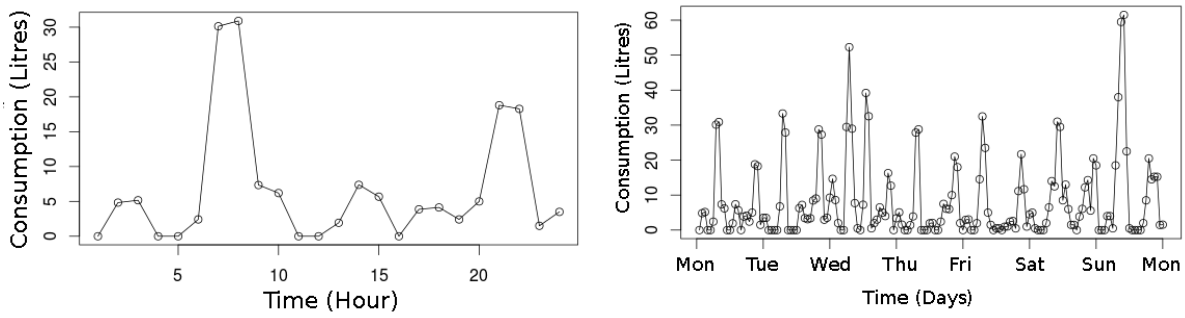

Figure 8: A day (le ft) and a week (right) of hourly water consumption of an individual household

In Figure 9, we can see the histogram of the water consumption for the above sample time-series. We observe that consumption follows a *heavy tailed distribution*, which suggests the existence of outliers in the values. This is a factor that might significantly affect the effectiveness of certain machine learning algorithms (e.g. algorithms that use squared loss functions, which would lead to excessively large errors for outlier values).

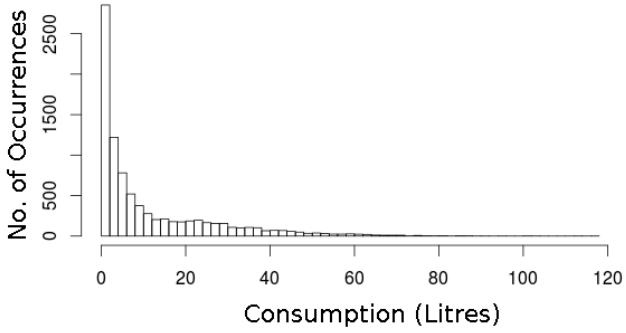

Figure 9: Histogram of individual household SWM consumptions

Next, we can see the auto-correlogram of the sample time-series, which shows the correlation of each measurement with its previous. There is a relatively large correlation between each measurement and the previous one, which is one hour before. Also, there is a clear sign of seasonal structure. The seasonality is daily and weekly as seen by the increased values around 24 and 168. The correlation is generally small but well above the statistically significant level, because of the large size of the data (~8,000 measurements). The relatively small correlation (i.e. large noise) is the main challenge of this kind of data: the noise affects both the magnitude of the water consumption as well as the time of its occurrence, which may shift.

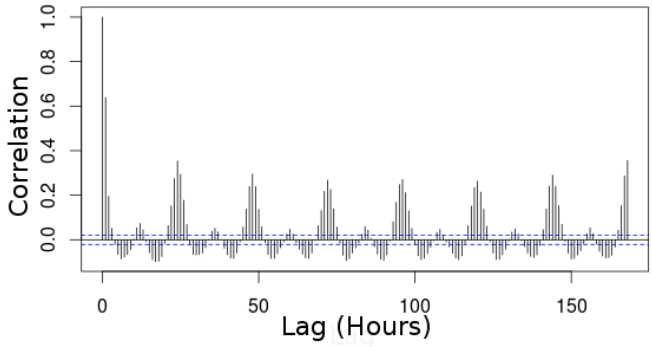

Figure 10: Auto-correlogram of individual household SWM consumptions

3.2.3. Aggregate historical SWM data

The third dataset consists of a single time-series corresponding to the aggregate consumption of all individual consumption time-series presented in the previous subsection. It is derived by considering the aggregate hourly consumption of all SWM time-series of the dataset described in Section 3.2.2, and forming a single aggregate time-series. However, we consider it as separate case, in order to demonstrate the large difference in the effectiveness of state of the art time-series analysis algorithms when applied on individual and on aggregate datasets (see Section 3.3.4).

This time-series is fairly regular and follows a predictable pattern as seen in Figure 11, especially on the right sub-figure, where similar daily consumption patterns can be identified. This happens because, as more time-series of individual consumptions are added, the random noise cancels out and the time-series converges to its average behavior. By adding more time-series, this effect is increased and the aggregate time-series becomes even more regular.

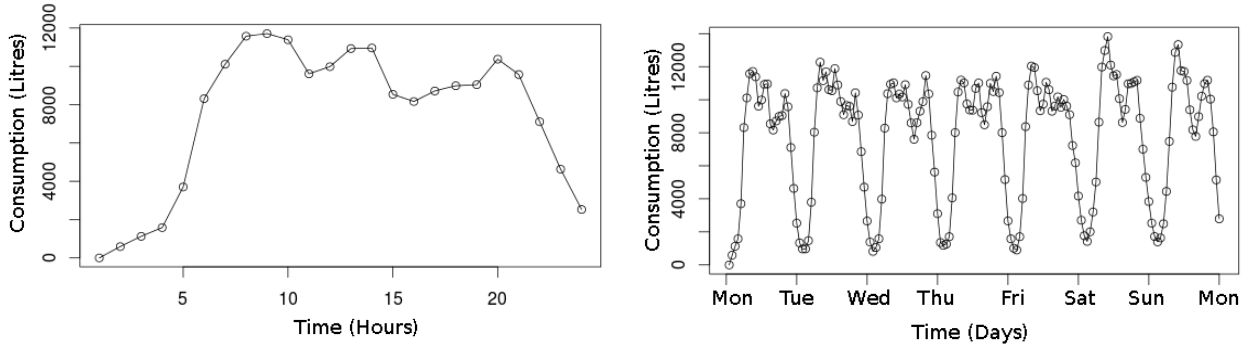

Figure 11: A day (left) and a week (right) of hourly water consumption of aggregate household consumptions

In the histogram of Figure 12, we see that the data expectedly seem to come from a mixture of distributions, one of them being a normal (centered around 9000 in this case).

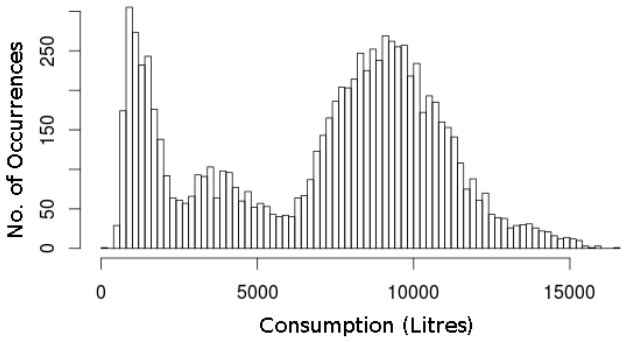

Figure 12: Histogram of aggregate household SWM consumptions

From the auto-correlogram of Figure 13, we can see that there is very large correlation between each value and its previous ones. The periodicity of the correlations suggests the existence of daily and weekly seasonality. It is known from the literature [Tay10] that there also exists yearly seasonality in this kind of data (see also the analysis in Section 3.4 of deliverable D1.1 "State of the art Report" [Ath14]), which we cannot observe currently because we only have only one year of measurements.

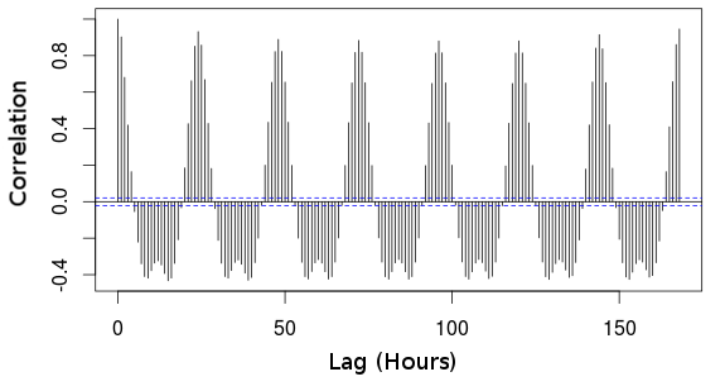

Figure 13: Auto-correlogram of aggregate household SWM data

3.3. Evaluation of state of the art Forecasting algorithms

In this section, we present the comparative evaluation we performed on a series of forecasting algorithms, on the three different consumption datasets, described in Sections 3.2.1, 3.2.2 and 3.2.3. In our experiments, we assessed the forecasting precision of each algorithm on each of the three datasets. Specifically, depending of the dataset, in this evaluation we consider two forecasting settings:

(a) forecast the hourly consumption of the next day (when time-series of consumptions are available – historical SWM data), and

(b) forecast the consumption of the next consumption event (when sequences of consumption events without timestamps are available – Amphiro historical shower data).

Further, we implemented two first-cut algorithms that attempt to capture distinct consumption patterns and exploit them for forecasting. We included these two proposed algorithms in the experimental evaluation, showcasing that a method that considers consumption patterns can outperform state of the art forecasting algorithms on water consumption data.

This study did not aim to perform an exhaustive evaluation and comparison on every available method in the literature, since this is out of the scope of our work. Contrary, we aimed on obtaining mostly qualitative findings on the behavior of the algorithms and their shortcomings when executed on water consumption data. This was a crucial step before implementing our core work, described in Sections 3.4 and 3.5.

Next, we briefly describe the state of the art forecasting algorithms we evaluated, along with two basic, first-cut methods we proposed, based on our intuitions on the specificities of the data. Then, we present the three experiments we performed, measuring and comparing the next hour forecasting precision of each algorithm, on each of the three datasets. Finally, we sum up our findings and insights gained through this process. This work is also presented in [CGA16] and the code for the performed evaluation is available on GitHub[2].

[2] https://github.com/DAIAD/home-web/tree/master/jobs/study

3.3.1. State of the art algorithms

The evaluated state of the art forecasting algorithms are described next. Most of them were adopted from existing frameworks/libraries, while some were slightly adapted to fit our setting.

- **Linear (Auto-)Regression (LAR)**. The idea of LAR is to model the consumption to be predicted as a linear function of the previous consumption values of the time-series. This is achieved, in general, by trying to find the optimal linear hyperplane that minimizes the distance from the observed data points.

  In our evaluation, in order to limit the effects of noise (overfitting) that is always present in the data, we use *ridge regression* [Bis09], which aims at minimizing the coefficients of the model.

  For the implementation of the algorithm we used the linear algebra operations available in the EJML[3] Java library.

- **Support Vector Machines (SVM)**. Support Vector Machines, in its simplest form is an algorithm for binary linear classification [Bis09]. Instances are analyzed into features and represented in a multidimensional feature space. There, the algorithm tries to find a hyperplane that best separates two classes of instances by maximizing the distance of the nearest instances of each class to the separating hyperplane.

  In the specific setting, we use as features the previous consumption values of the time-series and take as output a discretized consumption prediction (ranges of consumption).

  The algorithm used was available in the LIBSVM[4] Java library.

- **Support Vector Regression (SVR)**. Support Vector Regression is a version of SVM used for regression [SS98]. However, instead of searching for a linear hyperplane that best separates instances of two classes, it aims at finding the simplest hyperplane so that all the points stay within a specified distance from it. Intuitively, in two dimensions, it would try to find the simplest (less affected by changes) line so that all the points stay within a tube around the line.

  In the specific setting, we use as features the previous consumption values and take as output the prediction for the next consumption value.

  The algorithm used was available in the LIBSVM[4] library.

- **Autoregressive Integrated Moving Average (ARIMA)**. Autoregressive Integrated Moving Average [Tay10] is a class of algorithms for time-series forecasting that model the current value as a linear combination of the previous values and the errors of the previous predictions. The seasonal ARIMA model incorporates seasonality information within the model, by taking into account values from the previous seasonal cycles. This model is expected to perform well on datasets that are dominated by seasonal patterns.

  The algorithm used was available in the forecast[5] package of R language.

- **Exponential Smoothing (Exp smooth)**. Exponential Smoothing is another class of algorithms used for time-series forecasting [Gar06]. It composes the time-series from three parts: level, trend and

[3] http://ejml.org/wiki/

[4] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[5] https://cran.r-project.org/web/packages/forecast/

seasonality. The level is the part of the time-series that is considered locally constant, the trend is the expected increase or decrease between subsequent observations and the seasonality is the dependence from the value of a specific previous moment (e.g. same hour previous day). Each part is estimated as a weighted average based on the previous observations of the time-series, with weights that fade exponentially as they move to more distant observations.

The algorithm used was available in the forecast[5] package of R.

- Artificial Neural Network (ANN). Artificial Neural Networks are composed of layers of nodes, at each of which the output is a linear combination of the inputs passed through an activation function [Bis09]. The first layer takes as input the explanatory variables. Each layer feeds the next one and the last layer gives the dependent variable. The number of layers, the number the nodes at each layer, the activation function and the algorithm used for training are parameters of the model. The most popular training algorithm is backpropagation. In backpropagation, starting from the output, the error is propagated backwards at each node according to the partial derivative of the error function and the coefficients are adjusted to minimize the error.

  The algorithm used was available in the neuralnet[6] package of R.

3.3.2. Proposed algorithms

The following two algorithms were proposed by us, as *two basic/first-cut versions* based on our intuitions of handling the forecasting problem using discrete historical patterns identified in the consumption time-series.

Sequential Regression (SR). By considering the characteristics of the aforementioned state of art algorithms and the available water consumption datasets, we attempted to develop a first-cut algorithm for water short term consumption forecasting (either on time-series or on sequences of consumption events).

In Figure 14 below, we see an example of a sequence of shower water consumptions. We can observe the existence of three clusters/classes of water consumption (of high, medium and low consumption as denoted by the horizontal lines). The classes are identified by applying a k-means clustering algorithm [Bis09] on the volumes of all consumption events. Each class may represent a different type of consumption, perhaps taken by a different person in the household or for a different occasion. Further, we can see the existence of some patterns in the sequence. For example, a consumption of the higher class is never followed by another one of the same class.

[6] https://cran.r-project.org/web/packages/neuralnet/

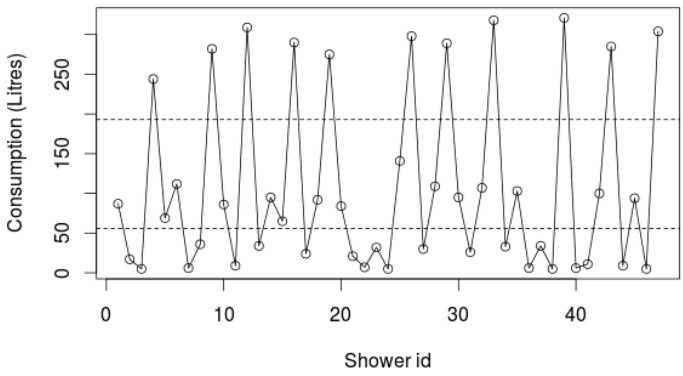

Figure 14: Example of shower water consumption sequence

Motivated by the above observations, we proposed an algorithm that discretizes the consumption and then models the patterns in the subsequences of the discretized sequence. The problem is defined as follows: first discretize the data and then, given the classes of the previous b consumption events, predict the class of the next consumption event along with an exact prediction value. Intuitively, the symbol (class) that has occurred most times, given the b previous symbols, is the most likely to occur again. To calculate this, we count the occurrence of each subsequence of size $b + 1$.

More formally, we define a mapping data structure C. C is indexed by a subsequence of size $b + 1$ and returns an integer. The elements of C count the occurrences of subsequences. We count all the subsequences in the training set by scanning the time-series and provide as prediction the value that corresponds to the most probable of the available subsequences. For subsequences that have never been observed before we give as prediction the most probable symbol of the training set. SR is a complex, non-parametric, model, that can learn arbitrary patterns but requires a large sample to be trained from.

Classed Linear Regression (CLR). Based on the same discretization assumption that we made in the case of SR, we also performed linear regression on the discrete sequence of the consumptions. In particular, we consider the classes of the previous b consumptions as explanatory variables in the linear model. To indicate the classes, we use indicative variable vectors, i.e. vectors where each dimension corresponds to a class, the dimension of the current class is 1 and all other dimensions are 0. Then we apply ridge regression as described previously. CLR is a simpler (linear) model, which can be trained from a smaller sample.

3.3.3. Evaluation setting

In our experiments, we aim at measuring the precision of each algorithm in the setting of using historical time-series data in order to forecast the *hourly consumption of the next day*. Especially for the Amphiro historical dataset, the setting adapts into forecasting the *next event's (shower) consumption*. Next, we describe the evaluation measures used, the naïve baselines we considered for comparison with the assessed algorithms and the parameterization of the algorithms.

3.3.3.1. Evaluation measures

As an evaluation measure, we consider Mean Average Percentage Error (MAPE), a widely used measure for assessing the prediction performance of forecasting algorithms. Given a time-series y of size n, with y_i a measured value and y'_i the respective predicted value in time i, MAPE is defined as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - y_i^{'}|}{y_i}$$

Especially for the setting of individual household consumptions, where it is often the case that a household has zero consumption in some hours, MAPE measure is problematic, since the denominator y'_i becomes zero. Thus, for the specific setting, we use a variation, the Normalized Mean Absolute Error (NMAE), which is defined as follows:

$$NMAE = \frac{1}{n}\frac{1}{mean(y)}\sum_{i=1}^{n}|y_i - y_i^{'}|$$

where *mean(y)* is the average value of all y_i.

3.3.3.2. Baselines

As baselines, we consider the following: (i) **BR0** which provides as prediction the median of the time-series; (ii) **BR1**, which provides the value of the previous hour; and (iii) **BR2**, which provides the value of the same time from the previous day.

3.3.3.3. Parameterization

Next, we present the parameterization of the algorithms.

- Parameter *b* controls how many of the previous measurements are used as features to predict the next water consumption. It is used in all algorithms except for Exponential Smoothing. A large *b* gives more information to the algorithm but requires significantly larger sample for the training of the model, depending on the specific model. For the Amphiro Shower dataset, we searched for b in range [1, 9] with step 1. For the SWM individual and aggregate datasets, we searched for b in the values (1,24,48,168).
- Parameter *k* determines the number of classes for the algorithms that discretize the time-series: SR0, CLR and SVM. We searched for *k* in range [1,48] with step 1, for all datasets.
- SVM and SVR models use parameters *C* and SVR additionally uses *epsilon*. *C* controls the regularization of the model and *epsilon* controls the width of the error insensitive tube. Also for SVM and SVR, a choice of kernel must be made. The kernel controls the transformation of the data before the model is fitted. We set the parameters for the SVM and SVR empirically. For the shower consumption dataset, we used the polynomial kernel with *C*=0.5 and, for the SVR, *epsilon*=0.001. For the individual household consumption, we used the linear kernel with *C*=0.1 and *epsilon*=0.0001. For the aggregate consumption dataset, we used the polynomial kernel with *C*=0.5 and *e*=0.001.
- For ANN, the choice of hidden layers and nodes per layer controls the complexity of the model. We empirically selected: 3 hidden layers of size 5, 3 and 2 respectively for the shower dataset and 2 hidden layers of size 10 and 5 for the individual household and aggregate datasets.
- For ARIMA the configuration was $(3,0,3)(2,0,2)_{24}$ (in ARIMA notation) for both individual household and aggregate datasets and Exponential Smoothing was used with daily and weekly seasonality (24 and 168 hours respectively).

3.3.4. Evaluation results

Next, we present the evaluation results on the forecasting precision of the state of the art algorithms (presented in Section 3.3.1) and two proposed, first-cut approaches (presented in 3.3.2) executed on the three datasets presented in (Section 3.2 and Annex 6).

3.3.4.1. Amphiro Shower Data

Figure 15 presents the MAPE values of each evaluated algorithm on the Amphiro historical dataset, where only shower events without timestamps are available. We can see that only the Support Vectors-based methods (**SVM, SVR**) and the **Sequential Regression** method we propose achieve better precision than the naïve baseline BR0.

The best performance is that of SVR, for which the polynomial kernel was used. This, along with the poor performance of LAR, is consistent with the fact that the linear autocorrelation of the time-series is low, which means that there is no linear pattern in the data. SR0 and SVM perform a little worse than SVR but are on the same level. The performance of CLR is better than simple linear regression but not as good as SR0. This suggests that there is improvement due to the discretization but the linear model is too limiting to describe the time-series. Exp. Smooth and ARIMA, which are usually used for water/energy time-series forecasting, but not of this type of data (missing timestamps), perform particularly poorly as does the ANN too. We note that they perform worse than the baseline which treats the dataset as random and always predicts the median value.

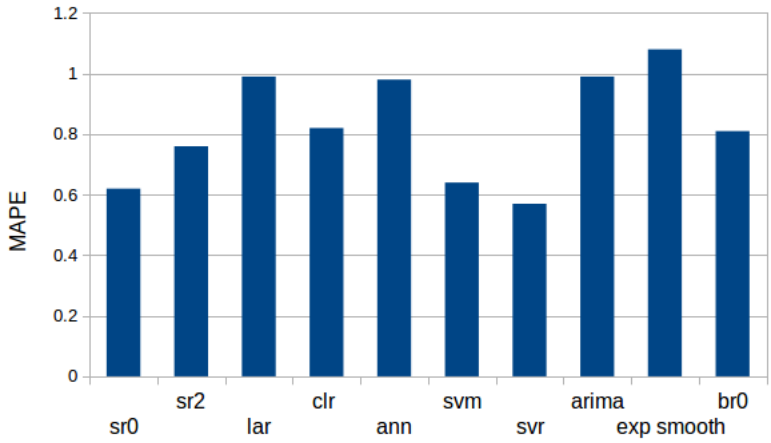

Figure 15: Forecasting performance on Amphiro Showers dataset

3.3.4.2. Individual historical SWM data

Due to the large parameterization space and the large run-times of some of the evaluated algorithms (Exp Smooth, ARIMA, ANN)), out of the 1000 households, we focused on an initial set of time-series for only 121 households, thus, we report our experiments on them. As stated before, each time-series consisted of hourly measurements of water consumption for a period of one year. In this experiment, we applied the forecasting algorithms on each individual time-series separately and we aggregated the forecasting errors on all time-series.

In Figure 16 we can see the performance of the algorithms in terms of NMAE. The error is *above 80%* for all algorithms except **SVR**, which is a poor performance. This can be partly justified by the fact that the mean

value of the time-series, which is at the denominator of the metric, is low and that results to big values for the relative error. Nevertheless, the performance compared to the baselines is disappointing. The best performance comes from SVR and is only 20% better than that of BR1, with SR0 being between them. All other algorithms have *worse performance* than BR1. This is probably due to the fact that the time-series do not follow some simple analytical model. However, there may be patterns that can be captured by a more complex, data-driven analysis, like that of SR0.

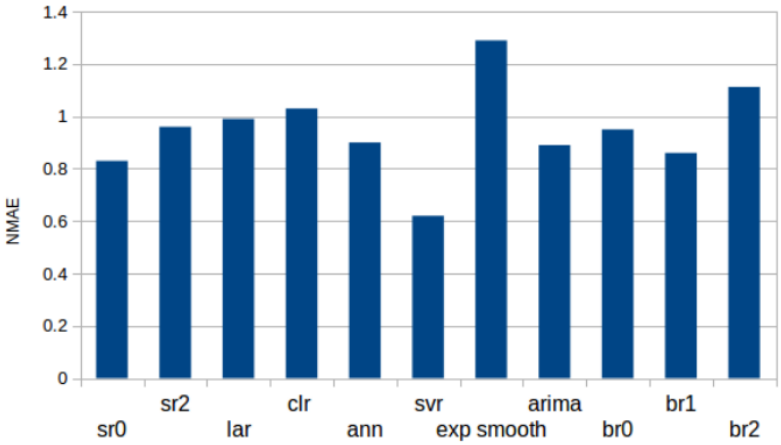

Figure 16: Forecasting performance on individual consumption time-series

The above indicate a possible direction for improvement of forecasting on the specific dataset. Providing a model that relies on *past consumption patterns* (rather than plain consumption values) can lead to a better performing algorithm. Moreover, trying to identify specific recurring patterns by inspecting the time-series seems to be more suitable for this kind of data, than trying to identify an analytical model. These findings are realized in the Pattern Forecasting model we implemented, which is described in Section 3.4.

Finally, motivated by the low precision of the evaluated algorithms, and by examining the distributions of the time-series through time and the precision of training forecasting models on different parts of the historical time-series data, we observed that the model of several time-series *changes through time*, something which negatively affects the performance of the algorithms, since the training of the model remains fixed. These findings motivated us into implementing an algorithm (Iterative Training Method) that identifies changes in the model of the time-series and exploits them to increase the forecasting precision, as presented in Section 3.5.

3.3.4.3. Aggregate historical SWM data

For the third experiment, we aggregated the consumption time-series from all households into one time-series and evaluated consumption forecasting on this single, aggregate time-series.

In Figure 17, we can see the performance of the algorithms on the aggregate data. The performance is relatively good and much better than in the previous setting. This was expected because of the very strong seasonal patterns of the time-series. The best performance comes from the ANN. SVR and LAR are the second and third best approaches, respectively. The good performance of LAR was expected because of the

high linear autocorrelation of the time-series. However, we would expect more sophisticated algorithms like ARIMA and Exp Smooth to outperform LAR which did not happen. This could be attributed to the relatively high noise of the time-series, which is explained by the fact that it aggregates only 121 individual consumption time-series. In the literature, the errors reported for the same algorithms on energy consumption datasets for one step ahead prediction are even lower (< 5%). The datasets in those cases are aggregates of much larger samples and, thus, even more regular. Also the time-series of energy consumption tend to be more canonical.

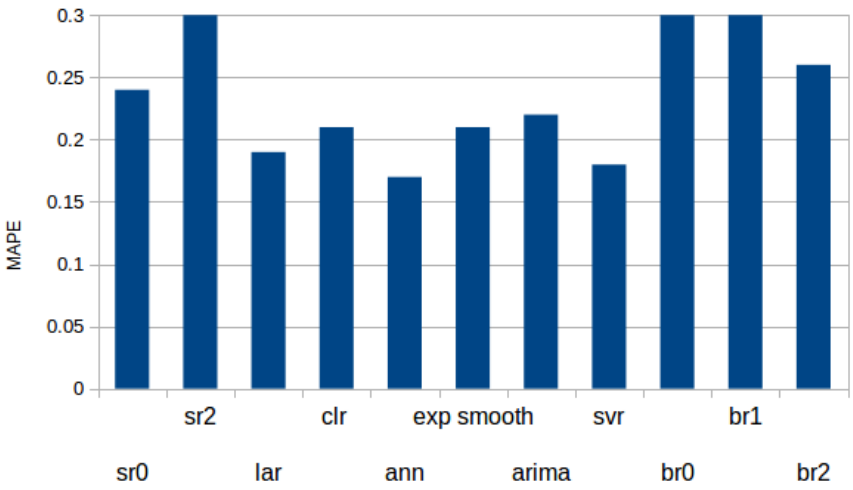

Figure 17: Forecasting performance on aggregate consumption time-series

3.3.5. Insights and discussion

From the analysis of the data and the evaluation of the results we reached several conclusions and insights that gave us the motivation for the algorithms we developed (in Sections 3.4 and 3.5).

- The performance of state of the art forecasting models is (as expected) good for aggregate water consumption time-series. However, it is lacking for forecasting shower consumption events and water consumption time-series for an individual household.
- The main challenge on analyzing the individual historical SWM dataset is the existence of high noise, in the volume of water consumption, as well as in the time of water consumption (due to time-shifts of consumption events through different days). This makes the modeling of the time-series difficult. To address this, we propose to move the modelling to a higher level of abstraction, that of human activity inside the day. In order to achieve that, we propose to capture discrete patterns of activity inside the day (i.e. the morning activity, the afternoon activity etc.). Those patterns may define qualitatively different kinds of days and provide information that facilitates the forecasting of the time-series. To this end, we implement an algorithm that jointly identifies patterns and forecasts consumptions, the Pattern Forecasting algorithm described in Section 3.4.
- Another challenge identified in the individual household SWM consumption data is the changes in the underlying model of the time-series. These may be caused due to abrupt changes in the schedule of the household members and influence the performance of the forecasting models. In order to improve the forecasting, we need to identify those changes and modify the models

accordingly. To this end, we implement the Iterative Training Method for models change detection and forecasting, described in Section 3.5.

3.4. Pattern Forecasting (PF)

In this section, we present the model we have developed for *jointly identifying consumption patterns and forecasting consumption*, on water consumption time-series of individual households. With respect to forecasting, we focus on short term forecasting, i.e. forecasting the *hourly consumption of the next day*. The presented algorithm (Pattern Forecasting – PF) is the evolution of the first-cut methods that were proposed in Section 3.3.2 and evaluated in Section 3.3.3, namely SR and CLR methods. The idea of pattern forecasting is to *first identify* and analyze *recurring patterns* within several days and then try to *forecast* their appearance in the days to come. The method is based on identifying distinct *activity zones*, based on the consumption levels and the time of the day they occur. The code for the implemented algorithm is available on GitHub[7].

3.4.1. Pattern recognition

By studying the SWM time-series of individual households we have observed that there are certain parts of the day where patterns of water consumption appear, that may correspond to daily household activities. For example, in the Figure 18 below, we can see a household's water consumption of a day, with three periods of significant water consumption (highlighted in red), one early in the morning, one around noon and one in the afternoon.

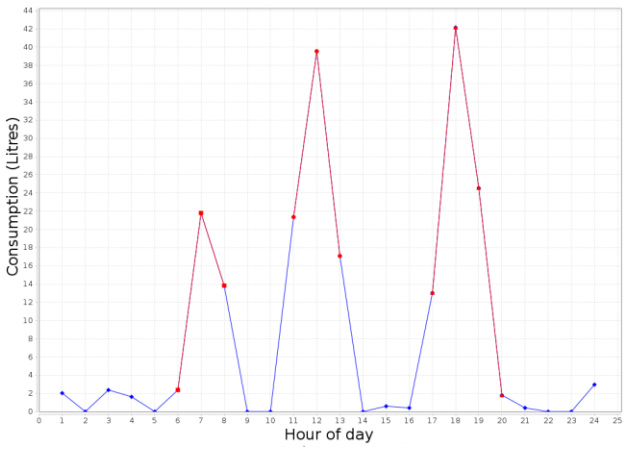

Figure 18: Example of daily consumption patterns

The above example represents a usual daily consumption pattern, recorded in many households. However, several other types of, less usual, daily patterns are also recorded, indicating divergences and variations in consumptions. In Figure 19, we can see several examples of days, from a single household, demonstrating different patterns.

[7] https://github.com/DAIAD/home-web/tree/master/jobs/pattern-forecasting

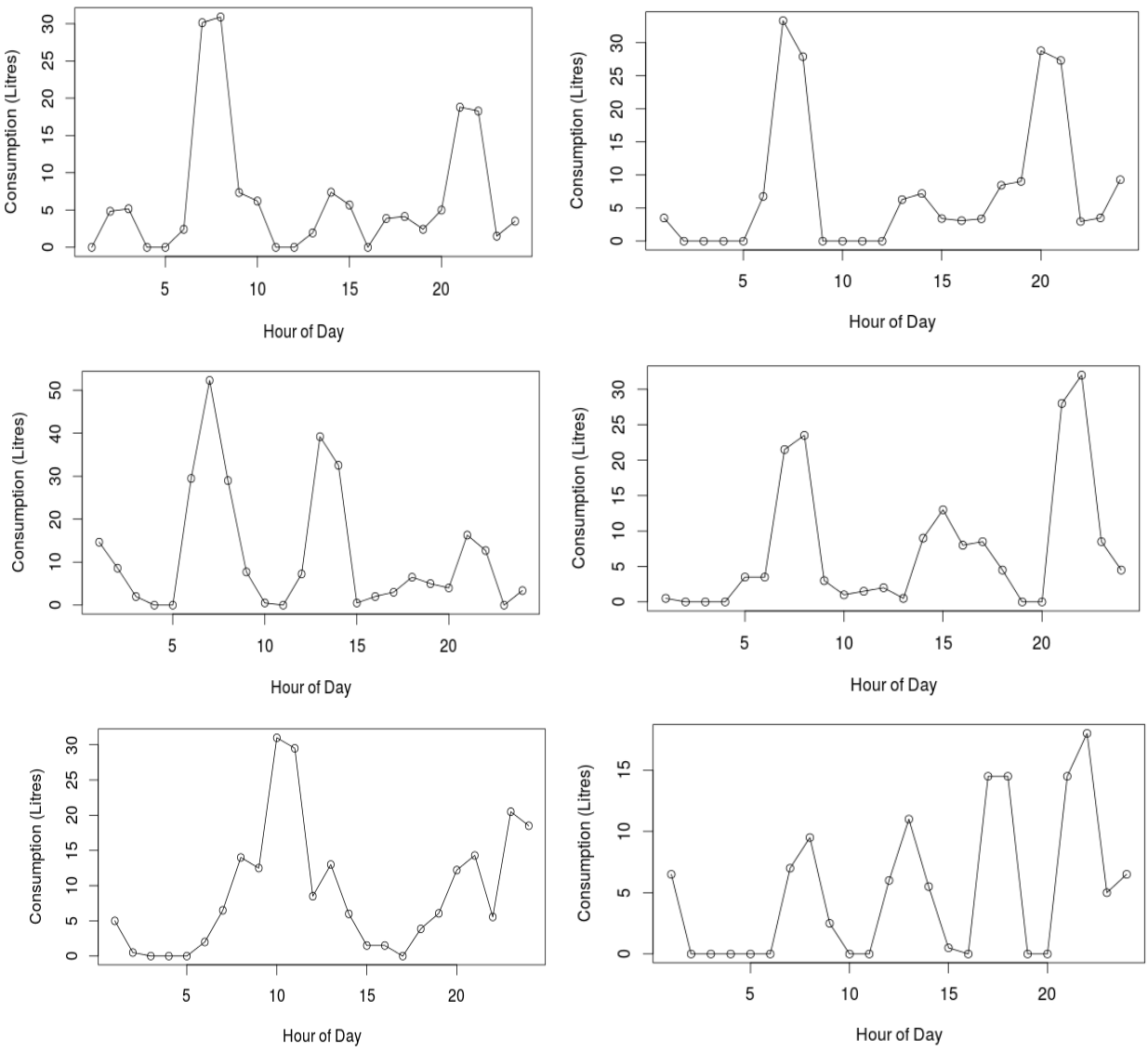

Figure 19: Different types of daily consumptions,, measured by SWMs

We observe that the consumption does not change for each hour independently. Instead, there are periods of high consumption that span several hours and usually appear at specific parts of the day. In some days there are two periods of high consumption (first row), while in others there are three (middle row). However, there are also days where consumption is more unusual (bottom row) and are not similar to other days. We make the hypothesis that different patterns represent days with discretely different activities for the household (i.e. a day off work, a rainy day etc.). There may also be recurring external events that influence consumption, like weather, sport events, market hours etc.

As we observed in the examples of the above figures, the consumption usually appears in formations with gradual changes that last a few hours. We call these formations patterns.

Specifically, we define as a *pattern,* a part of the time series of the daily consumption, which starts when the consumption exceeds a predefined threshold and ends when the consumption falls below the same threshold.

The occurrence of patterns does not happen randomly throughout the day, instead they usually appear at specific time intervals (e.g. 7:00-10:00 in the morning, 20:00-22:00 in the evening), which we call activity zones.

An *activity zone* is defined as an interval in the day where patterns appear for many days of the year (according to a selected threshold).

The exact way in which the patterns and the activity zones are calculated is described next, through Algorithm 1 for obtaining activity zones. As input we have the consumption measurements of each day (in our specific dataset case, 24 consumption measurements per day). We scan each day to detect the patterns. We consider that a pattern starts when the consumption exceeds a threshold **A** and ends when the consumption falls below said threshold. It is often the case that a new pattern starts before the current one has finished. In this case, if the consumption between two peaks has fallen at below a factor **B** of the consumption of the peaks, we break the pattern in two parts. This process corresponds to Algorithm 1, line 1.

Algorithm 1: Obtain Activity Zones

Data: TimeSeries Ts
Result: ActivityZones $Zones$
1 $Patterns$ = IdentifySignificantPatternsForEachDay(Ts)
2 N=IdentifyNumberOfPattersPerDay($Patterns$)
3 $Hours$ = CountPatternPeaksPerHourOfDay($Patterns$)
4 $Zones = \emptyset$
5 **for** i *from* 1 *to* N **do**
6 $Peak$=SelectHourWithMaxPatterns($Hours$)
7 $Zone$=FindZoneAboveThreshold($Hours$,$Peak$)
8 AddNewZone($Zones$,$Zone$)
9 RemoveZoneFromHours($Hours$)
10 **end**

Figure 20: Pattern Forecasting - Algorithm 1

Each pattern is characterized by its total consumption. Also, for each pattern, a *center* value is defined as the weighted average of all the hours of the duration of the pattern, with the hourly consumption as a weight.

Next, we identify the activity zones. To do so, we consider only the patterns that make up for more than **F%** of the daily consumption. We count the number of patterns of each day (Algorithm 1, line 2). We consider the maximum number of patterns per day **C** as the minimum integer for which **D %** of the days have more patterns. Thus, we exclude possible outliers that may have many patterns.

Then, we count the pattern centers that occurred for each hour of the day across all days (Algorithm 1, line 3). We also consider a threshold **E**. We start at the hour with the most centers (Algorithm 1, line 6) and move to next and previous hours until the occurring centers fall below **E** (Algorithm 1, line 7). This is considered an activity zone. After identifying an activity zone, we store it and remove it from the data in order to identify the next (Algorithm 1, lines 8,9). We repeat **C** times.

After those steps are performed, we have gathered a set of activity zones $A = \{a_z, 1 \leq z \leq C\}$. Each activity zone is characterized by its staring time $start_z$ and its ending time end_z. Also for each activity zone we define a threshold t_z that we use to evaluate whether there is significant water consumption in the

activity zone. In order to calculate t_z we estimate the distribution of the value of total water consumption in a_z and then search for two discrete peaks in the distribution, one near 0 and one in some positive value. Then, we set as t_z the point where the distribution is the lowest between those two peaks (Algorithm 2, line 1).

Algorithm 2: Train Model for Activity Zones Prediction

Data: TimeSeries Ts, ActivityZones $Zones$
Result: ModelForActivityZones $Model$
1 $Thresh$ = FindThresholdsForActivity(Ts,$Zones$)
2 $ActTs$ = ObtainActivityTimeSeries($Thresh$,Ts)
3 $Model$ = InitEmptyModel()
4 **forall the** *ActivityZones* $Az \in Zones$ **do**
5 $ActModel$ = TrainModelForActivity($ActTs$,Az)
6 AddToModel($ActModel$,$Model$)
7 **end**

Figure 21: Pattern Forecasting - Algorithm 2

3.4.2. Forecasting

Forecasting is performed in two parts. First, an array of Support Vector Regression (SVR) models are used to obtain forecasts for every hour of the next day. Then a probabilistic model is used to predict the activity zones of the next day. Finally, the predictions of the SVR models are modified according to the predictions for the activity zones.

The SVR array contains 24 SVR models, one for each hour of the day. The features for each SVR are the 24 values for the hourly water consumption of the previous day and an indicator vector for the day of week. The target variable for each SVR_j is the consumption for time of day *j*. After training the SVRs using historical values we can obtain a forecast for the 24 hours of the next day.

In order to predict the activity zones for the next day we build a probabilistic model for the water consumption in the activity zones. Let each day i be represented by a vector $\mathbf{d_i}$ of size 24 with d_{ij} representing the water consumption for time j. We have a series of k days $\boldsymbol{D} = [\boldsymbol{d_1}, \boldsymbol{d_2} \dots \boldsymbol{d_k}]$. By using the set of activity zones we transform each day $\boldsymbol{d_i}$ to a vector $\boldsymbol{r_i}$ of size C, representing whether there is significant activity in each activity zone (Algorithm 2, Line 2). If the total consumption inside the activity zone a_z exceeds a threshold t_z, it is considered significant and $r_{iz} = 1$, else $r_{iz} = 0$:

$$r_{iz} = \begin{cases} 1, & \sum_{j=start_z}^{end_z} d_{ij} > t_z \\ 0, & \sum_{j=start_z}^{end_z} d_{ij} \leq t_z \end{cases}, \qquad 1 \leq z \leq C \qquad (1)$$

In this way we obtain series $\boldsymbol{R} = [\boldsymbol{r_1}, \boldsymbol{r_2}, \dots, \boldsymbol{r_k}]$. Then, using $\boldsymbol{R}$, we build a model that calculates the probability that there will be significant activity in each activity zone of day $i+1$ by given the state of the activity zones of day i (Algorithm 2, lines 5,6):

$$p\left(r_{(i+1)z} = 1 \middle| r_{i1}, r_{i2} \dots, r_{iC}\right) = \frac{p\left(r_{(i+1)z} = 1,\ r_{i1}, r_{i2} \dots, r_{iC}\right)}{p(r_{i1}, r_{i2} \dots, r_{iC})} \qquad (2)$$

We estimate the probabilities of the above fraction directly by counting the occurrences of the events in the dataset. Essentially we count, from the days that had similar behavior to day i, how frequently there was a significant activity in each activity zone in the next day.

Because we have a relatively small sample of 365 days, in order to avoid the problem of dimensionality, we also calculate the probability of Equation (2) by assuming that the states of the activity zones are conditionally independent, which results to the Naïve Bayes classifier [NJ02]:

$$p(r_{(i+1)z} = 1 | r_{i1}, r_{i2} \dots, r_{iC}) = \frac{p(r_{(i+1)z} = 1) \prod_{j=1}^{C} p(r_{(i+1)z} = 1 | r_{ij})}{\prod_{j=1}^{C} p(r_{ij})} \quad (3)$$

Again we calculate the conditional probabilities by counting the occurrences in the dataset. Then if $p(r_{(i+1)z} = 1 | r_{i1}, r_{i2} \dots, r_{iC}) > 0.5$, we predict that $r_{(i+1)z} = 1$, else $r_{(i+1)z} = 0$ (Algorithm 3, line 1). After we have a prediction for $r_{(i+1)z}$ we modify the prediction of the SVRs for the hours from $start_z$ to end_z to match this prediction (Algorithm 3, line 2).

Algorithm 3: Forecast Water Consumption

Data: BaselineSvrForecast $SvrPred$, TimeSeries Ts, ModelForActivityZones $Model$
Result: Prediction $Pred$
1 $AzPred$ = ForecastActivityZones($Model$,Ts)
2 $Pred$ = ModifySvrPrediction($SvrPred$,$AzPred$)

Figure 22: Pattern Forecasting - Algorithm 3

Specifically, from the values $\boldsymbol{d_{i+1}}$ that the SVR predicts we calculate if there is significant activity in each activity zone, in the way described in Equation (1). Then, for each activity zone z, if our probabilistic model predicts that $r_{(i+1)z} = 0$ and the SVR predicts $r_{(i+1)z} = 1$ we set $d_{(i+1)j} = 0$, $start_z \leq j \leq end_z$.

3.4.3. Evaluation

We evaluate the results of Pattern Forecasting algorithm in two aspects: (a) the accuracy of predicting whether there will *be significant activity in next day's activity zones* and (b) the NMAE of the *next day's hourly time-series forecast*. Accuracy is a classification metric which measures how many of the instances are correctly classified, as a fraction of the total instances. For our evaluation, we used the individual household SWM dataset as described in Section 3.2.2, which contained 800 transformed and cleaned time-series. We used the first 240 days of the year as training set and the following 125 as testing set. The training features used for each SVR was the hourly consumption of the previous day and the day of week.

As far as parameterization is concerned, we experimented as follows.

- For the SVR array we performed a different optimization for each household. The range of the search was $[10^{-6}, 10^{3}]$ with multiplicative step of 10 for *C* and *epsilon*. We experimented with

linear, polynomial and RBF kernels. We do not present the exhaustive list of parameters combinations, since they consist of approximately 2000 parameterizations, which are stored in files.

- The Pattern Forecasting algorithm has several parameters that are used in order to identify the patterns and the activity zones. Below, we present the configuration for each parameter, using the notation of Section 3.4.1.

Parameter	Value
A	10% * mean consumption of the day
B	50% * peak consumption of the pattern
F	8%
D	0%
E	Mean value of pattern centres that occur per hour

Table 1: Pattern Forecasting Parameterization

The results, shown in Figure 23, are the average results for all examined time-series:

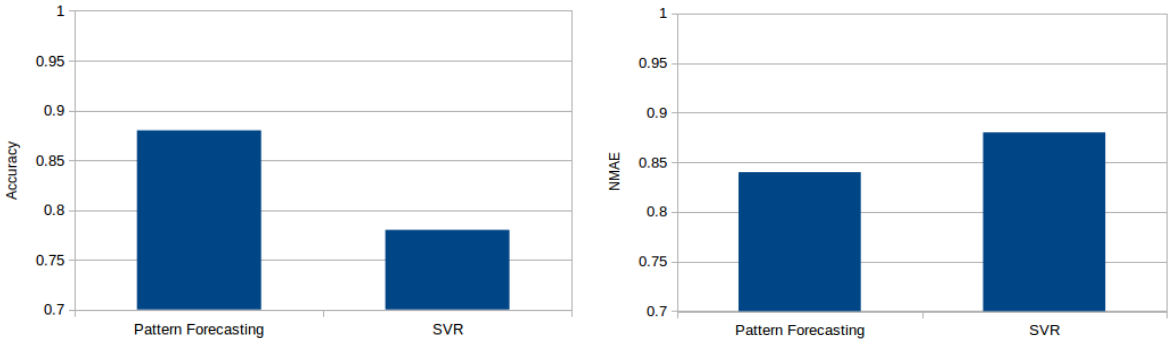

Figure 23: Accuracy (left) and NMAE (right) comparison

We can see that Pattern Forecasting provides a 10% increase in Accuracy of predicting the activity zones (88% over 78%) compared to the SVR. The decrease in NMAE is also considerable (84% over 88%). Both results are statistically significant, with p-values below the 0.01 threshold, measured by a paired sample permutation test.

From the above results we can conclude that there are, in fact, discrete patterns in the time-series that our algorithm captures. By forecasting those discrete patterns, we have been able to significantly improve the time-series forecasting. However, the improvement in terms of accuracy seems greater that the gain in terms of continuous value forecast. This suggests that there is probably room for improvement in the process that transforms the discrete forecast of the PF algorithm into continuous value forecast. Another direction for improvement for the PF algorithm is to implement additional evaluation methods for the activity zone identification process, as well as identify more than two (active-inactive) states for each activity zone.

3.5. Iterative Training Method (ITM)

3.5.1. Introduction

By experimenting on the individual household SWM consumption dataset we have observed that the behavior of the time-series may change through time. Those changes can be due to seasonal effects that happen smoothly throughout the year and can be predicted by state of the art models (e.g. ARIMA, Exponential Smoothing). However, they can also be random unpredictable changes, either smooth or rapid, that are due to changes in the behavior of the household members. For example, a change in work schedule, a trip, a guest arriving etc.

While several state of the art methods (e.g. change-point detection approaches) examine the distribution of the time-series [Gus00] or the training residual errors of an algorithm [OL10] to identify changes and relate them to the change of the model, we claim that this is not always sufficient. Take, for example, the two graphs of Figure 24. The top graph presents the distribution of the water consumption of a household for one year. We can identify a slightly increasing trend, as well as a few outlier consumptions. The bottom graph presents the correlation of next day's consumption with the previous day's consumption at 7:00 am. This graph informs us that, for different periods through the year, the dependence of next day's consumption to previous consumptions (in this case previous day's at 7:00am) changes potentially significantly. These changes in the model of the consumption behavior are not reflected on the top graph (distribution of the time-series). This indicates that, in a setting where we want to forecast the next values of the time-series, if we include all the historical observations into the training set of the forecasting model, changes in the behavior of the time series will, generally, increase the forecasting error of the model. Thus, we argue that there is a point in time prior to which including samples in the training set of the model leads to an increase in the error. We define this point as a *point of change*.

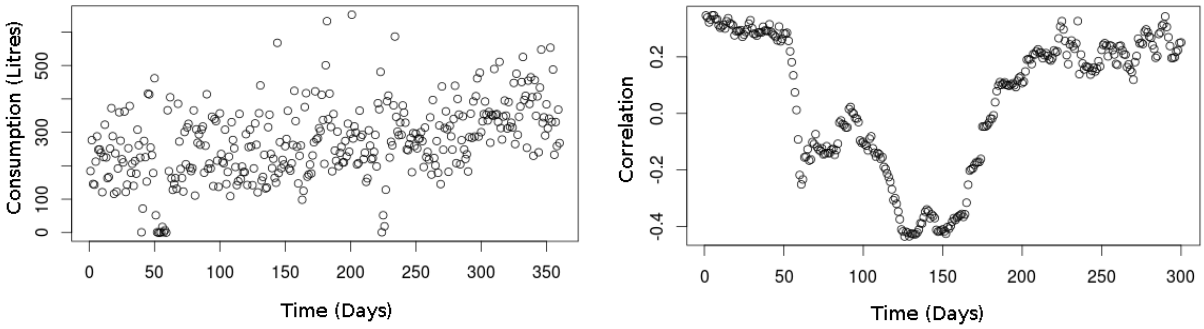

Figure 24: (left) Water consumption of a household through a year
(right) Correlation between consumption at 7:00 am and next day's consumption, for the same household

As stated above, unpredictable changes influence the models used for time-series forecasting and can hinder their performance. This issue is usually addressed by partitioning the time-series, using change-point detection [HD15] or time-series segmentation [GS99] algorithms, and fitting the machine learning model in each part separately. The first problem with several of these approaches (as stated in the example above) is that they only consider changes in the distribution of the time-series and not on the machine learning model that is trained on it. A second shortcoming of existing methods is that they *do not address the generalization error of the model that is going to be fitted*, because they do not evaluate the performance of

the model out of its training sample. This means they do not take into account all the effects that the change in the time-series has to the forecasting error of the model.

To address this issue, we developed an algorithm called **Iterative Training Method (ITM)** that partitions the time-series to the *point of change* that is optimal in terms of generalization error of the model. The algorithm achieves this through the use of an iterative training process and a validation set.

3.5.2. The ITM algorithm

In order to identify the optimal point of change, the algorithm considers a training dataset of past time-series observations and a machine learning model that is to be trained on the dataset. It starts from a recent point in time, t_n and scans the time-series backwards in order to identify the historical point, t_h where a significant change in the time-series is detected. The change is detected by examining the variations in the error of the model, while adding previous samples in the training set. The algorithm then isolates the sub-time-series $t_h, \ldots, t_n$ and provides only this specific part as training input for the machine learning model to be trained. Next we describe the algorithm in more detail.

We consider time-series Y, of water consumption, for which we have measurements up to the current time t: $Y = \{ y_i, 1 \leq i \leq t \}$, where i is the index of time. For each y_i we have a vector of associated features $\boldsymbol{x_i}$. We use as training features the water consumption of the previous 24 hours and the day of week. From $\boldsymbol{x_i}$ and $\boldsymbol{y_i}$ we form the tuple $(\boldsymbol{x_i}, y_i)$ that we denote $(\boldsymbol{x}, y)_i$. Then, we have a set of observed tuples:

$$D = \{ (\boldsymbol{x}, y)_i, 1 \leq i \leq t\}$$

which is our data-set. From the data-set we select a validation set $D_v = \{ (\boldsymbol{x}, y)_i, t - n_v < i \leq t \}$, of size n_v . We also select an initial training set $D_{train} = \{(\boldsymbol{x}, y)_i, t - n_v - n_t < i \leq t - n_v \}$, of size n_t. Then we iteratively perform the following process (Algorithm 4): Add a previous data-point on D_{train} ; Train a model on D_{train}; Calculate the error on D_v.

Algorithm 4: Obtain sequential errors

Data: dataset D, parameters n_v, n_t
Result: sequences of error $\boldsymbol{E}, \boldsymbol{E_\mu}$

$D_{train} = \{(\boldsymbol{x}, y)_i, t - n_t - n_v < i \leq t - n_v\}$
$D_{val} = \{(\boldsymbol{x}, y)_i, t - n_v < i \leq t\}$
for i *from* $t - n_v - n_t$ *to* 1 **do**
  $D_{train} \leftarrow D_{train} \cup (\boldsymbol{x}, y)_i$
  $\boldsymbol{\theta_i^*} \leftarrow TrainModel(D_{train})$
  $\boldsymbol{e_i} \leftarrow MeanAbsoluteError(\theta_i^*, D_{val})$
end

Figure 25: Model Change Detection - Algorithm 4

After this process is performed, we have a sequence of error vectors $\boldsymbol{E}$, and their mean values $\boldsymbol{E_\mu}$.

$$\boldsymbol{E} = [\boldsymbol{e_1}, \boldsymbol{e_2} \ldots, \boldsymbol{e_{t-n_v-n_t}}]$$

$$\boldsymbol{E_\mu} = [\bar{e}_1, \bar{e}_2 \ldots, \bar{e}_{t-n_v-n_t}]$$

Using the sequences of errors $\boldsymbol{E}$ and $\boldsymbol{E_\mu}$ we want to identify a point with significant improvement in the error, if such a point exists. The improvement is considered in comparison to $\boldsymbol{e_1}$ which corresponds to using the entire dataset for the training of the model. The significance of the improvement is measured using a

statistical significance test. In our implementation, we use the Student's t-Test [BC02] (Algorithm 5, line 6) or, alternatively, an empirical test, that we developed by experimenting with the individual household consumption dataset. In the empirical test, we consider the improvement over $\boldsymbol{e_1}$ significant, if it is above a certain percentage (e.g. 5%) of $\overline{e_1}$.

Algorithm 5: Find optimal point

Data: sequence of errors $\boldsymbol{E}, \boldsymbol{E_\mu}$, parameters w, α
Result: optimal point $index$

1 $min_e \leftarrow E_\mu[1]$
2 $index \leftarrow 1$
3 **for** i *from 1 to* $t - n_t - n_v - w + 1$ **do**
4 Let j be the index of the median of the set $\{E_\mu[l], i \leq l < i + w\}$
5 $\boldsymbol{d_j} = \boldsymbol{e_1} - \boldsymbol{e_j}$
6 **if** $Student's\ t\ Test(\boldsymbol{d_j}, \alpha)$ *and* $E_\mu[j] < min_e$ **then**
7 $index \leftarrow i$
8 $min_e = E_\mu[j]$
9 **end**
10 **end**

Figure 26: Model Change Detection – Algorithm 5

The threshold of statistical significance in the tests is controlled by parameter $\boldsymbol{\alpha}$. In order to avoid false positive results, that are known to occur in cases of multiple statistical significance tests, we scan $\boldsymbol{E_\mu}$ using a sliding window of length $\boldsymbol{w}$ and, each time, compare only the point with the median error inside the sliding window to $\boldsymbol{e_1}$ (Algorithm 5, line 4). This way we avoid outliers and spurious results. If several points with statistically significant improvement over $\boldsymbol{e_1}$ are found, we select the one with the minimum error. After identifying the point with the optimal error we select the start of the training set that this error corresponds to as the optimal point of change.

Our algorithm works on top of several machine learning models and optimizes their training process, taking into account the forecasting error of the model. The models that we have used in our implementations are the Support Vector Regression model, the Elastic-Net regularized Linear model [ZH05] and the Artificial Neural Network model. Our work on this problem is also presented in a manuscript under review for an international conference [CGS16] and the code for the implemented algorithm is available on GitHub[8].

3.5.3. Evaluation

We evaluate our algorithm, ITM, assessing its precision in forecasting the next day's aggregate consumption. For our evaluation, we used the individual historical SWM dataset as described in Section 3.2.2, which contained 800 transformed and cleaned time-series. The training features used for each SVR was the hourly consumption of the previous day and the day of week.

We consider three baselines.

- Baseline 1 (BL1) uses the entire dataset for training the model.

[8] https://github.com/DAIAD/home-web/tree/master/jobs/model-change

- Baseline 2 (BL2) identifies changes in the time-series by using a change-point detection algorithm based on the Mann-Whitney statistic [HD10].
- Baseline 3 (BL3) identifies changes in the model by using the same change-point detection algorithm on the training residual errors of the forecasting model.

We evaluate ITM for 3 different forecasting models:

- Elastic-Net regularized linear model (ENET)
- Support Vector Regression (SVR)
- Artificial Neural Networks (ANN).

The metric that we use for the evaluation is the improvement of Mean Absolute Error (MAE) of the forecast for the next 10 days, over BL1, as a percentage. We compare each algorithm to BL1 because BL1 is the simplest and most usual approach. MAE is defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i'|$$

We examined the following values for the several parameters of the algorithms.

- For SVR we used the linear kernel, the heuristic function implemented in the LIBLINEAR[9] package of R for the *C* parameter and a default value of 0.01 for the *epsilon* parameter. The *C* parameter was set for each time-series separately.
- The ENET algorithm has two parameters that control its regularization: $\lambda_1 and\ \lambda_2$. Those were set using cross validation, on each time-series separately.
- The ANN was empirically tuned to 1 hidden layer consisting of 5 nodes.
- For ITM the parameters that need to be set are the size of the validation set n_v, the size of the initial training set n_t, the size of the sliding window w and the confidence threshold α . There is also a choice between the two criteria for identifying a significant change, the statistical (STAT) and the empirical (EMP). We searched in the values (7,15,20,25), (13,15,20,25), (10,20), (0.01,0.05,0.1,0.2,0.3) for n_v, n_t, w and α respectively. The configuration that was selected for each model is shown below.

Model	n_v	n_t	w	α	Criterion for significant change
SVR	25	15	20	0.2	EMP
ENET	20	20	20	0.3	EMP
ANN	7	13	10	0.05	STAT

Table 2: Model Change Detection Parameterization

In order to better simulate the setting of an evolving time-series, we applied the algorithms at several different parts of each time-series, i.e. points in time. Each column of the tables below demonstrates the average improvement, over all time-series of the dataset, when the algorithms were applied at the

[9] http://www.inside-r.org/packages/cran/LiblineaR/docs/heuristicC

respective day of year, specified in the top row. We evaluate our algorithm with both alternative methods that we developed for change significance evaluation: the statistical, denoted by the offset "stat", and the empirical denoted by the offset "emp". Bold values denote the best achieving method for the specific day.

Day	80	120	160	200	240	280	320	340	Avg
ITM-stat	**1.6**	2.2	**5.4**	5.2	3.4	**2.5**	2.4	2.7	3.2
ITM-emp	0.5	**2.5**	5.3	**5.9**	**5.0**	0.9	**4.4**	**3.4**	**3.5**
BL2	0.0	2.3	2.6	2.2	2.2	0.4	2.3	-0.1	1.5
BL3	0.2	**2.5**	1.0	2.9	2.3	0.0	1.8	0.3	1.4

Table 3: % Improvement in MAE compared to BL1 in ENET algorithm

Day	80	120	160	200	240	280	320	340	Avg
ITM-stat	**0.3**	1.5	3.5	3.1	1.5	0.4	0.4	1.0	1.5
ITM-emp	0.0	1.8	**4.8**	**3.4**	**2.2**	**3.2**	**2.3**	**3.1**	**2.6**
BL2	-1.5	1.6	2.2	0.9	-0.4	-0.9	-0.8	-0.6	0.0
BL3	00	**2.3**	1.6	0.7	-0.9	2.0	-1.4	-2.1	-0.2

Table 4: % Improvement in MAE compared to BL1 in SVR algorithm

Day	80	120	160	200	240	280	320	340	Avg
ITM-stat	**1.4**	2.7	**2.8**	3.8	2.4	**1.7**	1.7	**0.7**	**2.2**
ITM-emp	0.5	2.2	2.4	**4.1**	**3.4**	1.3	**2.1**	-0.5	2.0
BL2	0.4	**3.1**	1.6	2.2	1.3	**1.7**	1.8	0.1	1.5
BL3	0.2	2.5	1.3	1.5	1.3	0.4	1.6	-0.5	1.0

Table 5: % Improvement in MAE compared to BL1 in ANN algorithm

As we see, our algorithm provides a considerable improvement over the naive baseline (BL1), by 3.5%, 2.6% and 2.2% on average for ENET, SVR and ANN respectively. It also outperforms the two change-point detection approaches (BL2, BL3), by at least 2% for ENET and SVR and 0.7% for ANN. The improvement is consistent for every algorithm and every period of the year. In order to assess the statistical significance of the results we performed t-tests on the difference of the errors, between our algorithms and each baseline. For all forecasting models the statistical significance of the average improvement throughout the year over each baseline is very high, with p-values lower than 0.001. The only exceptions are, for the ANN model, between ITM-stat and BL2 (p-value = 0.135) and ITM-emp and BL3 (p value = 0.04). Thus, we can claim that there are, in fact, aspects in the training of a machine learning model that our algorithm captures more effectively than the baselines.

We note that the performance of the algorithms varies throughout the year. For example, in the case of the Elastic-Net algorithm, the improvement ranges from 0.5% to 5.9%, which shows that the improvement depends on the characteristics of the time-series. The highest improvement is in days 160, 200, 240, which correspond to months December, January and March. This can be attributed: (a) to the existence of the winter holidays, that affect water consumption and subsequently the training of the model and (b) to the change of weather conditions.

A direction for improving ITM would be to enable it to identify changes from smaller samples, thus being able to respond faster to changes in the time-series. This may be achievable through a cross validation scheme. Another modification that could improve the performance is to design a new significance-test function, which weighs both the probability that a change is statistically significant and the expected reduction in error, for retrieving the optimal training set.

3.6. Event detection and personalized recommendations

3.6.1. Event detection and alerting

As far as alerting mechanisms are concerned, the basis here is consumption forecasting, as described in the previous subsections. Given a reliable consumption forecast for the next day (*resp. hour, part of day, week, etc.*), we are able to compare it with the actual consumption, identify unusual events and behavior changes and alert the users if certain thresholds are surpassed. Finally, the personalization process described next can be applied in this setting too: identifying discrete consumption patterns corresponding to different users can result to defining different alerting thresholds.

3.6.2. Personalization and recommendations

As previously discussed, personalization and recommendation facilities of DAIAD@home are depended on our work on pattern and event detection: identifying fine-grained consumption patterns constitutes the key to being able to discriminate different household users and, consequently, personalize consumption analysis and recommendations.

While T4.2 progresses and allows the enhancement of the consumption pattern recognition task, we will soon be able to apply it for personalization. Being able to identify discrete consumption events, will allow mapping of specific users within the household to specific consumption events. This will, consequently, allow the personalization of consumption recommendations to each user: identifying different shower patterns for different users will facilitate the provision of discrete consumption statistics per user, as well as different recommendations on changing water consumption behavior. For example, the thresholds for recommending a decrease in showering water consumption could be set differently for users with different consumptions.

To this end, we are currently implementing an algorithm for personalized pattern recognition and recommendation processes. The algorithm identifies users implicitly, through the identification of different showering patterns. In order to do so, it performs clustering on the historical showering data measured by Amphiro b1 device. To improve the efficiency of the process, it extracts several features (like the average

temperature of the shower, the number of breaks etc.) and performs clustering on that feature space. Each cluster represents a different type of consumption, possibly belonging to a different user. Then, each new consumption is compared to each of the clusters and is characterized according to the most similar cluster to it. Thus, it is assigned to a specific consumption type, and, by extension, to a specific user. The algorithm will be able to be properly evaluated, when we obtain a large amount of highly granular shower consumption data from the trials.

Finally, another course of our ongoing work involves the construction of individual or collaborative insights-recommendations mechanisms that aim to provide targeted information to the users, in order to increase their awareness and/or induce consumption behavior changes. These insights might be produced by straightforward or more elaborate computations on daily/weekly/monthly consumptions of users. Also, they may focus exclusively on individual users/households or exploit collaborative information of groups of users.

3.7. Future Directions

Our future work involves the following tasks. We will constantly improve the forecasting precision of the implemented algorithms by exploiting new and richer water consumption datasets. Specifically, through the course of the trials, we will be able to obtain highly granular water consumption data, as well as labelled data (e.g. shower consumptions mapped to specific household users). These will facilitate us towards enhancing the current pattern recognition and forecasting functionality. Further, we will complete the implementation of event detection and personalized recommendations algorithms and will evaluate them on data gathered by the trials. Upon that, we will offer a rich set of static and dynamic recommendations, alerting mechanisms and insights-recommendations to the end users, taking also into account the deployment plans of the trials.

4. Annex: Data Schema

In this section we briefly present the major tables of the local database used from the DAIAD@home mobile application.

4.1. User Data

Table '*user*' stores user related information.

Column	Description
firstname	-
lastname	-
email	User's unique email address
password	-
gender	-
country	-
zip	Postal code
birth	Birth date
userkey	Unique user registration key generated by the application server

4.2. Device Data

Table '*devices*' stores information about DAIAD@feel device registrations. As mentioned in Section 2 the registration data is serialized as a JSON string.

Column	Description
user_mail	User's unique email address
devs	Array of user's registered DAIAD@feel devices serialized in JSON format

An example of a device registration in JSON format is shown next in pseudo JavaScript code.

```
[{
    // Encryption key
    'aeskey': '',
    // Unique device Id
    'id': '',
    // Device registration date and time
```

```
    'firstTimeOfConnection': '',
    // Last communication date and time
    'lastTimeOfConnection': '',
    // Last device update date and time
    'lastTimeOfUpdate': '',
    // Last shower unique Id
    'lastShowerID': '',
    // Array of pending BLE connections
    'pendingRequests': [],
    // Device properties  values
    'values': ['Amphiro b1 #0', 'Metric', 'EUR', '900', '5', '0', '100', '0.33', '0',
'14', '1', '180', '10'],
    // Unique device registration key generated by the application server
    'deviceKey': ''
 }]
```

4.3. Measurement Data

Table feel stores measurements collected by DAIAD@feel devices.

Column	Description
id	Device unique Id
history	Boolean flag. True if the measurement is real time; False otherwise.
indexs	Shower unique Id
cdate	Date received
category	Function code
volume	-
flow	-
temp	Temperature
energy	-
tshower	Unique shower Id

5. Annex: REST API

The REST API offers functionality for user registration and authentication, device registration and data exchanging. All requests and responses are expressed as objects serialized in JSON format. The available operations and the associated endpoints are enumerated in the next table. The input and output are the name of classes used for serializing requests and responses as JSON.

Operation	Endpoint	Input[10]	Output[11]
User registration	/api/v1/user/register	UserRegistrationRequest	UserRegistrationResponse
User authentication	/api/v1/auth/login	Credentials	RestResponse
Device registration	/api/v1/device/register	DeviceRegistrationRequest	DeviceRegistrationResponse
Data uploading	/api/v1/data/storage	MeasurementCollection	RestResponse

An example of request and response messages for user registration is show next.

```
{
    'username': 'george.papadopoulos@company.gr',
    'password': 'password',
    'firstname': 'George',
    'lastname': 'Papadopoulos',
    'gender': 'MALE',
    'birthdate': '1957-02-11',
    'country': 'Greece',
    'zip': '41221',
    'group': null
}
```

```
{
    'errors': [],
    'applicationKey': 'e573a8f7-4fea-4b1e-a047-5eea2837cf37',
    'success': true
}
```

[10] Class name of objects serialized as JSON representing input arguments

[11] Class name of objects serialized as JSON representing operation execution results

6. Annex: Evaluation datasets

6.1.1. Amphiro historical

This data set includes (a) shower extraction events from a1 devices and (b) detailed demographic information, produced in the context of an external study performed by Amphiro on 77 Swiss households (5,795 showers).

The dataset is provided in a plain-text file (CSV) with a well specified format in which every field is separated by a semi-colon (;) character. Its fields are:

- Household ID
- Trial group ID (treatment groups for particular study; ignored)
- Total number of showers per household
- An incremental shower ID (unique per device)
- Duration of a shower in seconds
- Volume of water used during a shower in liters
- Average flow rate of a shower in liters per minute
- Average temperature of a shower in ^{0}C
- Duration of interruptions during a shower (e.g. while soaping) in seconds
- Number of male residents living in a household
- Number of female residents living in a household
- Number of residents living in a household and aged between 0 and 5 years; 6 and 15 years; 16 and 25 years; 26 and 35 years; 36 and 45 years; 46 and 55 years; 56 and 65 years; 66 and 75 years; over 75 years
- Costs for water consumption included in the rent (YES/NO)
- Costs for heating energy included in the rent (YES/NO)
- Number of household residents using the monitored shower
- Number or household residents having long hair
- Gender, age, nationality, and level of education of the survey taker (one per household)
- Number of adults living in the household
- Number of children living in the household
- Form of housing
- Monthly net income of the household in CHF (Fr)

An example of several records is the following (consecutive commas indicate no value):

2640,2,47,47,0,1591,0,304,37.5,11.4645,2,1,1,,,,,,,2,,0,0,2,0,male,65+,Switzerland,apprenticeship,2,no child, rental apartment,81-115 m2,7000 - 7999 Fr.

2640,2,47,34,0,216,0,33,37.5,9.16667,2,1,1,,,,,,,2,,0,0,2,0,male,65+,Switzerland,apprenticeship,2,no child,rental apartment,81-115 m2,7000 - 7999 Fr.

2640,2,47,3,0,31,0,5,37.5,9.67742,2,1,1,,,,,,,2,,0,0,2,0,male,65+,Switzerland,apprenticeship,2,no child,rental apartment,81-115 m2,7000 - 7999 Fr.

2640,2,47,4,0,1406,0,244,37.5,10.4125,2,1,1,,,,,,,2,,0,0,2,0,male,65+,Switzerland,apprenticeship,2,no child,rental apartment,81-115 m2,7000 - 7999 Fr.

6.1.2. Historical SWM data

This dataset contains hourly SWM time-series for 1,000 households in Alicante (selected randomly) covering a time period of a whole year, i.e., from 1st of July 2013 to 30th of June 2014 (8.6M records).

SWM time-series are provided in plain-text files (CSV) with a well specified format in which every field is separated by a semi-colon (;) character. The default format is presented in the figure below.

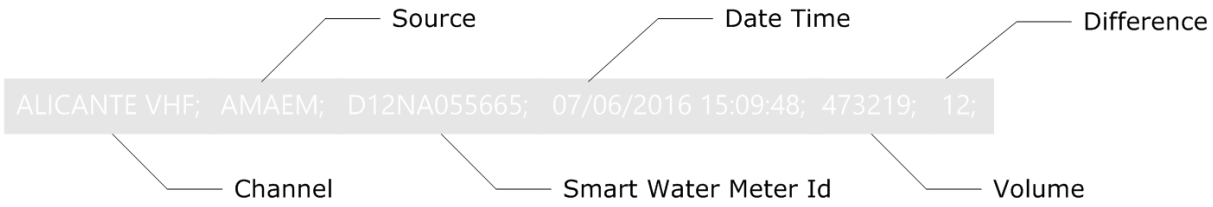

Figure 27: Smart Water Meter data example

Every record consists of six fields:

- **Channel**. A string representing the transmission channel used to receive the SWM readings. This is an optional field, as for all SWMs deployed in Alicante it has the same value ('Alicante VHF').
- **Source**. A string representing the source (i.e. utility) the SWM belongs to. This is an optional field, as for all SWMs deployed in Alicante it has the same value ('AMAEM').
- **SWM ID**. An alphanumeric value uniquely identifying each SWM.
- **Reading date**. A timestamp (date, time) of when the reading was taken from the SWM, expressed in local time zone (Europe/Madrid for AMAEM).
- **Volume reading**. The actual value measured by the SWM in liters.
- **Difference**. The difference in liters between two most recent SWM readings. This is an optional field.

This dataset proved extremely valuable for our entire body of work (beyond simply integration and testing) as it shed light to the real-world characteristics and issues of SWM data management and analysis. In particular, we identified the following data quality issues:

- Missing SWM readings for a particular SWM ID. This can be caused by a malfunctioning SWM, SWM RF component (add-on to the mechanical SWM), or RF connectivity (temporary connection issues between the SWM and the RF antenna/aggregator). Missing SWM readings can vary, from 1 missing reading/day to several days.
- Out of order SWM readings. This can be caused by a malfunctioning SWM (internal clock) or the underlying software for SWM data management (external to the project).
- Changing time period between consecutive measurements. This can be caused by a malfunctioning SWM or the underlying software for SWM data management (external to the project). The period between two consecutive measurements for the same SWM is not always the anticipated 1h, but may drift for a number of minutes (e.g., 80 min instead of the expected 60 min).

It is worth highlighting that such data quality issues are considered as an *accepted behavior* of a SWM roll-out (i.e. within normal operation parameters), as the emphasis is given on accurate billing (i.e. difference between two measurements in time for the billing period). Consequently, low data quality was acknowledged early in the project's lifecycle as an *integral characteristic* of production SWM roll-outs that must be gracefully handled by the entire DAIAD system.

A sample of several records for a single SWM follows (notice the optional fields Channel, Source, Difference are missing, as well as the slight change in time period):

> *C12FA151955;08/06/2014 03:07:17;112370;*
>
> *C12FA151955;08/06/2014 02:07:17;112369;*
>
> *C12FA151955;08/06/2014 01:07:17;112369;*
>
> *C12FA151955;07/06/2014 23:46:26;112368;*
>
> *C12FA151955;07/06/2014 22:46:26;112366;*

7. Annex: Shower sequences

The following sets of figures provide examples of consumptions sequences of showers, as recorded by Amphiro a1, contained in the Amphiro historical dataset described in Sections 3.2.1 and 6.1.1. Each set of figures presents the consumption sequences, the histogram and the correlogram respectively (in accordance to the example presented in Section 3.2.1) for different demographic cases. Specifically, we present cases of different numbers of household members (Figure 28 and Figure 29), different apartments (Figure 30 and Figure 31) and different incomes (Figure 32, Figure 33 and Figure 34).

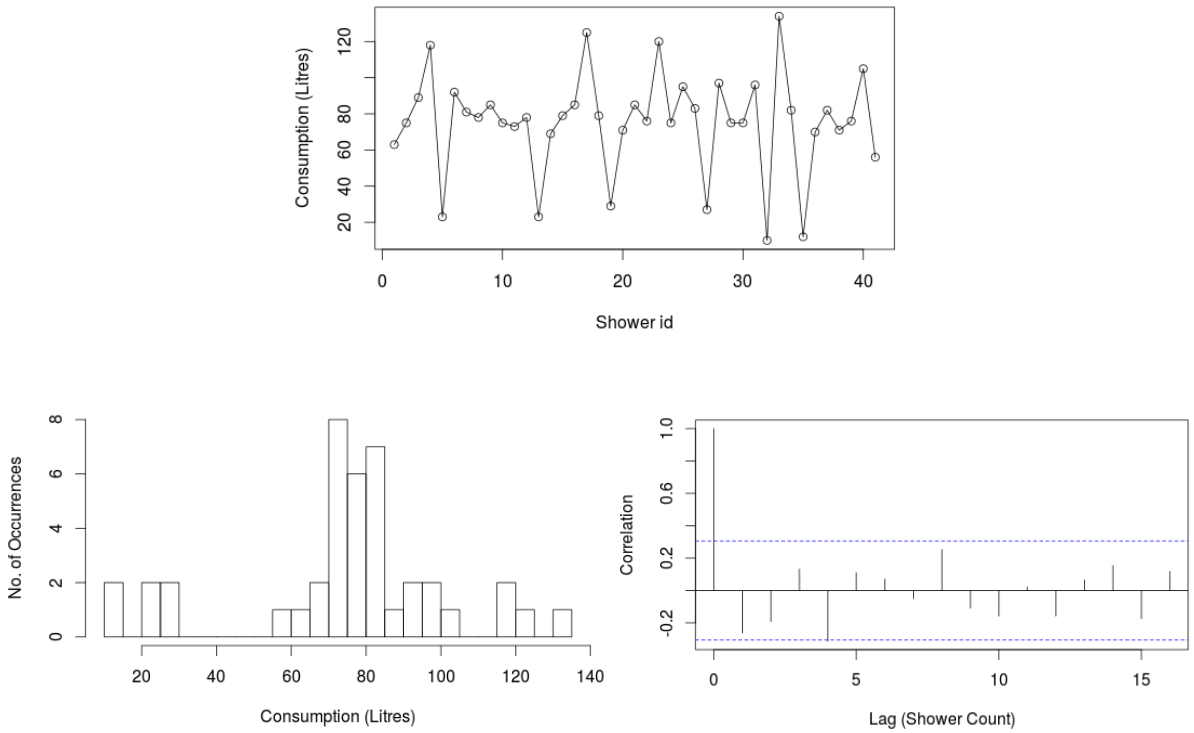

Figure 28: Single person household

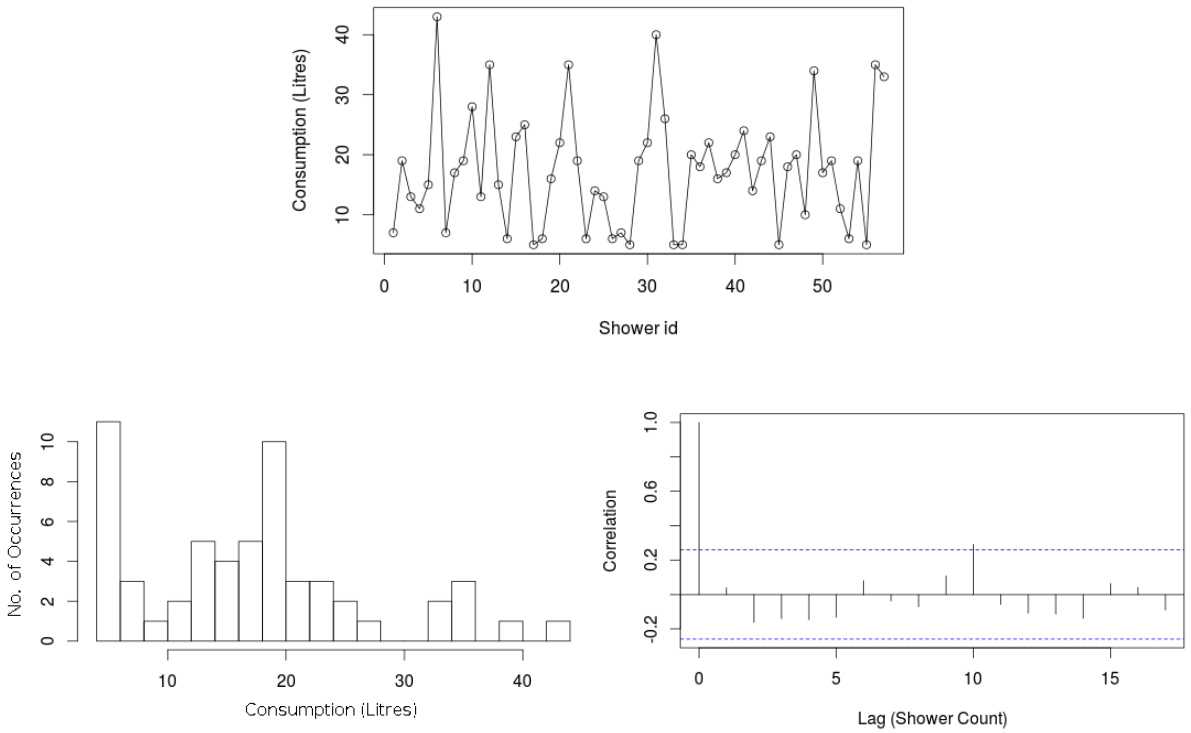

Figure 29: Two-person household

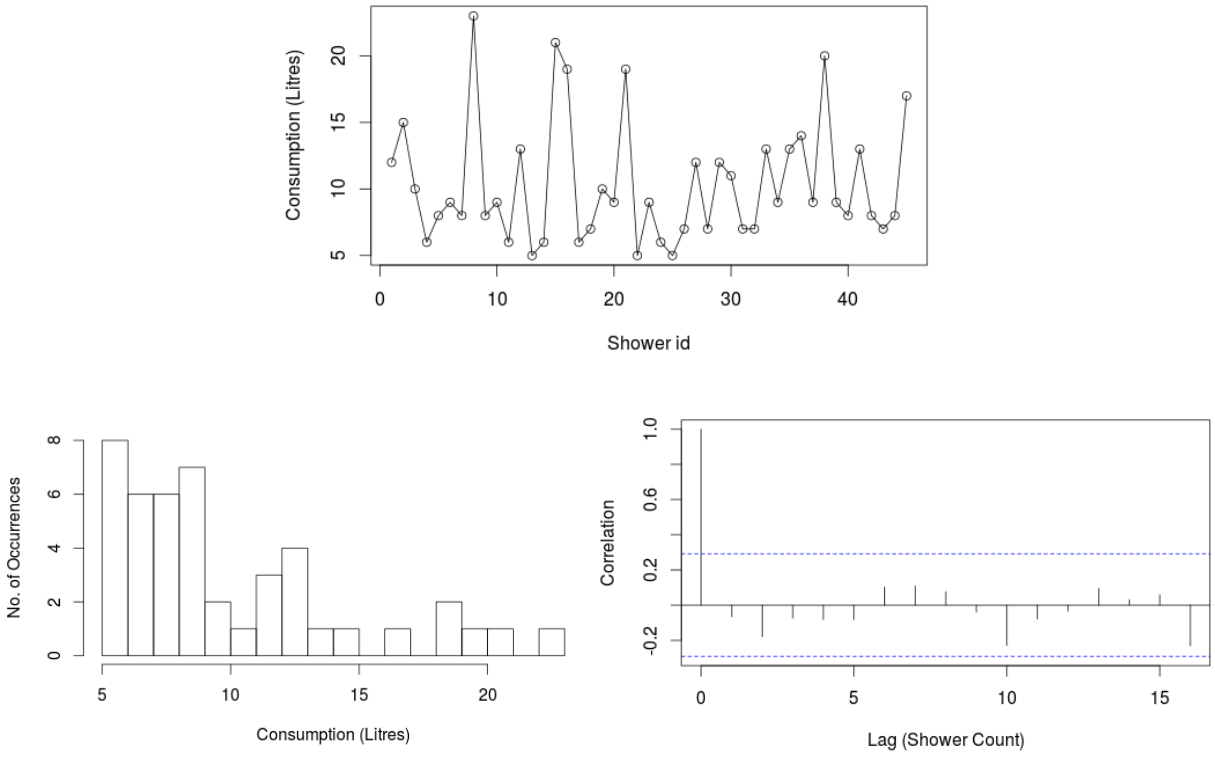

Figure 30: Condo apartment

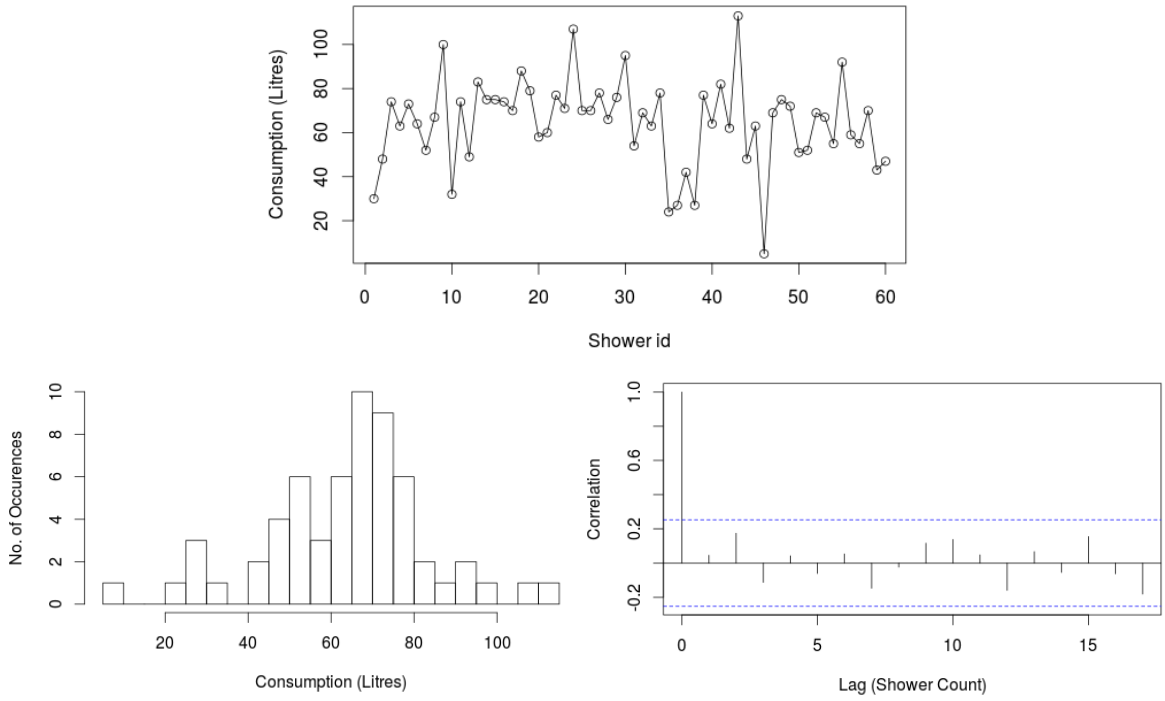

Figure 31: Rental apartment

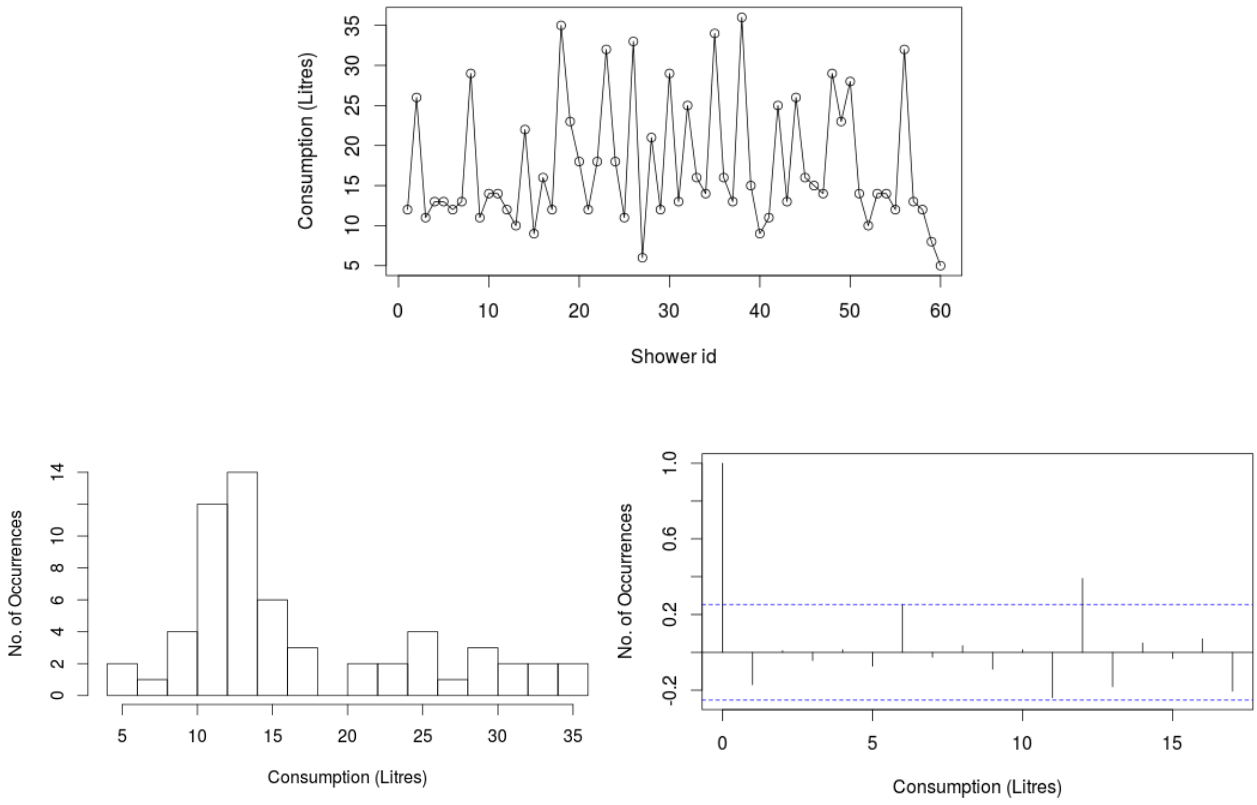

Figure 32: Household with income <5000 CHF

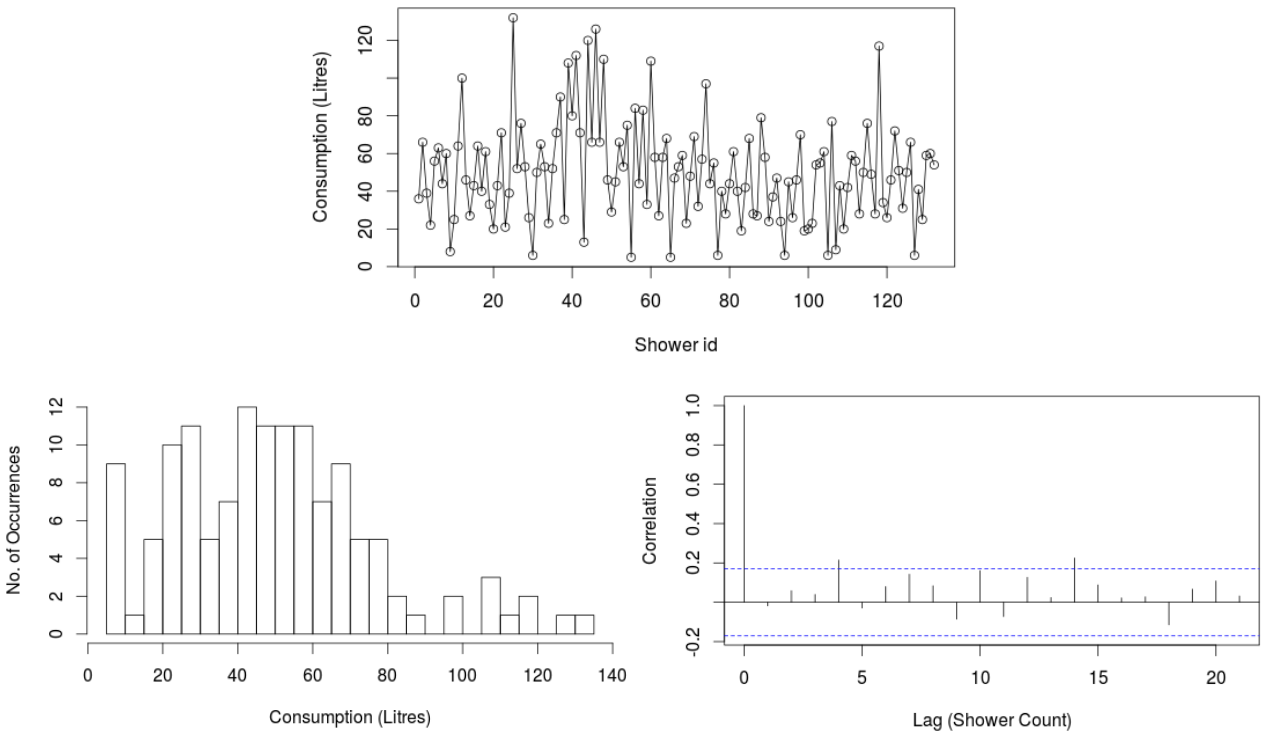

Figure 33: Household with income 5000 -9000 CHF

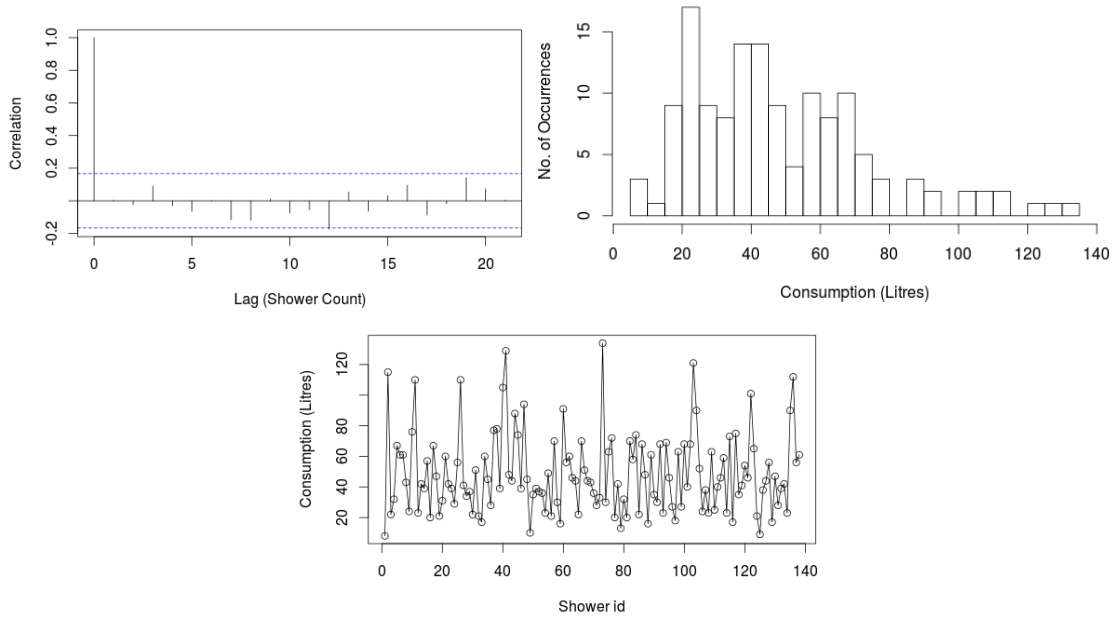

Figure 34: Household with income >9000 CHF

8. Annex: Implementation details

In this Annex, we provide additional information specifically on the forecasting algorithms study we performed, as well as on the novel algorithms we implemented for Pattern Forecasting and Model Change Detection.

8.1. Execution Environment

Our experimental evaluation was performed on a workstation with the following specifications:

- Intel(R) Core(TM) i7-3820 CPU @ 3.60GHz
    - 4 cores x 2 threads = 8 threads
- 32 Gb RAM
- 500 Gb HDD

8.2. Implementations and API

We divide our code into three separate implementations: one for the initial study for the performance of state of the art algorithms (Section 3.3); one for the Pattern Forecasting algorithm (Section **Error! Reference source not found.**) and one for the Iterative Training Method algorithm (Section 3.5). All implementations are in JAVA and R languages. Next we provide more details. The code is available on GitHub[12].

8.2.1. Evaluation of forecasting algorithms

The main JAVA classes of the implementation and the used libraries are:

- AllData: Class that is used to store the time-series and the metadata, in continuous and discretized form, and pass the data to the various algorithms
- BaselineRegressor: Class that implements the Baseline algorithms BR0, BR1 and BR2.
- CategoricalLinearRegressor: Class that implements the CLR algorithm. It uses the EJML[13] library that provides Linear Algebra operations
- DataPreprocessor: Class that reads the data from a csv file and transforms them to the types that are used in the rest of the code, for the shower consumption dataset. Performs clustering using the Apache Commons[14] library and discretization.

[12] https://github.com/DAIAD/home-web/tree/master/jobs

[13] ejml.org/wiki

[14] https://commons.apache.org/

- LinearAutoRegressor: Class that implements the LAR algorithm. It uses the EJML[13] library that provides Linear Algebra operations
- Results: Class that stores the predictions of each algorithm and calculates the errors
- SequentialRegressor: Class that implements the SR0 and SR2 algorithms.
- SupportVectorMachine: Class that implements the SVM. Uses JAVAML[15] and LIBSVM[16] libraries.
- SupportVectorRegression: Class that implements the SVR. Uses JAVAML[15] and LIBSVM[16] libraries.
- NewDataPreprocessor: Class that reads the data from a csv file and transforms them to the types that are used in the rest of the code for the SWM dataset. Performs clustering using the Apache Commons[14] library, discretization and data cleaning.
- EfficientSR: Class that contains an efficient, in terms of memory, implementation of SR0 and SR2 algorithms that is required for the SVW data.
- TimeFixer: Class that implements the required functions for the alignment of the SWM data.

There are also 3 algorithms implemented in the following R scripts:

- ANN.R : Implements the ANN algorithm, using the neuralnet[17] package of R
- arima_hw.R: Implements the Arima and the Exponential smoothing algorithms using the forecast[18] package of R.

For every class that implements a forecasting algorithm there exists a train method and a test method that can be used to Interface with the classes. Train takes as input training data and the required parameters for the algorithm and trains the model. Test provides the forecast of the algorithm. There are also several utility classes that are used for trivial tasks such as containing the data and performing minor transformations. The code for the performed evaluation is available on GitHub[2].

8.2.2. Pattern Forecasting

The main classes of the implementation of the Pattern Forecasting algorithm are:

- ActivityZone: A class that represents the activity zone. It contains information such as the start, the end, the threshold for the discretization of the activity and the usual consumption patterns in the activity zone.
- Day: Class that represents the water consumption for a day. Includes the timestamps and the identified patterns of the day
- DiscretePatternForecaster: Class that predicts whether there will be significant activity in each activity zone of the next day.
- Linear24hSVR: Class that implements the Support Vector Regression model. It contains 24 separate SVR models, one for each hour of the day. Uses the Liblinear library.

[15] http://java-ml.sourceforge.net/

[16] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[17] https://cran.r-project.org/web/packages/neuralnet/

[18] https://cran.r-project.org/web/packages/forecast/

- DiscretePatternFilter: Class that combines the predictions from DiscretePatternForecaster and the Linear24hSVR. It applies the prediction for the activity zones made by DiscretePatternForecaster to the time-series prediction made by Linear24hSVR.
- Preprocessor: Class that reads the water consumption time-series from a csv file and performs basic preprocessing and data cleaning.
- Results: Class that contains the results of each forecasting algorithm and calculates the error.
- TimeSeriesAnalyser: Class that contains the methods that perform the pattern recognition functionality. Mainly it identifies the activity zones.

DiscretePatternForecaster, Linear24hSVR and DiscretePatternFilter classes include train and test methods that can be used to interface with them and obtain their functionality, as described in the previous section. The code for the implemented algorithm is available on GitHub[7].

8.2.3. Model Change Detection

The ITM algorithm was implemented entirely in R. The implementations are contained in the following R scripts:

- itm_enet.R: This scripts contains the implementation of itm with the ENET model. Uses the glmnet library.
- itm_svr.R: This scripts contains the implementation of itm with the SVR model. Uses the libsvm and Liblinear libraries
- itm_ann.R: This scripts contains the implementation of itm with the ANN model. Uses the neuralnet library

For the baseline methods implemented in R the cpm library is used. Every script contains a method named select_start that identifies and returns the optimal change point. The code for the implemented algorithm is available on GitHub[8].

9. References

[Ath14] S. Athanasiou et al. D1.1 State of the art Report. DAIAD project, 2014. Available at: http://daiad.eu/wp-content/uploads/2015/02/D1.1_State_of_the_art_Report_v1.0.pdf

[BC02] G. Casella, R. Berger. Statistical Inference. 2nd edition, Duxbury, 2002.

[Bis09] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2009.

[CGA16] P. Chronis, G. Giannopoulos, S. Athanasiou. Open Issues and Challenges on Time-series Forecasting for Water Consumption. EDBT/ICDT Workshops, 2016.

[CGS16] P. Chronis, G. Giannopoulos, S. Skiadopoulos. Optimizing the training of time-series forecasting models: A change point detection approach. (Under Review).

[Gar06] E. S. Gardner. Exponential smoothing: The state of the art - Part II. International Journal of Forecasting, 2006.

[GS99] V. Guralnik, J. Srivastava. Event Detection from Time Series Data.
Conference on Knowledge Discovery and Data Mining, 1999.

[Gus00] F. Gustafsson. Adaptive Filtering and Change Detection. 1st ed., Wiley,
2000.

[HD10] D. M. Hawkins, Q. Deng. A Nonparametric change-point Control Chart.
Journal of Quality Technology, 2010.

[Mei02] E. Meijering. A chronology of interpolation: from ancient astronomy to modern signal and image processing". Proceedings of the IEEE, 2002.

[NJ02] A. Ng, M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. Conference on Neural Information Processing Systems, 2002.

[OL10] H. Ohlsson, L. Ljung, S. Boyd. Segmentation of ARX-models using sum-
of-norms regularization. Automatica Journal, 2010.

[SS98] A. J. Smola, B. Schölkopf. A Tutorial on Support Vector Regression. Technical report, 1998.

[Tay06] J. W. Taylor, L. M. Menezes, P. E. McSharry. A comparison of Univariate Methods for Forecasting Electricity Demand Up to a Day Ahead. International Journal of Forecasting, 2006.

[Tay10] J. Taylor. Triple Seasonal Methods for Short-Term Electricity Demand Forecasting. European Journal of Operational Research, 2010.

[ZH05] H. Zou, T. Hastie. Regularization and Variable Selection via the Elastic Net. Journal of the Royal Statistical Society, 2005.