PRE-PROCESSING OF HETEROGENEOUS DATA STREAMS FOR INTERNET OF THINGS APPLICATIONS

Klemen Kenda

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

Supervisor: Prof. Dr. Dunja Mladenić, Jožef Stefan Institute and Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Evaluation Board:
Prof. Dr. Aleš Švigelj, Chair, Jožef Stefan Institute and Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
Prof. Dr. Chrysi Laspidou, Member, University of Thessaly, Volos, Greece
Dr. Blaž Fortuna, Member, Jožef Stefan Institute, Ljubljana, Slovenia

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Klemen Kenda

PRE-PROCESSING OF HETEROGENEOUS DATA STREAMS FOR INTERNET OF THINGS APPLICATIONS

Doctoral Dissertation

PREDPROCESIRANJE HETEROGENIH TOKOV PODATKOV ZA APLIKACIJE V INTERNETU STVARI

Doktorska disertacija

Supervisor: Prof. Dr. Dunja Mladenić

Ljubljana, Slovenia, February 2024

Acknowledgments

I would like to take this opportunity to express my deep gratitude and appreciation to the many people who have supported and guided me throughout my doctoral journey.

First and foremost, I am profoundly thankful to my advisor, Prof. Dr. Dunja Mladenić, for her unwavering support, invaluable guidance, and patience throughout the entire research process. Her mentorship and expertise have been instrumental in shaping this work and my academic growth.

I also thank the members of my dissertation committee, Prof. Dr. Chrysi Laspidou, Prof. Dr. Aleš Švigelj, and Dr. Blaž Fortuna, for their insightful feedback and critical evaluation of my research. Their collective wisdom has greatly enriched the quality of this thesis.

I am grateful to my colleagues and fellow researchers at Jožef Stefan Institute, Erik Novak, Dr. Alenka Guček, Maja Škrjanc, Filip Koprivec, Zala Herga Rupnik, Prof. Dr. Primož Škraba, Marko Grobelnik, Dr. Inna Novalja, Swati, Dr. Jože Rožanec and Jože Peternelj for their stimulating discussions, collaboration, camaraderie and insightful debates that so often exceeded the formal limitations of the work environment. New horizons have been revealed to me also by the knowledgeable coworkers at the Qlector d.o.o., among them I especially thank Viktor Jovanoski, Dr. Jan Rupnik, and Dr. Blaž Kažič. Their contributions have played a significant role in shaping my ideas and refining my research. I want to express my sincere thanks to our office staff, Jasna Franko and Mojca Kregar. Their practical assistance, patience, and emotional support have been invaluable throughout this project. Their dedication and willingness to go the extra mile have been greatly appreciated.

I would also like to thank numerous coworkers on several research projects through my years at the Jožef Stefan Institute. Especially all the researchers and students who shaped the work and ideas in FP7 NRG4CAST, FP7 Sunseed, H2020 Water4Cities, H2020 NAIDES, and H2020 PerceptiveSentinel projects. All of them cannot be mentioned here but have in various ways contributed to the successful completion of this thesis.

Lastly, I want to keep a special place in these acknowledgments for my friend Grega Milčinski. He helped to kick off my career after enrolling in the PhD program by inviting me to collaborate in H2020 PerceptiveSentinel proposal, which became my first of several accepted proposals in the European Commission research calls. Moreover, he always offered me a view from afar of the problems I faced during this time and helped me with constructive solutions.

Abstract

The rapid evolution of sensor technologies, particularly within the Internet of Things (IoT) domain, has led to an era dominated by massive real-time data flows. This transition resulted in the need for novel data processing methodologies that facilitate the shift from traditional offline batch analytics to agile real-time predictions and analyses.

Dissertation supports the deployment of analytical solutions from the laboratory environment to the real-world setting. In particular, we address the problems of autonomous data cleaning, data enrichment and fusion, feature selection, and overall design of such a solution. At the core of this study lies an incremental data fusion technique for generating feature vectors suitable for machine learning models that are built from a set of heterogeneous data streams. The significance of such a methodology has been largely ignored in most studies in this field, which tend to concentrate solely on the effectiveness of machine learning models. These studies assume that the data used are consistent, aligned, and readily accessible, which is rarely the case in real-world scenarios.

The aim of this work is to provide an architecture and functional framework that address the limitations discussed above. First, we present a data cleaning methodology that takes advantage of the ability of the Kalman filter to make short-term predictions, including predicting variance. The method can be used to clean data streams with a sampling rate that is much higher than the rate at which the measured phenomena change, which is a common situation in the Internet of Things (IoT). Second, we present the methodology for fusing multiple heterogeneous streaming data sources into feature vectors. The proposed methodology has the ability to address the challenges posed by the features of heterogeneous data streams, such as time delay and varying data sampling rates. In addition, it enables the incorporation of both predicted and precalculated values into the feature vector. Using the proposed methodology, the system generates feature vectors consisting of data values that are aligned with the appropriate timestamps, values that have been aggregated or enriched, delayed values, static values that are relevant, and predictions, such as weather forecasts for the corresponding timestamp. This type of system has the capability to generate comprehensive feature vectors, which allows for effective modeling of the system. A large quantity of possibly correlated features does not necessarily lead to the optimal modeling outcomes. Hence, FASTENER, a novel feature selection algorithm that employs genetic algorithms and multi-objective optimization. The algorithm was designed for the task of segmenting Earth observation images. However, it has demonstrated unexpected effectiveness in various other situations, such as time-series forecasting. Subsequently, we place the created solutions within the Big Data lambda architecture, which consists of two pillars: speed and batch. We propose extending the capabilities of the speed pillar, responsible for processing real-time data and providing low-latency results, with analytical capabilities such as anomaly detection and forecasting. These abilities are driven by presented incremental data fusion and enrichment techniques and surpass the event detection scenarios that have traditionally been envisioned in this pillar. We apply this architecture to facilitate the water management domain. In the water

management domain we also test the usability of incremental learning algorithms such as Hoeffding trees and compare them to traditional batch methods.

The effectiveness of the suggested approach has been evaluated in several scenarios, including domains such as energy management, transport, and environment. Most recently, the framework has been integrated into the final platform of the H2020 NAIADES project, successfully demonstrating its applicability in real-world scenarios of water management.

Povzetek

S hitrim razvojem senzorskih tehnologij, še posebej v okviru interneta stvari (IoT), smo vstopili v obdobje, ki ga zaznamujejo velike količine podatkov, ki so na voljo v realnem času. S tem so nastale potrebe po novih metodah obdelave podatkov, ki omogočajo prehod od tradicionalne paketne (batch) analize do uporabe metod analize podatkov v realnem času.

Doktorska disertacija se ukvarja s prenosom analitičnih rešitev iz nadzorovanega laboratorijskega v realno okolje. Posebej se ukvarjamo s problemi avtonomnega čiščenja, obogatitve in združevanje podatkov v realnem času, izbire značilk ter izdelave ustreznih informacijskih rešitev. Jedro te študije predstavlja nova tehnika inkrementalnega združevanja podatkov iz heterogenih podatkovnih tokov v vektorje vektorjev značilk, primerne za uporabo v modelih strojnega učenja. Pomembnost takšne metodologije je bila v večini študij na tem področju spregledana, saj se slednje večinoma osredotočajo zgolj na učinkovitost modelov strojnega učenja. Te študije predpostavljajo, da so uporabljeni podatki pravilni, časovno usklajeni in vedno dostopni, kar v realnih scenarijih redko drži.

Cilj te naloge je razviti arhitekturo, ki presega zgoraj navedene privzetke. V disertaciji najprej predstavimo metodologijo čiščenja podatkov, ki izkorišča zmožnost Kalmanovega filtra za izdelavo kratkoročnih napovedi, vključno z napovedovanjem variance. Ta metoda se lahko uporablja za čiščenje podatkovnih tokov, katerih vzorčenje je veliko višje od hitrosti sprememb merjenih pojavov, kar je običajno pri internetu stvari (IoT). Nadalje predstavimo metodologijo za sprotno združevanje množice heterogenih virov pretočnih podatkov v vektorje značilk. Predlagana metodologija zmore preseči izzive, ki nastanejo zaradi heterogenih podatkovnih tokov, vključno s časovnim odstopanjem posameznih meritev in različno hitrostjo vzorčenja meritev. Poleg tega sistem omogoča tudi vključevanje napovedi in predhodno izračunanih vrednosti v vektorje značilk. S pomočjo opisane metodologije je sistem sposoben generiranja vektorjev značilk, ki vključujejo časovno usklajene podatke iz različnih virov, agregirane in obogatene vrednosti, odložene vrednosti, statične vrednosti, in vrednosti relevantnih napovedi, kot so npr. vremenske napovedi. Takšen sistem je sposoben ustvarjanja zelo obsežnih vektorjev značilk, ki omogočajo učinkovito modeliranje. Velika količina značilk pa ne vodi nujno do optimalnih modelskih rezultatov, zato v nadaljevanju disertacije predstavimo še algoritem za izbiro značilk FASTENER, ki uporablja genetske algoritme in večkriterijsko optimizacijo. Algoritem je bil posebej zasnovan za nalogo segmentacije satelitskih slik za potrebe v kmetijstvu, vendar pa je pokazal nepričakovano učinkovitost tudi v različnih drugih primerih, npr. za napovedovanje časovnih vrst v energetiki, prometu in upravljanju z vodo. Vse predstavljene rešitve na koncu postavimo v lambda arhitekturo v okviru masovnih podatkov (Big Data). Lambda arhitekturi tako dodamo analitične zmožnosti, ki niso omejene zgolj na zaznavanje dogodkov. Predlagano arhitekturo smo uporabili pri implementaciji sistema za podporo odločanju na področju upravljanja voda. Na istem področju smo preizkusili tudi uporabnost algoritmov za inkrementalno učenje, kot so npr. Hoeffdingova drevesa, in jih primerjali s tradicionalnimi paketnimi metodami.

Učinkovitost predlaganega pristopa smo ocenili v več scenarijih, ki obsegajo področja, kot so upravljanje z energijo, prometom in okoljem. Najnovejša implementacija celotne rešitve je bila izvedena v okviru evropskega projekta H2020 NAIADES, kar dokazuje njeno uporabnost na področju upravljanja voda.

Contents

Abbreviations

AI	...	Artificial intelligence
AKF	...	Augmented Kalman filter
AO	...	Additive outlier
API	...	Application programming interface
ARIMA	...	Auto-regressive integrated moving average
ARSO	...	Slovenian Environment Agency
ARVI	...	Atmospherically resistant vegetation index
ASTFS	...	Automatic spectro-temporal feature selection
AUC	...	Area under a curve
AUF	...	Area under a (Pareto) front
CRISP-DM	...	Cross-industry standard process for data mining
DB	...	Database
DS	...	Data source
DT-forward	...	Forward selection with decision trees
DEM	...	Digital elevation model
DNA	...	Deoxyribonucleic acid
EFDT	...	Extremely fast decision trees
ELM	...	Extreme learning machine
EM	...	Expectation minimization
EMA	...	Exponential moving average
EO	...	Earth observation
ESA	...	European Space Agency
EPANET	...	Application for modeling drinking water distribution systems
EVI	...	Enhanced vegetation index
FASTENER	...	Feature selection enabled by entropy
FIMT-DD	...	Fast incremental model trees with drift detection
FIWARE	...	Open-source framework for interoperable smart solutions
FP7	...	Seventh framework programme of the European Community for research and technological development including demonstration activities
FS-SDS	...	Feature selection using stochastic diffusion search
FTP	...	File transfer protocol
GUI	...	Graphical user interface
H2020	...	Horizon 2020 European Union's research and innovation funding programme from 2014-2020
HAT	...	Hoeffding adaptive trees
HTTP	...	Hypertext transfer protocol
IF	...	Impact factor
IO	...	Innovation outlier
IoT	...	Internet of Things
ISDI	...	IoT Streaming Data Integration framework
IUWM	...	Integrated urban water management

JDL ... Joint Directors of Laboratories
JSON ... JavaScript object notation
KD ... k-dimensional
KNN ... k-nearest neighbors
LDA ... Latent Dirichlet allocation
LPIS ... Land parcel identification system
LS ... Level shift
LSTM ... Long short-term memory
NDVI ... Normalized differential vegetation index
NDWI ... Normalized differential water index
NLPCA ... Non-linear principal components analysis
NP ... Nondeterministic polynomial time
NSGA-II ... Non-dominated sorting genetic algorithm
MA ... Moving average
MAPE ... Mean absolute percentage error
MI ... Mutual information
ML ... Machine learning
MLOps ... Machine learning operations
MLP ... Multi-layer perceptron
MOA ... Massive online analysis
MQTT ... Message queuing telemetry transport
NIR ... Near-infrared
PB ... Petabyte
PLS ... Partial least squares
POSS ... Pareto optimization for subset selection
PPOSS ... Parallel Pareto optimization for subset selection
REST ... Representational state transfer
RF ... Random forest
RGB ... Red, green and blue
RMSE ... Root-mean-square error
RPSGAe ... Reduced Pareto set genetic algorithm with elitism
SAVI ... Soil-adjusted vegetation index
SC ... Scientific contribution
SIPI ... Structure-intensive pigment index
SLS ... Seasonal level shift
SPE ... Stream processing engine
SQL ... Structured (English) query language
SVC ... Support vector classification
SVM ... Support vector machine
SVR ... Support vector regression
SWM ... Smart water management
SWIR ... Short wave infrared
TC ... Temporary change
TSKF ... Time series Kalman filter
TSDB ... Time series database
VFDT ... Very fast decision trees
VHT ... Vertical Hoeffding trees
VNIR ... Visible/near-infrared
WMAP ... Water Management Analytical Platform

Chapter 1

Introduction

> In theory, theory and practice are the same. In practice, they are not.
>
> *Albert Einstein*

With the growth of the number of smart sensors, we are increasingly able to access vast amounts of data that can be used to improve the knowledge about observed systems. This data is usually high frequency and is often updated in (almost) real time. Because of this, the Internet of Things (IoT) has also changed the focus in machine learning from offline analysis (batch) to real-time (online) prediction and analysis.

To achieve the full analytical potential of streaming data from the internet of things, the interconnection of various data sources is needed. By definition, those sources are heterogeneous, and their integration is not a trivial task. A common approach to exploiting the potential of streaming sensor data is to use machine learning techniques for predictive analytics in a way that is agnostic to domain knowledge. This approach can be easily integrated in various use cases.

In the dissertation, we consider a methodology for development and deployment of real-world machine learning models that solve problems such as: predicting energy consumption on smart grid or demand of potable water in a water distribution network in the next days, predicting groundwater and surface water levels at various spatial points, predicting energy demand of an electrical train, etc.

The aim of the dissertation is to develop a generic pre-processing framework for heterogeneous data streams and test its modeling capabilities in real-world scenarios. The core of the framework is a real-time data fusion component, which is able to integrate several heterogeneous real-world data sources (streams or static) into a descriptive feature vector. The framework is based on the lambda architecture approach, which divides processing into two pillars: speed (real-time) and batch. We propose a modeling workflow, where the model is developed in a batch setting (including automated feature selection) and then pushed into the deployment and chained with autonomous data cleaning, feature engineering and heterogeneous streams data fusion.

We base our work on the Big Data lambda architecture (see Figure 1.1), building on main pillars [1]. We propose a workflow for development and deployment of machine learning models that extends the initial lambda architecture proposal. Our main innovation is the extension of the speed pillar, where we overcome the limits of simple event detection and processing. We introduce autonomous on-line data fusion of heterogeneous data sources [2], which supports either incremental models (online training and inference) or batch model inference and training data generation. We also propose an autonomous data

cleaning methodology based on Kalman filter for sensor data cleaning [3]. In the batch pillar, the tools have been developed in depth in the past and several industry-standard tools already exist (from data storage to data manipulation and model development) [4]. In the batch pillar, we propose a novel feature selection methodology (FASTENER), based on a genetic algorithm [5]. The methodology is able to select a subset of relevant features faster than competing algorithms and offer fast model development even with large datasets. Several offline and online modeling techniques dealing with time series prediction and anomaly detection ([1], [6]–[9]), both batch and incremental, were tested within the scope of the proposed framework.

1.1 Motivation

The scientific community has been discussing the increasing amount of data originating from the Internet of Things (IoT) for more than a decade. The IoT reached the mass market in early 2014 [10] and its ubiquitous influence and challenges are still permeating the scientific literature. The field of big data processing has improved drastically and a plethora of solutions for various IoT problems have reached their production stage.

The volume of data keeps rising, and as technology is penetrating new markets (e.g., water management), new challenges are put in front of the industry and academia. The need for efficient and accurate analysis of these data is still an issue. Stream processing has been established as a potential answer to the analysis of big data, and incremental learning has been rediscovered to answer some of the challenges (like concept drift or learning efficiency). While the field of incremental learning has matured through the last decade and a wide variety of algorithms have been described, tested, and implemented in various software libraries, the applications of methodologies from a laboratory to the real world have been scarce.

Throughout our work in various applications within the environmental domain, water management, traffic, energy efficiency, and smart grid modeling, we have (among others) identified the following shortcomings: most of the scientific work on incremental learning has taken place inside the lab, emulating unreal (ideal) conditions, which are rarely encountered in the real world; mostly this remark applies to the data availability and data preparation step (including data fusion and generation of machine-learning-ready rich data streams).

The ongoing lack of on-line data pre-processing techniques ([11], [12]) reduces the possibility of using streaming and also hybrid approaches in real-world scenarios, where data pre-processing is done on-line and prediction models are implemented using traditional machine learning (ML) approaches. McKinsey [13] has established that up to 40% of the data value emerging from IoT is hidden within the synergistic effects of different systems. With the exception of the IoT Streaming Data Integration (ISDI) framework [14], which solves time alignment issues of data integration, a generic methodology for the generation of feature vectors for machine learning approaches in the IoT scenario does not yet exist and this work aims to fill this gap. Other proposed solutions ([15]–[19]) can solve application problems; however, they lack the functionality to be implemented in a general scenario.

The framework proposed in the dissertation offers a complete streaming methodology for building rich feature vectors, describing important process characteristics (or features), suitable for traditional or incremental machine learning algorithms. Within the dedicated big data framework, the proposed methodology is able to merge data from a set of heterogeneous streaming data sources (i.e., from the IoT, weather forecasts, and data about human behavior) in a real-world setting and enables machine learning models to yield more

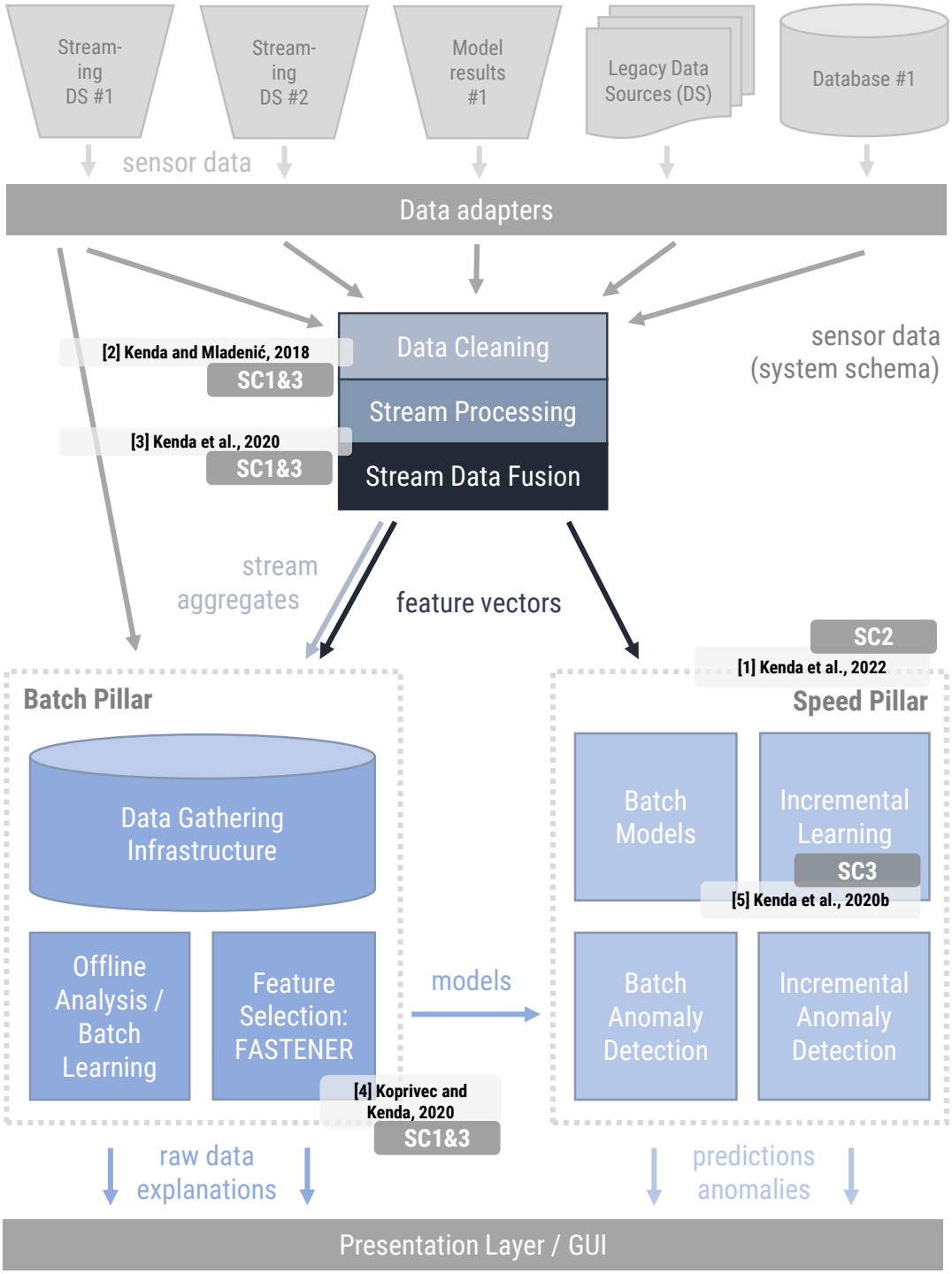


Figure 1.1: Lambda architecture for processing Big Data with scientific contributions and references to journal papers describing particular components. Paper [1] also describes the overall architecture of the system.

accurate, and thus more useful results in real time.

1.2 Aims and Hypotheses

The thesis has two main hypotheses that will be addressed and tested experimentally.

- **Hypothesis 1**: *Including multiple relevant data-sources improves the overall models' performance in real-time predictive time-series analytics.*

  Prior state of the art: No generic incremental framework for data fusion has been available in an incremental setting [2]. The majority of the incremental learning literature assumes artificial modeling scenarios, where all the data are available immediately (no delays, no missing/wrong data). It is a well-known fact that usage of several contextual data sources adds to the overall information gain and therefore improves the models' performance, which has been tested in several settings, mostly batch (offline).

  Aim: Our goal is to create a formal definition of data fusion of heterogeneous streaming data sources, as well as autonomous methodologies and prototypes for online data cleaning (based on Kalman filter), fusion, and enrichment of heterogeneous data streams. With the vast number of derived time series from the original data, our goal is to develop a novel algorithm for feature selection based on information theory and multi-objective optimization approach, which would enable efficient definition of feature vectors.

- **Hypothesis 2**: *Unified big data architecture that includes the cleaning of autonomous data and the fusion of heterogeneous source data is suitable for different scenarios in the real world (IoT, Earth observation, energy grids, water management, transport, etc.).*

  Prior state of the art: Numerous proposals have been put forward for architectures related to Big Data. The lambda architecture already distinguishes between two main pillars, batch (for offline processing) and speed (for online processing). However, in all the definitions, the speed pillar is composed only of (complex) event processing and does not include other analytical functionalities. Most of the research dealing with contextual data is performed offline, the dedicated systems are run in batched mode, or the online fusion is suitable only for a specific application.

  Aim: Our goal is to propose an extension to the big data lambda architecture for hybrid model development and deployment that will also include analytical capabilities (such as predictive analytics and anomaly detection) in the speed pillar, as well as means for pushing the trained model (or incremental model definitions) from batch to speed pillar (house architecture). Our aim is also to introduce and test the feasibility of incremental learning in the field of water management and earth observation. Finally, we aim to evaluate the proposed architecture and methods in several real-world scenarios from the energy management, water management, smart cities, transport energy management, and earth observation domains.

1.3 Scientific Contributions

The main scientific contributions (SC) of the dissertation are as follows:

- **SC 1 - Novel methodologies for data cleaning, data fusion, and feature selection**: A formal definition of data fusion of heterogeneous streaming data sources

and development of autonomous methodologies for online data cleaning (based on Kalman filter), fusion, and enrichment of heterogeneous data streams. Development of a novel algorithm for feature selection based on information theory and the multi-objective optimization.

- **SC 2 - Extension of lambda architecture**: An extension of big data lambda architecture for hybrid model development and deployment.
- **SC 3 - Evaluation in real-world scenarios**: Evaluation of the proposed architecture and methods in several real-world scenarios from energy management, water management, smart cities, transport to Earth observation. Introduction of incremental learning in the field of water management.

1.4 Organisation of the Thesis

Thesis begins with an introduction, motivation and an overview of the aims, hypotheses and scientific contributions. Along the published journal papers, the latter are presented in Figure 1.1.

Chapters 2 through 5 present five papers published in peer review journals. Each chapter gives a short introduction to the paper presented in the same format as published in the journal. Chapter 2 is dedicated to the first step in the overall picture (see Figure 1.1) - data cleaning [**SC1**]. The chapter presents a paper titled *Autonomous Sensor Data Cleaning in a Stream Mining Setting* [3]. In Chapter 3, we proceed with the introduction of online heterogeneous data fusion tailored for machine learning applications [**SC1**]. The paper titled *Streaming data fusion for the internet of things* [2] represents the cornerstone of the thesis. In order to efficiently use data fusion, an efficient feature selection is needed. A novel feature selection algorithm called FASTENER, based on genetic algorithms and a multiobjective optimization approach, is presented in Chapter 4 [**SC1**] alongside the paper titled *FASTENER feature selection for inference from earth observation data* [5]. Next, in Chapter 5, we present an implementation of the framework based on architecture and components presented in this thesis [**SC2**] as well as the final piece in the mosaic, the usage of incremental learning techniques in the platform. The framework is described in depth in the paper titled *Computer Architectures for Incremental Learning in Water Management* [1], whereas incremental learning techniques have been tested in *Usage of statistical modeling techniques in surface and groundwater level prediction* [20].

Evaluation of the methodologies in the real-world scenarios [**SC3**] is given in each of the papers presented from Chapter 2 through 5.

Chapter 6 concludes the thesis by looking back at the main scientific contributions and clearly stating objectives and directions for future work and research on the topic of pre-processing of heterogeneous data streams for Internet of Things applications.

Chapter 2

Autonomous Data Cleaning on a Stream

Garbage in, garbage out.

George Fuechsel

In this chapter, we introduce the paper entitled *Autonomous Sensor Data Cleaning in a Stream Mining Setting*, authored by Klemen Kenda and Dunja Mladenić. This paper has been published in the Business Systems Research Journal [3]. Klemen Kenda contributed to the conceptualization, methodology, software development, evaluation, and visualization. Furthermore, he took the lead in writing the paper.

Data streams originating from the IoT are inherently prone to various inconsistencies and flaws. Autonomous data cleaning engine is a prerequisite for an efficient real-world implementation of a machine learning pipeline and represents the first building block after data ingestion of such a pipeline (see Figure 1.1).

Our approach to online data cleaning is based on the utilization of the Kalman filter. In the context of machine learning, the Kalman filter can be regarded as an incremental short-term predictive model. The fundamental premise underlying the Kalman filter is the concealment of the true state of the system from the observer. Consequently, the filter's primary objective is to deduce the hidden actual state of the system based on the observed states. Both the hidden state and its dynamics are stipulated by the user and serve as foundational components of the filter. With new data incoming, the Kalman filter incrementally adjusts its parameters, aligning with the user-defined criteria. The filter operates across two distinct phases.

In the first phase, also known as the *projection* phase, the filter forecasts the forthcoming hidden state of the system along with the corresponding covariance matrices. This prediction facilitates the data cleaning system in establishing an interval that includes the next expected data point. If the subsequent measurement falls outside this interval, it is flagged as an outlier and the predicted value takes its place.

The methodology has been tested on 9 artificial labeled datasets as well as 5 unlabeled datasets (two synthetic datasets and three real data sets of groundwater levels, server load, and from smart grids). In the labeled artificial datasets, our Kalman filter method produced an average F_1 score that was 0.03 lower than the score achieved by the ARIMA method. The method's success is remarkable, given that it is incremental and only considers past data. An indirect evaluation approach was used to estimate the effectiveness of the data

cleaning process for the unlabeled datasets. The evaluation was based on the improvement observed in modeling results when using the cleaned data. For 3 real data sets (ground water, server load and smart grids) the regression models improved on average in more than 66% of the time series.

sciendo

Autonomous Sensor Data Cleaning in Stream Mining Setting

Klemen Kenda, Dunja Mladenić
Jožef Stefan Institute, Ljubljana, Slovenia
Jozef Stefan International Postgraduate School, Ljubljana, Slovenia

Abstract

Background: Internet of Things (IoT), earth observation and big scientific experiments are sources of extensive amounts of sensor big data today. We are faced with large amounts of data with low measurement costs. A standard approach in such cases is a stream mining approach, implying that we look at a particular measurement only once during the real-time processing. This requires the methods to be completely autonomous. In the past, very little attention was given to the most time-consuming part of the data mining process, i.e. data pre-processing. **Objectives:** In this paper we propose an algorithm for data cleaning, which can be applied to real-world streaming big data. **Methods/Approach:** We use the short-term prediction method based on the Kalman filter to detect admissible intervals for future measurements. The model can be adapted to the concept drift and is useful for detecting random additive outliers in a sensor data stream. **Results:** For datasets with low noise, our method has proven to perform better than the method currently commonly used in batch processing scenarios. Our results on higher noise datasets are comparable. **Conclusions:** We have demonstrated a successful application of the proposed method in real-world scenarios including the groundwater level, server load and smart-grid data.

Keywords: big data, autonomous processing, real-world applications, data cleaning, stream mining, water management, data-centre management, smart-grids
JEL classification: C55, C81, C63, C67
Paper type: Research article

Received: Jan 31, 2018
Accepted: Apr 21, 2018

Citation: Kenda, K., Mladenić, D. (2018), "Autonomous Sensor Data Cleaning in Stream Mining Setting", Business Systems Research, Vol. 9, No. 2, pp. 69-79.
DOI: 10.2478/bsrj-2018-0020

Acknowledgments: This work was supported by the Slovenian Research Agency and the ICT program of the EC under project OPTIMUM (H2020-MG-636160) and Water4Cities (H2020-MSCA-RISE-734409).

Introduction

Big Data is a term that is used for datasets that are too large in size and complexity to be handled with the current methodologies (Fan et al., 2013). The meaning of this definition changes constantly with the development of technology and advances in computer science. However, translating the data analysis into a streaming on-line

process is always considered a good approach. Stream mining exposes another benefit of the methodology - real-time responsiveness of the system, which has been identified as desirable by many different authors regarding reporting (Belfo et al., 2015), intrusion detection (Al Quhtani, 2017) and others.

The field has received a lot of attention. Many stream modelling (regression, classification, clustering etc.) and evaluation methods have been developed. However, some data mining process phases as identified in the cross-industry standard process for data mining (CRISP-DM) methodology (Shearer, 2000), have been left aside (Kandel et al., 2011; Krempl et al., 2014). One of those phases, which data cleaning is a part of, is "Data preparation" and is crucial for real-world data mining applications (Zekić-Sušac et al., 2015).

Even in classical data mining task, where all the data is available beforehand, the practitioners claim that data preparation takes up to 80% of the time (Press, 2016). A lot of work is done manually. In stream mining scenario there is no possibility for a constant human intervention, all the data pre-processing needs to be completely autonomous.

Data cleaning represents the first step in data pre-processing. It represents a permanent challenge in data analytics. If not done or badly performed it can result in inaccurate predictions and later in unreliable business decisions. The issue has been tackled recently both by industry and academia, mostly to address the issues of scalability (Big Data), interfaces, new abstractions and statistical techniques (Chu et al., 2016).

The field of time-series analysis has been lively for a number of decades. Kalman published his work on linear filtering already in 1960 (Kalman, 1960). Kalman stands out of the *crowd* due to the successful application of the equations to trajectory estimation in the NASA Apollo space program. Different applications have been reported since then and the field of time-series analysis has been *reinvented* in correspondence with advances in computer science and technology. In the last years many applications were created for on-line streaming data analysis.

Outlier detection in time series has been thoroughly discussed already in 1993 by Chen and Liu (1993). The paper identifies five different types of time series outliers: (1) Additive Outlier (AO), (2) Innovation Outlier (IO), (3) Level Shift (LS), (4) Temporary Change (TC) and (5) Seasonal Level Shift (SLS). Authors propose usage of different models from ARIMA family (AR, MA, IMA, Seasonal IMA) for outlier detection, using its short-term prediction capabilities.

To the best of our knowledge the usage of Kalman filter for cleaning of streaming sensor data has firstly been proposed in our work (Kenda et al., 2013). The paper proposed an algorithm for additive outlier detection in a stream mining setting using short-term prediction based on Kalman filter. The very same idea has been proposed in (Xu, 2015), where it has been studied in depth and extended to a wider context. The authors coined the methodology as time series Kalman filter (TSKF). The method has been improved in (Kenda et al., 2017), where we proposed the usage of unsupervised machine learning approach for automatic parameter fine-tuning and tested the method on an artificial data set. In the current work we further extend the methodology by introducing the indirect modelling-based evaluation procedure and extensive testing on 5 real-world data sets.

Recently, literature is examining other potential Kalman filter extensions for data cleaning. For example, (Marczak et al., 2018) studies usability of augmented Kalman filters (AKF).

The paper is structured as follows. "Methodology" section describes Kalman filter algorithm and how it was implemented in our methodology. In the "Results" section

we provide evaluation of our methodology on artificial and real-world datasets. We also describe the indirect evaluation procedure. Next, we discuss the usability of our methodology in real-world scenarios and compare it to current state-of-the-art in batch setting. Finally, we conclude the paper.

Methodology

The notion of additive outlier

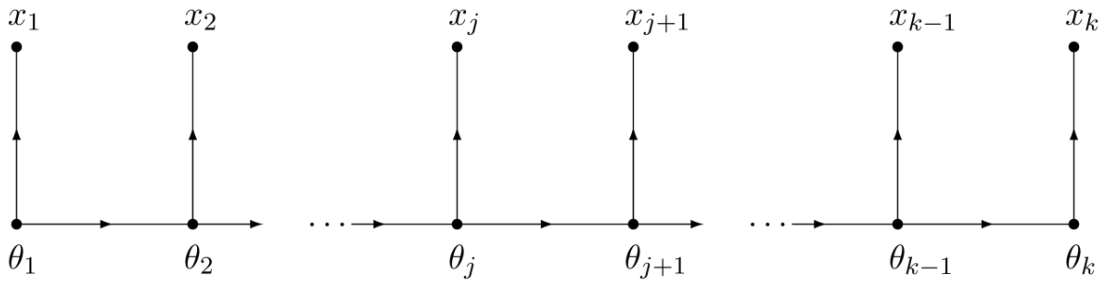
Additive outlier is a point outlier, which occurs at a given timestamp t_j and affects a single observation. In sensor data such outliers can be a consequence of a sudden change in ambient conditions, communication glitch or some similar unexpected event. With sensor measurements we assume that they arrive much faster than the data changes.

We propose a method with short-term prediction, based on previous measurements. Short term prediction is compared to the new measurement and classified as an outlier if the difference exceeds a specified threshold. As proposed in (Kenda et al., 2013) we introduce a safe guard to overcome a potential instability of the algorithm and enlarge the threshold in case that the detected outlier is a false positive, which might be an indication of a sudden concept drift in the data.

Kalman Filter

Kalman filter is a very suitable algorithm to be applied to data cleaning in a streaming scenario. It is an on-line algorithm that can produce short term predictions and even calculate covariance error matrix (used to calculate a threshold for outlier classification). Algorithm assumes that our process can be described as a Gauss-Markov process.

Figure 1
Diagram of Gauss-Markov process

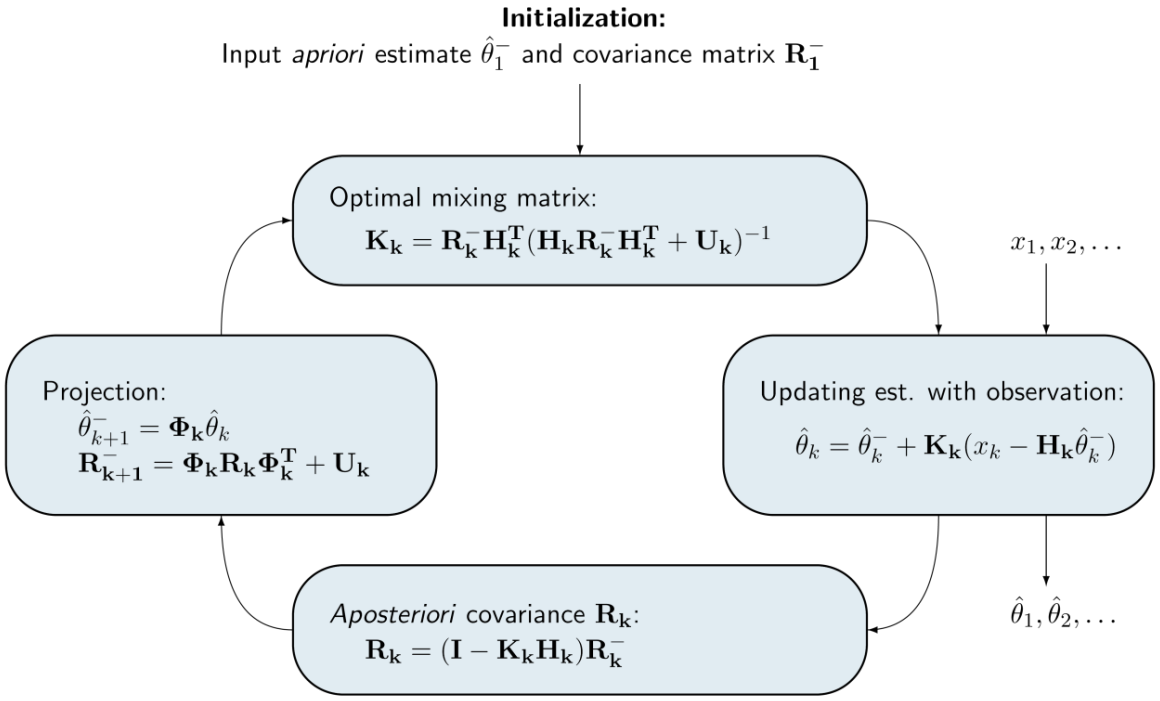


Source: (Kenda et al., 2017)

The process is depicted in Figure 1. Arrows from internal state θ_j to another internal state θ_{j+1} depict transitions (transition equation) and arrows from internal state θ_j to observation x_j depict observation equations. The process has two properties:

- Every consequent internal state θ_{j+1} only depends on a prior internal state θ_j. Both states are connected through transition matrix $\boldsymbol{\Phi_j}$.
- Each internal state θ_j can be inferred through its observation x_j, which is linked to the internal state via observation matrix $\boldsymbol{H_j}$ and is a subject of Gaussian noise.

Figure 2
Kalman filter application cycle



Source: (Kenda, Mladenić, 2017)

In general, matrices $\boldsymbol{H_j}$ and $\boldsymbol{\Phi_j}$ can change over time, but in our case they remain the same as we assume the underlying process does not change through time. Kalman filter equations are depicted in Figure 2.

Kalman filter application cycle starts with initialization of *a priori* estimates for internal state θ_1^- and covariance matrix $\boldsymbol{R_1^-}$. With each new observation x_j the state and covariance matrix get updated. The next phase is dedicated to short-term one step ahead prediction (projection). Finally, optimal new mixing matrix gets calculated (responsible for optimal updating of the projected state with an observation). $\boldsymbol{U_k}$ represents normal distribution variance noise matrix.

Computational complexity of our implementation of Kalman filter is $O(n^3)$ where n is the dimension of internal state space. In the proposed 2nd degree model the number of internal state components is $n = 3$.

Parameter Learning

Initialization of Kalman filtering algorithm can be very demanding and there can be many free parameters involved, depending on the observation and transition matrix dimensions. Usage of expectation maximization (EM) algorithm (Dempster et al., 1977; Xu, 2015) can yield estimates for the initial internal state of the system and corresponding covariance matrices. Clean initial dataset is needed to obtain these parameters.

In our experiments with time series data the results from EM algorithm have not provided good results (confidence into last state was exaggerated), therefore we propose an additional data-oriented approach. EM calculates estimates of the following parameters: *a priori* initial state θ_1^-, transition covariance $\boldsymbol{Q}$, observation covariance $\boldsymbol{R_k}$ and initial state covariance $\boldsymbol{R_1^-}$. We propose multiplying EM estimates with an additional factor in order to minimize F_1 score of outlier classification on a

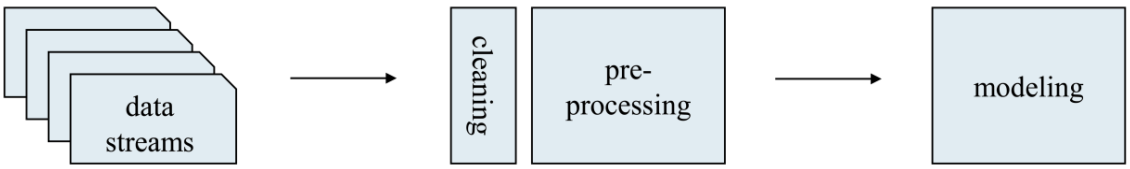
labelled dataset. Parameters can be obtained by a grid search over a predefined multiplier space.

Grid search is time consuming, but it can find configurations which result in much smoother model that better follows the underlying dynamic processes in the data. We have implemented exhaustive and randomized grid searches in our solution, reported results are based on the randomized version.

Streaming Sensor Data Platform with Data Cleaning

We propose the usage of the filter at the lowest possible level in the pre-processing platform. The data-cleaning component should be implemented at the entry point of a particular data source to the pre-processing platform (see Figure 3). Clean data is then inserted into stream pre-processing engine, which is in charge of data enrichment and heterogeneous data fusion and finally this data is pushed into the appropriate stream modelling method. Cleaning at this level uses only autoregressive features. On a higher level, however, data-cleaning, which takes advantage of data fusion, could be used.

Figure 3
Position of data-cleaning system within the stream-mining analytical platform



Source: (Kenda, Mladenić, 2017)

Results

We tested our results on artificial and real-world data sets. Functionality of the algorithm is illustrated in Figure 4. It shows the impact of Kalman filter's short-term prediction and its variance on additive outlier detection. The measurement (depicted in dark blue) that falls outside the admissible interval around short term prediction (depicted in light blue) is considered an outlier.

Results on Annotated Artificial Data Set

We provide an artificial dataset, following the usual daily profile of a family of typical sensors. Each time-series in the dataset introduces a different level of Gaussian noise $N(\mu = 0; \sigma)$. We have made the dataset publicly available at ResearchGate (Kenda, 2017). Data points are a subject of noise, 1% of data points have been considered as candidates for an additive outlier. Amplitude of additive outliers has been uniformly sampled on the interval from 0 to $0.714 \cdot \max(f(t))$, where $\max(f(t))$ is the maximum value of the underlying dynamics function. Amplitudes that were lower than $2 \times \sigma$ have been dismissed.

Artificial set experimental results are depicted in Table 1. Different data sets (from 1 to 9) introduce different Gaussian noise, which makes it more and more difficult to correctly classify the outliers, which can be observed in decreasing values of precision, recall and F_1 in Table 1. As expected, ARIMA (batch) method gives slightly better results than Kalman (streaming) method. F_1 scores are similar, whereas ARIMA method is optimized towards better precision and Kalman towards better recall.

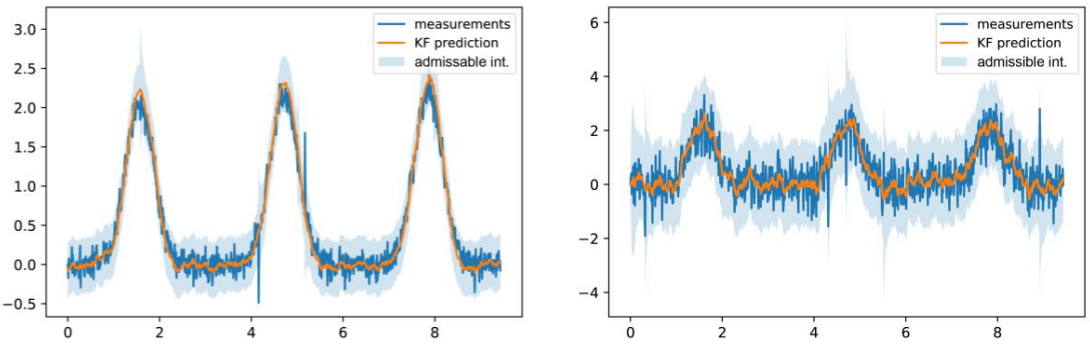
Figure 4 shows algorithm results with 2 different datasets: left - little noise ($\sigma = 0.036$), right - more noise ($\sigma = 0.179$). Kalman filters' short-term prediction is depicted in orange, measurements in dark blue. Any measurement outside of the admissible light-blue interval (defined by Kalman filter variance) is considered as an outlier.

Table 1
Comparison of Kalman filter additive outlier detection results with current batch methodology (Chen et al., 1993)

		Kalman filter method			ARIMA method		
Dataset	**Noise σ**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
1	0.036	0.866	0.967	0.914	0.624	0.874	0.728
2	0.071	0.776	0.983	0.867	0.940	0.829	0.881
3	0.107	0.737	0.872	0.799	0.906	0.750	0.821
4	0.143	0.681	0.946	0.792	0.944	0.740	0.830
5	0.179	0.695	0.592	0.640	0.902	0.643	0.751
6	0.213	0.455	0.873	0.598	0.896	0.520	0.658
7	0.250	0.587	0.373	0.456	0.790	0.448	0.571
8	0.286	0.435	0.779	0.558	0.816	0.461	0.589
9	0.321	0.353	0.545	0.428	0.741	0.336	0.462

Source: (Kenda et al., 2017).

Figure 4
Illustration of the algorithm results with 2 different datasets: lower noise (left) and higher noise (right); measurements outside the admissible intervals are detected as outliers



Source: (Kenda, Mladenić, 2017)

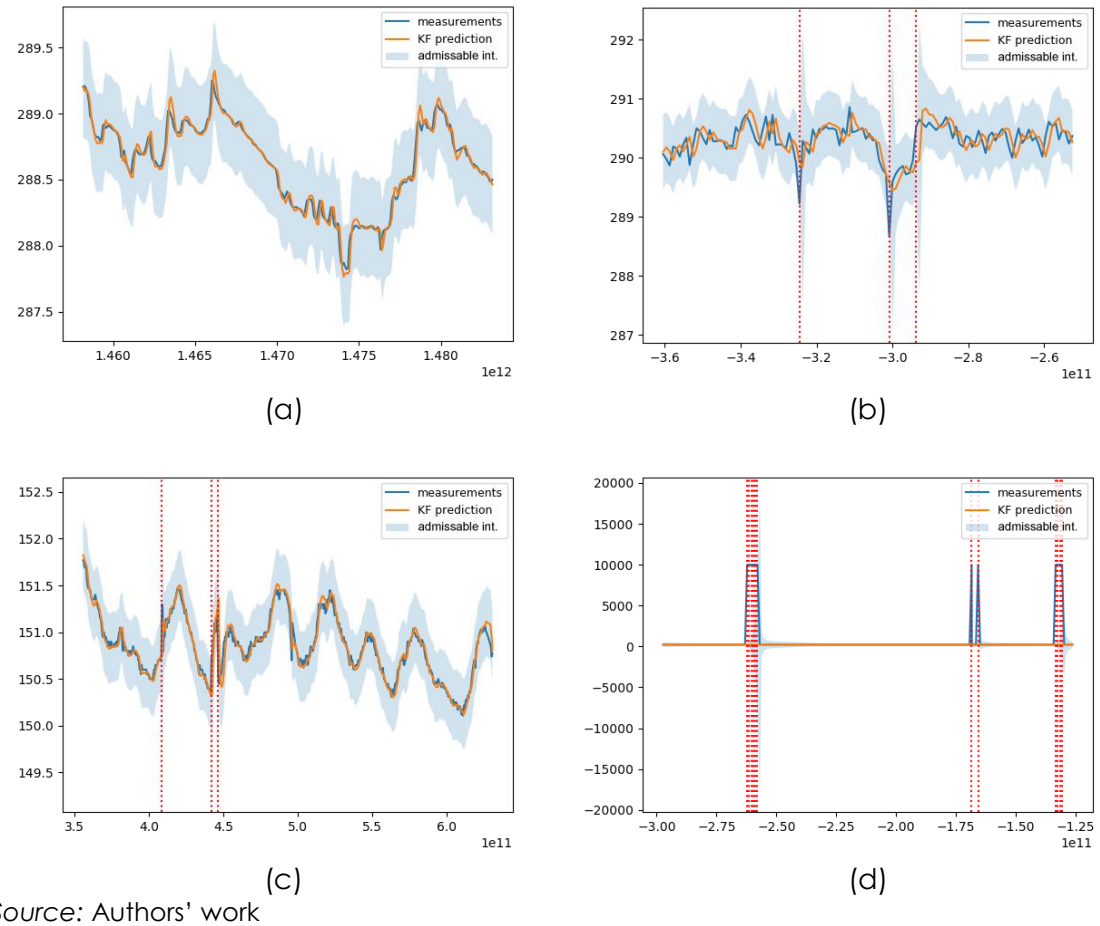
Results on Real-world Data Sets

There are two major problems concerning real-world sensor data sets: (1) these data sets are not annotated, therefore it is impossible to calculate proper accuracy measures of a data cleaning algorithm, (2) without accuracy measures it is also impossible to apply machine-learning techniques for parameter learning.

To overcome these shortcomings, we need to take a look into characteristics of sensor data. We have observed in many sensor data sources that outliers are rare. Most of the data is clean. It is therefore easy to introduce artificial outliers into original data and use such augmented data set to solve the problem (2). With the algorithm we are able to learn adequate parameters for a successful application of the algorithm. Solving problem (1) is more difficult. We can apply human-based anomaly classification for the rare detected outliers, which enables us to calculate

precision (is detected outlier really an outlier?). The second method is to compare modelling performance (i.e. regression) between the clean and the original datasets.

Figure 5
Illustration of the algorithm results with underground water level dataset: (a) time-series without outliers, (b) and (c) time-series with true and false positive outliers, (d) time-series with obvious outliers



Source: Authors' work

We have analysed performance of our method on 340 time-series data sets of groundwater levels from Slovenia. Results are depicted in Figure 5. Y-axis depicts groundwater levels in meters above sea level, x-axis depicts unix timestamp. Figure 5(a) shows a smooth and clean time series, which is easy to model with Kalman filter. The algorithm successfully identifies even bigger shifts in the groundwater levels. Figures 5(b) and (c) show sensors with more noise. The timestamps where potential outliers were detected are marked with a vertical red dotted line. We can observe two true positives (first two outliers) and one probable false positives in Figure 5(b), which is a consequence of a fast change in the data and is difficult to model in an on-line setting. Similarly, we can notice one true and two false positives in Figure 5(c). Figure 5(d) depicts extreme errors in the data that get detected correctly, even in cases, where there is more than one consecutive noisy measurement present.

Indirect Evaluation of Data Cleaning with Modelling Results

Without a labelled dataset from real-world scenarios, we cannot directly estimate the effect of data cleaning. Thus we are estimating the benefits of data cleaning through observation of the improvements of machine learning models on the data. It has been previously shown that data cleaning can significantly improve the model accuracy (Krishnan et al., 2016). We have compared root mean squared error (RMSE) of ARIMA (1, 1, 0) models on raw and on cleaned datasets. Lower RMSE measure means better fit of the models to the dataset.

Furthermore, we have developed a meta-classification algorithm for time-series to detect suitable candidates, where RMSE can be improved. Based on the meta-data obtained from the time-series (such as variance, mean data frequency, Kalman filter parameters, confidence of the Kalman model, etc.) and from the data cleaning algorithm learning phase, such as (learning parameters, number of errors, length of data frame and cleaning model score), we were able to build a classifier, which can predict whether our cleaned time-series can be modelled worse, better or equally good on cleaned data. The classifier has been built using the random forests algorithm (Breiman, 2001).

Experiments have been conducted on 5 different datasets: (i) 340 time-series of groundwater levels in Ljubljana region, (ii) 67 time-series from Yahoo! A1 Server Load (Yahoo! Webscope, 2015), (iii) 400 time-series from smart-grid observations (active power) in SW Slovenia, (iv) and (v) 100 synthetic time-series from Yahoo! anomaly detection benchmark. Results are depicted in Table 2. Table presents KPIs related to the algorithm and the meta-classifier performance as follows. *Improvement* indicates fraction of time-series with better fit after cleaning (0.805 means that 80.5% of time-series benefited from the proposed data cleaning). RMSE ratio expresses ratio of improvements of RMSE against the losses (443.6 indicates that RMSE is improved much more than it deteriorates in cases, where data cleaning fails; this happens as groundwater data contains significant human-made errors). Precision, recall and F_1 are standard classifier evaluation measures for our meta-classification algorithm.

Table 2
Algorithm performance on unlabelled data and prediction of the meta-classifier regarding the success of the algorithm

	Algorithm performance		Classification performance		
Dataset	**Improvement**	**RMSE ratio**	**Precision**	**Recall**	**F_1**
Groundwater	0.513	443.6	0.737	0.737	0.737
Server load	0.530	1.400	0.746	0.740	0.739
Smart-grid	0.805	1.270	0.850	0.861	0.850
Yahoo! A2 (synthetic)	1.000	N/A	1.000	1.000	1.000
Yahoo! A3 (synthetic)	0.000	N/A	N/A	N/A	N/A

Source: Authors' work

The most illustrative are results on the two synthetic datasets. On the first dataset (Yahoo! A2) our algorithm works perfectly, while on the second dataset (Yahoo! A3) it fails completely. The main difference between these two datasets is that the periodicity in the first dataset is much larger and noise is much lower. The same properties are illustrated on real-world datasets, where we see the best performance (80.5%) of the algorithm on a smart-grid dataset. Typical period in this dataset is one day and measurements are taken every 15 minutes. Groundwater (i) and server load (ii) datasets have a sampling interval much closer to the typical period (significant

change in the data can happen within a single sampling interval, i.e. groundwater can rise significantly in a day with substantial amount of rainfall). Performance of our algorithm is 51.3% and 53.0%, respectively.

Usability of the cleaning algorithm was further improved with a meta-classifier. Based on time-series metadata the classifier is able to identify the data sources which are likely to improve with our algorithm with a precision, that is much higher than the improvement ratio (between 73.7% and 85.0%).

Discussion

As presented in the previous section our algorithm achieves the best performance with a typical stream of sensor data, as we can find in Internet of Things. In such scenarios sensor measurements are frequent and systematic changes in the data are low (sampling interval is much shorter than periodicity). In comparison with a commonly used ARIMA methodology in batch data pre-processing (Chen et al., 1993), our method works better with lower noise data. An obvious downside of the ARIMA methodology is that it requires fitting of ARIMA model to the whole dataset, which makes it unusable with data streams.

Our approach is applicable in any kind of streaming scenario. However, there are some additional restrictions that need to be considered. When testing on real-world dataset we have observed heterogeneous characteristics of sensor data with respect to noise, volatility and measurement intervals. When dealing with large and diverse amounts of sensors (nowadays it is not unusual to have more than 10.000 sensors in the system, i.e. in a regional smart-grid system) it is not feasible to do individual cleaning model learning, therefore some basic clustering of sensors into groups with similar properties is needed. Fine tuning of the parameters can be performed on a representative time-series only and then applied to the whole cluster.

Based on their characteristics efficiency of our methodology differs between the datasets. However, efficiency of the algorithm can be further improved with a classification algorithm on the top of time-series/learning-phase metadata, which is able to select a suitable time-series for the data-cleaning algorithm. In this way we were able to achieve precisions between 73-85%.

Conclusion

In this paper we have identified that efficient data pre-processing is very important in streaming data scenarios. We have focused on the first part of the data pre-processing pipeline: data cleaning. We conducted a short research on the state-of-the-art in the field and proposed our own method based on Kalman filter. The method has been quantitatively tested on an artificial data set. We have compared our method to the ARIMA state-of-the-art method and have obtained better results on the datasets with lower noise ratio and comparable results on the datasets with higher noise ratio. The main advantage of our method is, that it can work with Big Data in a streaming scenario.

Additionally, we have applied our method to a heterogeneous set of real-world time-series. We have tested the efficiency of our cleaning method with an indirect approach, where we tried to fit an ARIMA model to raw data and to clean data to compare the respected error measures. The proposed data cleaning was shown to be beneficial on time-series that have properties like majority of sensor streams available in the IoT domain. We also developed a meta-classification method which can predict the success of the data cleaning with 75%-85% precision.

By observing differences in Yahoo! A2 and Yahoo! A3 datasets we identified the major limitation of our algorithm. When changes in a time-series are rapid (i.e. if periodicity is short in comparison to measurement frequency) many valid measurements are classified as outliers and algorithm accuracy is low. Future work should therefore be directed into improving Kalman filter parameter fine-tuning procedure, which should capture such behaviour. Additionally, usability of the algorithm should be tested on different real-world datasets and in the production environment.

References

1. Al Quhtani, M. (2017), "Data Mining Usage in Corporate Information Security: Intrusion Detection Applications", Business Systems Research, Vol. 8, No. 1, pp. 51-59.
2. Belfo, F., Trigo, A., Estébanez, R. P. (2015), "Impact of ICT Innovative Momentum on Real-Time Accounting", Business Systems Research, Vol. 6, No. 2, pp. 1-17.
3. Breiman, L. (2001), "Random Forests", Machine Learning, Vol. 45, No. 1, pp. 5-32.
4. Chen, C., Liu, L. (1993), "Joint Estimation of Model Parameters and Outlier Effects in Time Series", Journal of the American Statistical Association, Vol. 88, No. 421, pp. 284-297.
5. Chu, X., Ilyas, I. F., Krishnana, S., Wang, J. (2016), "Data cleaning: Overview and emerging challenges", in Özcan, F., Koutrika, G. (Eds.), Proceedings of the 2016 International Conference on Management of Data, ACM, San Francisco, pp. 2201-2206.
6. Dempster, A. P., Laird, N. M., Rubin, N. M. (1977), "Maximum likelihood from incomplete data via the EM algorithm", Journal of Royal Statistical Society Series B, Vol. 39, No. 1, pp. 1-38.
7. Fan, W., Bifet, A. (2013), "Mining big data: Current status, and forecast to the future", ACM siGKDD Explorations Newsletter, Vol. 14, No. 2, pp. 1-5.
8. Kalman, R. E. (1960), "A new Approach to linear filtering and prediction problem", Journal of basic engineering, Vol. 82, No. 1, pp. 34-45.
9. Kandel, S., Heer, J., Plaisant, C., Kennedy, J., van Ham, F., Riche, N.H., Weaver, C., Lee, B., Brodbeck, D., Buono, P. (2011), "Research directions in data wrangling: Visualizations and transformations for usable and credible data", Information Visualization Journal, Vol. 10, No. 4, pp. 271-288.
10. Kenda, K. (2017), Artificial data-set for testing time-series additive outlier detection methods, available at: https://www.researchgate.net/publication/317721142_Artificial_data-set_for_testing_time-series_additive_outlier_detecion_methods (18 February 2018).
11. Kenda, K., Mladenić, D. (2017), "Autonomous on-line outlier detection framework for streaming sensor data", in Zadnik Strin, L., Kljajić Borštnar, M., Žerovnik, J., Drobne, S. (Eds.), Proceedings of the 14th International Symposium on Operational Research, Bled, pp. 103-108.
12. Kenda, K., Škrbec, J., Škrjanc, M. (2013). "Usage of Kalman Filter for Data Cleaning of Sensor Data", in Gams, M. (Ed.), Proceedings of the 16th International Multiconference Information Society – IS 2013, Ljubljana, pp. 172-175.
13. Krempl, G., Žliobaite, I., Brzezinski, D., Hüllenmeier, E., Last., M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopolou, M. (2014), "Open challenges for data stream mining research", ACM siGKDD Explorations Newsletter, Vol. 16, No. 1, pp. 1-10.
14. Krishnan, S., Wang. J., Wu, E., Franklin, M. J., Goldberg, K. (2016), "ActiveClean: interactive data cleaning for statistical modeling", in Chaudhuri, S., Haritsa, J. (Eds.), Proceedings of the VLDB Endowment, Vol. 9, No. 12, pp. 948-959.
15. Marczak, M., Proietti, T., Grassi, S. (2018), "A data-cleaning augmented Kalman filter for robust estimation of state space models", Econometrics and Statistics, Vol. 5, pp. 107-123.
16. Press, G. (2016), "Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says", available at: https://www.forbes.com/sites/suntrustprivatewealth/2017/12/21/wealth-transfer-are-you-sure-your-beneficiaries-are-prepared/ (31 January 2018).
17. Shearer, C. (2000), "The CRISP-DM model: the new blueprint for data mining", Journal of

data warehousing, Vol. 5, No. 4, pp. 13-22.
18. Xu, S. (2015), "Data Cleaning and Knowledge Discovery in Process Data", PhD thesis, University of Texas, Austin.
19. Yahoo! Webscope (2015), "S5 - A Labeled Anomaly Detection Dataset, version 1.0", available at: http://research.yahoo.com/Academic_Relations (28 February 2018).
20. Zekić-Sušac, M., Has, A. (2015), "Data Mining as Support to Knowledge Management in Marketing", Business Systems Research Journal, Vol. 6, No. 2, pp. 18-30.

About the authors

Klemen Kenda is a Ph. D. candidate at Jožef Stefan International Postgraduate School and a researcher at Artificial Intelligence Laboratory at Jožef Stefan Institute in Ljubljana, Slovenia. He received his diploma degree from the Faculty of Mathematics and Physics at University of Ljubljana, with a thesis "Usage of machine learning techniques with analysis of the data from ATLAS detector". His main research interests are in information and communication technologies. He is focused on data pre-processing, data fusion, machine learning and data-driven modelling in the context of streaming big data. He is actively engaged in various science, industrial and start-up projects (H2020, cooperation between research and industry, energy efficiency start-up). The author can be contacted at **klemen.kenda@ijs.si.**

Dunja Mladenić works as a researcher and a project leader at Jožef Stefan Institute, Slovenia, leading Artificial Intelligence Laboratory and teaching at Jožef Stefan International Postgraduate School, University of Ljubljana and University of Primorska. She has extensive research experience in study and development of machine learning, data/text mining, semantic technologies, sensor data analysis methods and their application on real-world problems. She has published papers in refereed journals and conferences, coedited several books, served on program committees of international conferences and organized international events. She serves as a project evaluator of project proposals for European Commission and USA National Science Foundation. She served on the Institute's Scientific Council (2013-2017) as a vice president (2015-2017). She serves on executive board of Slovenian Artificial Intelligence Society SLAIS (as a president (2010-2014)) and as an advisory board member of ACM Slovenija. The author can be contacted at **dunja.mladenic@ijs.si.**

Chapter 3

Heterogeneous Streaming Sources Data Fusion for Machine Learning

> Data fusion is like a marriage: integrating data sources together requires patience, compromise, and a common language.
>
> *Kirk Borne*

This chapter presents the paper titled *Streaming data fusion for the internet of things* by Klemen Kenda, Blaž Kažič, Erik Novak and Dunja Mladenić and presents the centerpiece of this thesis. The paper was published in the Sensors journal (IF 2019: 3.275) [2]. Klemen Kenda is the main author of the paper. He contributed to conceptualization, methodology, software development, validation, formal analysis, data curation, writing, visualization, project administration and funding acquisition.

In the context of the Internet of Things (IoT), data fusion, also known as sensor data fusion or information fusion, refers to the process of combining and integrating data from multiple sensors or sources to create a more accurate, comprehensive, and meaningful representation of the environment or system being monitored.

The main goal of data fusion in IoT is to extract valuable insights, improve decision-making, and enhance the overall efficiency of IoT applications. By fusing data from multiple sensors, the resulting information can provide a more accurate understanding of the environment, which can result in better modeling capabilities such as creating predictions or anomaly detection, and a more holistic view of the system's behavior.

There are several types of data fusion techniques used in IoT that vary from local to more global approaches and are based on the JDL model [21]:

1. **Sensor-level Fusion:** This involves combining raw sensor data at the individual sensor level to reduce noise and errors. It can include techniques like filtering, calibration, and data alignment.

2. **Feature-level Fusion:** In this approach, extracted features from the data of different sensors are combined to create a more informative representation. This can help in reducing data dimensionality and improving the efficiency of analysis.

3. **Decision-level Fusion:** Here, the outputs or decisions from multiple sensors are combined to make a more reliable decision. This technique is often used to improve accuracy and robustness in applications like object tracking or localization.

4. **Information-level Fusion:** This is the highest level of fusion, where higher-level knowledge or context is integrated into the data to create a comprehensive understanding of the situation. This can involve incorporating external data sources, historical data, or expert knowledge.

Our methodology focuses on feature-level fusion and is ignorant of any high-level semantic information regarding the observed system, such as sensor metadata or even the system's knowledge graph [2]. The purpose of the research was to create a useful component in the lambda architecture that would enable fast deployment of real-world IoT applications of machine learning, such as predictive modeling or anomaly detection.

The paper provides a formal definition of heterogeneous data streams fusion, a working streaming data fusion framework, and conceptual architecture of the system. In the lambda architecture, the data fusion component is located immediately after data cleaning (see Figure 1.1).

The methodology is able to ingest three types of data: streaming sensor data, updating predictions (such as weather forecasts) and static data (such as data describing human behavior, for example working days). Three algorithms are presented that enable the framework to generate enriched data streams and fuse them locally and globally.

The system has been deployed in both a cloud environment and an edge device. Validation of the methodology's efficiency has been accomplished through cloud deployment, using public train data from edge devices on the smart-grid dataset. In addition, a performance evaluation was conducted on the traffic data set.

The findings strongly support the advantages of employing data fusion techniques. Enhanced data streams have demonstrated a higher model accuracy when compared to nonenriched streams. Moreover, the use of multiple data streams that provide contextual information has yielded superior model outcomes compared to relying solely on single-source data streams.

In the train scenario, when predicting electricity consumption for a prediction horizon of 10 seconds on the edge device on a train, the model accuracy increased from $R^2 = 0.62$ (when only using additional auto-regressive features) to $R^2 = 0.8$ using random forest regression with the top 20 correlated features. In the smart grid scenario, the most successful predictive algorithm was also the random forest. The prediction horizon was set to 10 hours. Performance improved from $R^2 = 0.90$ for models using only auto-regressive features to $R^2 = 0.93$ with the top 20 features. In the traffic data set, we predicted the energy demand for an electric train power station 10 hours in advance. Random forest performance improved from $R^2 = 0.68$ to $R^2 = 0.80$.

The implemented system has been successfully deployed across two distinct environments: a cloud-based infrastructure and an edge computing device. In a cloud-based deployment, the methodology's efficiency has been validated on the smart grid dataset. On edge devices, deployed in public electric trains, the effectiveness of the methodology has been validated on the traffic data set.

The results confirm the benefits of using data fusion methods: enriched data streams yield better model accuracy than nonenriched ones, and streams from multiple data streams that describe more context yield better model results than single-source data streams. Additional validation of the overall system in water management scenario is also given in Chapter 5.

sensors

MDPI

Article

Streaming Data Fusion for the Internet of Things

Klemen Kenda [1,2,*], Blaž Kažič [1,2], Erik Novak [1,2] and Dunja Mladenić [1,2]

[1] Artificial Intelligence Lab, Jozef Stefan Institute, 1000 Ljubljana, Slovenia; blaz.kazic@ijs.si (B.K.); erik.novak@ijs.si (E.N.); dunja.mladenic@ijs.si (D.M.)
[2] Jozef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia
* Correspondence: klemen.kenda@ijs.si; Tel.: +386-1-477-3127

Received: 31 March 2019; Accepted: 22 April 2019; Published: 25 April 2019

Abstract: To achieve the full analytical potential of the streaming data from the internet of things, the interconnection of various data sources is needed. By definition, those sources are heterogeneous and their integration is not a trivial task. A common approach to exploit streaming sensor data potential is to use machine learning techniques for predictive analytics in a way that is agnostic to the domain knowledge. Such an approach can be easily integrated in various use cases. In this paper, we propose a novel framework for data fusion of a set of heterogeneous data streams. The proposed framework enriches streaming sensor data with the contextual and historical information relevant for describing the underlying processes. The final result of the framework is a feature vector, ready to be used in a machine learning algorithm. The framework has been applied to a cloud and to an edge device. In the latter case, incremental learning capabilities have been demonstrated. The reported results illustrate a significant improvement of data-driven models, applied to sensor streams. Beside higher accuracy of the models the platform offers easy setup and thus fast prototyping capabilities in real-world applications.

Keywords: data fusion; stream mining; machine learning; incremental learning; time-series analysis

1. Introduction

The scientific community has been discussing the rising amount of data originating from the internet of things (IoT) for more than a decade. The IoT reached the mass market in early 2014 and its ubiquitous influence and challenges are still permeating the scientific literature. The field of big data processing has improved drastically and a plethora of solutions for various IoT problems have reached their production stage [1].

The volume of the data keeps rising and as the technology is penetrating new markets (i.e., water management), new challenges are put in front of the industry and academia. The need for efficient and accurate analysis of these data is still an issue [2]. Stream processing [3] has been established as a potential answer to the analysis of big data and incremental learning has been rediscovered to answer some of the challenges (like concept drift [4] or learning efficiency [5]). While the field of incremental learning has matured through the last decade and a wide variety of algorithms have been described, tested and implemented in various software libraries, the applications of methodologies from a laboratory to the real world have been scarce. Throughout our work in various applications within the environmental domain, water management, traffic, energy efficiency and smart grid modeling, we have identified the following shortcomings: (i) the most comprehensive software library [6] for stream mining methods is an academic project and therefore requires an additional effort when migrating to production, (ii) modern stream mining frameworks (like Apache Spark, Flink and others) do not implement the state-of-the-art incremental learning methodologies or on-line data fusion strategies; it is also extremely difficult to find an operational implementation of an

advanced incremental learning regression, (iii) most of the scientific work on incremental learning has taken place inside the lab, emulating unreal (ideal) conditions, which are rarely encountered in the real world; mostly this remark applies to the data preparation step (including data fusion and generation of machine-learning-ready rich data streams).

Lack of on-line data pre-processing techniques also reduces the possibility of using hybrid approaches, where data pre-processing is done on-line and prediction models are implemented using traditional machine learning (ML) approaches. McKinsey has established that up to 40% of the data value emerging from the IoT is hidden within the synergy effects of different systems [7]. With the exception of the IoT Streaming Data Integration (ISDI) framework [8] which solves time alignment issues of data integration, a generic methodology for generation of feature vectors for machine learning approaches in the IoT scenario does not yet exist and this paper aims to fill this gap. The proposed framework offers a complete streaming methodology for building rich vectors, describing important process characteristics (or features), suitable for traditional or incremental machine learning algorithms. Throughout this document, we will refer to such rich vectors as feature vectors. The proposed methodology is able to merge data from a set of heterogeneous streaming data sources (i.e., from the IoT, weather forecasts and data about human behaviour) in a real-world setting. Our experiments show that this enables machine learning models to yield more accurate and thus more useful results.

In this paper we show use cases related to energy management and traffic, however, the methodology could be useful also in other scenarios such as: algorithmic trading, health care, production line monitoring, intrusion and fraud detection, traffic monitoring, vehicle and wildlife tracking, sports analytics, context-aware promotions and advertising, computer systems and network monitoring, predictive maintenance, geospatial data processing, public transport, public health management, efficient and cost-effective services.

Motivation and Contributions

For almost two decades the review papers on stream mining [9–14] have been identifying the need for proper pre-processing of the data for the needs of stream mining techniques. According to the related work mentioned above, this still remains an open issue. There are systems that enable fast processing or automated data retrieval or single-stream enrichment, however, ensuring semantically correct generation of rich feature vectors from multiple heterogeneous data sources in a streaming scenario has only been partially solved.

Based on our research experience with various applications of stream mining techniques, including prediction of energy consumption in public buildings and smart grids, traffic prediction, prediction of public train energy consumption, spot market price prediction, groundwater levels prediction and others, we have developed a novel approach to be implemented in real-world scenarios. Contributions of our work are as follows:

1. **A formal definition** of heterogeneous data streams fusion. We provide a rigorous mathematical definition of the problem, where we define data streams and operators needed to provide final results—rich feature vectors to facilitate accurate predictive modeling.
2. **A generic streaming data fusion framework** for heterogeneous data streams. To the best of our knowledge, we provide the first generic framework for generation of feature vectors from heterogeneous data streams which supports applications of machine learning techniques in a streaming scenario.
3. **A conceptual architecture** for real-world application of stream mining techniques on heterogeneous multi-sensor data streams. Our experiments extend beyond the laboratory environment and are integrated into real-world scenarios. We propose embedding of the stream fusion framework within big data lambda architecture and its use in the cloud and edge infrastructure.

4. **An improvement of modeling capabilities** of the real-world IoT systems. We demonstrate that the proposed approach improves modeling accuracies in various scenarios. We provide a result-based methodology for evaluation of stream fusion frameworks.

The rest of the paper is structured as follows. Related work is described in Section 2, which is followed by a rigid mathematical problem definition in Section 3. Architecture and methods, used to solve the identified problems, are described in Section 4, integration scenarios are presented in Section 5. Results from real-world use cases are reported and discussed in Section 6. Finally, the paper is concluded in Section 7, where also possible future work directions are presented.

2. Related Work

In this section we present a selection of recent use cases, where streaming data fusion has been applied with success. We differentiate between common streaming data fusion methodologies, which integrate domain-knowledge into models, from domain agnostic methodologies, like ours. Streaming data fusion is naturally extended with incremental learning techniques, where we give a basic overview of the state-of-the-art. Finally, we conclude the section with a presentation of academic and production-grade stream processing engines and an overview of comparable streaming data fusion platforms.

Sensor fusion is helpful for improving certain functionalities and model accuracy **in various domains**, i.e., in positioning and navigation [15–17], in activity recognition [18,19], in system monitoring and fault diagnosis [20–25], in transport [26], in health care [27] and in others.

In health care, for example, data fusion is used in IoT-enabled settings such as remote patient monitoring systems. Here, the patient is monitored with different body and environmental sensors, whose signals are processed and used to inform the doctors of the patients condition. Data fusion is used to combine the different signals on three distinct levels: raw level (fusion of raw sensor data), feature level (combining features provided by different methods), and decision level (combining decisions or confidences of medical experts). The fusion is also used for computing context awareness, which is among others used for assigning dynamic roles to doctors. Use of fuzzy logic in context awareness is discussed in [27]. Integration of our framework in a remote patient monitoring system could provide additional improvements (i.e., inclusion of historical data in combination with current values).

Most of the mentioned sensor fusion methodologies expect all the data to be coherent, available immediately and arriving in the correct order. In many localized systems this is the case, however, in the IoT scenarios, the availability of the data contributes to most of the problems. Access control plays an important issue in data management and is a vivid topic in the recent literature [28–30]. Our platform builds on top of mechanisms, described in the literature and can take advantage of recent findings, especially those related to streaming platforms. Rare contributions discuss handling of delayed or out-of-sequence measurements [31]. Many of the systems also incorporate significant domain knowledge (model-driven approaches) into the data fusion model (mainly into the Kalman filter's transition matrix) [32], by which the models lose their generalization potential. Frameworks that have the potential to be applied in various use cases need to be domain knowledge agnostic (purely data-driven), at least with the modeling algorithm [33]. In this sense, any machine learning algorithm acts as a data fusion model since it combines multiple indicators into a single prediction. The idea has been developed further by heterogeneous feature fusion machines [15] that consider mapping multidimensional feature vectors into 1-dimensional output by using classic kernel functions, such as linear, polynomial and Gaussian with different regression methods. With these approaches, the challenge of generating correct and expressive feature vectors to support accurate modeling remains unsolved.

Big data and stream pre-processing. In large data sets, where stream mining is the approach of choice, data pre-processing and reduction are becoming critical methodologies for knowledge discovery [14,34]. The authors identify the essential role of such methodologies in efficient machine learning systems. Crucial pre-processing functionalities include concept drift detection and adaptation,

missing data imputation, noise treatment, data reduction and efficient and accurate stream discretization algorithms (we refer to these operators as *stream aggregators* in this paper) and imbalanced learning. Automated data analysis is of no use if data pre-processing requires manual intervention [35]. The authors present adaptive pre-processing which benefits the final prediction accuracy on real sensory data. We also use the same evaluation strategy for our methodology, however, our methodology focuses on building rich feature vectors, whereas the paper addresses adaptation to concept drift in the input data stream.

Stream mining and incremental learning. Stonebraker and co-authors [36] have identified eight requirements of a stream processing engine (SPE) already in 2005. Among them are a requirement to handle stream imperfections (i.e., delayed or missing data), a requirement to integrate stored and streaming data as well as requirements to keep the data moving and process the data and respond instantaneously. Stream processing engines often base their modeling capabilities on incremental learning methodologies [3,37]. The most popular method for incremental learning is still the Very Fast Decision Tree (VFDT) [38], which has been improved numerous times over the years. An interesting alternative, which is able to learn faster (achieve better accuracy sooner) and converges to batched decision tree form, is the Extremely Fast Decision Tree (EFDT) [39]. Vertical Hoeffding Trees (VHT) are the first distributed streaming algorithm for decision trees and offer significantly improved computation speed in comparison to VFDT and EFDT [40]. A lot of effort has also been dedicated to incremental learning in the deep learning domain [41]. With network architectures that include long short-term memory (LSTM) modules, the problems of heterogeneous data fusion might be at least partially solved already within the learning method. Evaluation of incremental learning techniques is usually achieved with the prequential evaluation approach [42]. Our framework supports such evaluation of incremental learning methods.

Frameworks for stream processing. Several architectures and solutions have arisen from the wave of distributed processing engines originating at Hadoop. A couple of generations of Apache domain projects have arisen in the last decade like Apache Spark [43], Apache Samza [44], Apache Flink [45] and Apache Apex [46]. In addition, message distribution systems (like Apache Kafka) have evolved, providing infrastructure for fast stream processing. Some of the systems support the enrichment of data streams with aggregations, some even offer to merge data streams based on the premise, that the most recent data is available immediately in the stream. More complex data fusion strategies are up to the user. Our methodology does provide those missing strategies as well as it implements aggregation operators. All described Apache Software Foundation's top-level projects take into account distributed processing of data streams, which is not the focus of our research. As we will describe in the following sections, within the IoT, the distributed processing emerges naturally as most often we have to process data from many IoT devices, where each device offers a limited problem, that can be handled within one processing unit.

No efficient production targeted tool mentioned above implements state-of-the-art incremental learning methods. These implementations are still limited to academic community. The most well-known tool for stream mining are MOA (Massive Online Analysis) [6] and its clones in other languages (i.e., `streamDM-cpp` [47] and `scikit-multiflow` [48]). While these tools provide implementations of the state-of-the-art stream learning algorithms, they completely ignore the need for on-line data pre-processing and streaming data fusion.

QMiner [49] is a stream processing engine (SPE). It offers operators for aggregating data streams as well as operators for merging and resampling multiple streams. We have built our methodology on top of QMiner infrastructure and extended its functionality to support heterogeneous streaming data fusion.

Streaming data fusion platforms. A conceptual platform [50] for the usage of stream mining in the domain of big data is describing a lambda architecture [51] approach. While the authors list all the relevant technologies and mention methods for a summary of streaming data, the platform does not present any details on data fusion implementation.

Real-time probabilistic data fusion for the large-scale IoT applications [26] demonstrates the usage of multi-modal data streams (the IoT data, weather and social media data streams) for efficient prediction of traffic congestions. The method implements a two-level architecture, where the first level analytics derives events from data streams and the second level is essentially a probabilistic complex event processor. They generate the rules with efficient batch processing. They also expose the problem of using a common time scale for heterogeneous data sources but exclude the possibility of delayed measurements. A similar approach is described in [52], where the authors formalize the position of machine learning/analytics within the *hut* architecture, where it is used to support event processing by providing rules through batch analytics. Both approaches, however, perform this operation in the batch processing part. On the contrary, our approach includes machine learning methodologies in the streaming part of the architecture and introduces incremental learning approach, which can work without the support of the batch processing part.

Autonomous discovery of high-level knowledge from ubiquitous data streams [16] is one of the rare works that does not focus only on combining specific information with well-defined meaning, but rather tries to provide a general framework, agnostic to a specific problem. The authors use data aggregation over time to summarize detailed data streams (and their derivatives) and provide fixed feature vectors based on n uniformly sampled data streams. In addition, we provide a general framework that is able to ingest multiple heterogeneous (non-uniformly sampled) data streams and is able to provide user-defined feature vectors based on current as well as historical aggregates over the data and their derivatives.

Multiple streams data fusion is presented in [53]. The methodology exploits multiple sensors measuring the same property to predict anomalies and does not attack the issue of heterogeneous streams data fusion. The IoT streaming data integration (ISDI) paradigm is introduced in [8] and the proposed ISDI framework solves real-time data integration using the generic window-based algorithm. The work addresses the crucial timing alignment issue in the IoT setting. While both, ISDI and our framework, solve similar issues, our proposal includes a solution that works in a truly streaming manner (using a single-pass over data records), includes integration of historical values and provides a more direct interface for generation of stream aggregates.

Data fusion is one of the central research topics within the IoT, however, rare domain agnostic platforms for the fusion of the heterogeneous streaming data sources which support machine learning techniques have been presented in the scientific literature so far. Related contributions have, however, increased noticeably over the last couple of years.

3. Problem Definition

One of the exploitation scenarios for the vast IoT data is to take advantage of its predictive potential through machine learning methods. For example, based on historical data from a smart grid we can build a model that is capable of predicting energy consumption profiles for the next day, which will help better planning of the energy distribution and thus provide cheaper energy for the end user. In order to create the best possible predictions we need to be able to not only work with the current power consumption values, but also with historical data, different stream aggregates and derivatives and, what is even more important, we need to be able to expand the data streams with relevant contextual information, such as weather data, human behaviour data and weather forecasts. Moreover, an on-line algorithm for creating rich feature vectors for machine learning methodologies is needed in order to provide new feature vectors as soon as possible and to support incremental learning scenarios. Our framework builds such feature vectors and exposes them to machine learning methods. Incremental learning methods are well tailored to the needs of the IoT since the models are computationally cheaper and since they usually capture concept drift (change of statistical properties of the target variables), which often appears in the IoT scenarios.

The two dominant reasons why this kind of data fusion task is not trivial are the heterogeneity of the IoT data and its time incoherence.

According to [54], heterogeneity is an intrinsic property of big data. Many definitions of heterogeneous data can be found in the literature and there is no common agreement on the definition among various authors. Among the properties that illustrate the issue are: Multi-modality of the data (even considering a mixture of continuous and categorical features and structured and unstructured data) and the technical aspects (i.e., format of the data), rate of independence, concept drift and dynamics of change, and privacy. In this work we consider data coming from different sources and focus on heterogeneity based on the discrepancies in the time component [55,56], which is, in our opinion, the most important in the IoT data streams. It is manifested through the following properties: (i) sampling frequency, (ii) time delay, and (iii) data availability. Sampling frequency differs from sensor to sensor. Some sensors implement constant sampling frequency. Different sensors within a setup could implement constant, but different sampling frequencies. Many sensors implement approximately constant sampling frequency. he readings happen approximately in the prescribed interval, but due to different effects, the reading might be slightly early or late. Some sensors might use arbitrary sampling frequencies (i.e., they might only report an event). Time delay is introduced with transmission latencies, legacy systems and privacy/access issues. Measurements might be late from a few milliseconds up to one day (i.e., when data is transmitted from a legacy system via FTP connection). Delay is closely related to data availability.

A data stream is a sequence of values with a corresponding timestamp (in IoT a timestamp denotes the time when the measurement was taken). We define a coherent time series to be such a sequence, where each subsequent measurement in a series has been taken later than the previous. The most obvious data source which breaks time coherence is the weather forecast data stream. The forecasting models update their predictions regularly, usually every hour. With every update, older forecasts are updated with more accurate values, based on recent data.

Based on the issues arising from heterogeneity of the data we define a harmonic set of data streams. A harmonic set of data streams consists of a set of data streams, where each stream has a matching sampling frequency (and phase), at least one matching timestamp, and all the data that is needed for the successful generation of a feature vector.

The following subsections (Sections 3.1–3.7) present different types of data sources and their main characteristics as well as a thorough mathematical formulation of the basic concepts and approaches used in our methodology.

3.1. Types of Data Sources

Modeling in the IoT scenarios is based on the three different types of data sources. (1) Sensor data. Sensor data most often originates from IoT devices but can be obtained also by crawling a particular web resource. The data is usually obtained close to real-time. However, different lags can be introduced due to various reasons. The data fusion system should be able to handle these time-related inconsistencies and build correct feature vectors, based on the most recent available data. (2) Weather forecasts. Weather forecasts are available for the future (usually we need them up until the time of our prediction horizon). The forecasts represent an incoherent data stream (as the values are updating with time), which needs to be handled appropriately in the stream fusion system. (3) Static data. This is the data that by definition does not change in time. Values are known for the indefinite future. The data includes attributes like: day of the week, day in the year, hour of the day, holiday, day before the holiday, working hour, etc. For simplicity reasons, we handle static data as a stream.

Availability of the three different types of data differs as depicted in Figure 1. Depending on the delivery mechanisms, sensor data arrives to the stream processing components with different lags. Figure 1 also introduces the available data horizon which is the latest timestamp for which all sensor data streams are available. It represents the latest timestamp for which feature vector generation can be triggered. Feature vectors are built for calculating predictions at the prediction horizon. Static and weather forecast data are therefore usually considered for that timestamp within feature vectors.

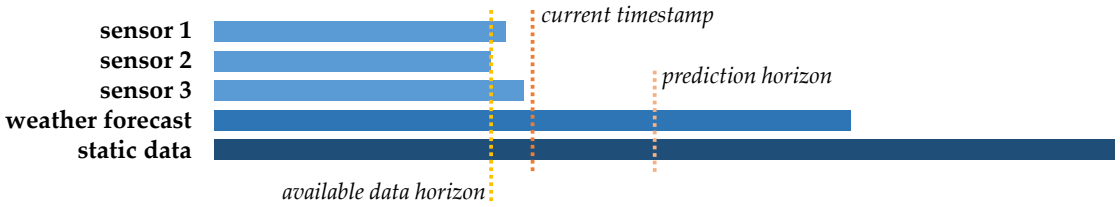


Figure 1. Data availability of different types of data sources. Sensor data is delivered in (almost) real-time. However, some legacy systems might introduce longer lags. Weather forecasts are available for a particular time in the future, while static data (i.e., date/time features and human behaviour data) are usually always available.

3.2. Data Streams

In most literature a data stream is represented by a sequence of values $x_1, x_2, \ldots, x_n$ where $x_i \in \mathbb{R}$ is an observation for all $i \in \{1, 2, \ldots, n\}$. We recognize that this notation has a drawback: it does not contain any information about when a particular value has been provided. Time is an important factor in deciding when to start a particular process on the data stream. To that end, we present a new definition of a data stream that includes time.

A **data stream** $\mathbb{O}_x^n$ is an ordered set of value-time observation pairs provided by a sensor or some other source of data and is described as

$$\mathbb{O}_x^n = \{(x_1, t_x^{(1)}), (x_2, t_x^{(2)}), \ldots, (x_n, t_x^{(n)})\},$$

where $t_x^{(k)}$ is the time of observation x_k and $t_x^{(i)} \leq t_x^{(j)}$ for all $i \leq j$. A **data stream window** $\mathbb{O}_x^{i,j}$ is a subset which contains observations between the i-th and j-th entries of a data stream $\mathbb{O}_x^n$ and is described as

$$\mathbb{O}_x^{i,j} = \{(x_i, t_x^{(i)}), (x_{i+1}, t_x^{(i+1)}), \ldots, (x_j, t_x^{(j)})\}.$$

In addition, we will write $\mathbb{O}_x^{1,n} = \mathbb{O}_x^n$.

Remark 1. *A data stream can also be defined as an ordered set of vector-time observation pairs, e.g., by replacing the values with vectors in the definition above. By doing so we would allow an observation to contain multiple values and thus generalize the data stream. For the sake of simplicity, we will use the value-time data stream definition in the rest of the document and will reference this remark when required.*

3.3. Static Data Stream

Data streams are dynamic in nature; they are created by retrieving signals from sensors which are then transformed and added to the stream. In some cases, we know what data will come at a particular timestamp. One such case is the day-of-the-week data stream where for a given timestamp we know which day of the week it corresponds to. This type of data streams is defined as static data streams $\mathbb{S}_w^n$ and is described as

$$\mathbb{S}_w^n = \{(w_1, t_w^{(1)}), \ldots, (w_n, t_w^{(n)})\}.$$

Returning to the day-of-the-week static data stream, it contains values $w_i \in [1, 2, \ldots, 7]$ where the number corresponds to a particular day of the week associated with its timestamp (1 corresponding to Monday, 2 to Tuesday etc.). Another example is the *weekend static data stream* which contains information weather a timestamp is in a weekend interval. Its values are $w_i = 1$ if the timestamp $t_w^{(i)}$ is inside a weekend interval and $w_i = 0$ otherwise. Similarly, a *holiday static data stream* is a static data stream which contains information if a timestamp falls in a holiday. Notice that some data streams depend on the context (e.g., culture, country).

3.4. Data Stream Aggregate

When we process data streams we might want to group observations together to form a new value that summarizes the data stream. To do this, we require a data stream **aggregate** function which is able to combine the data stream observations and returns the summarized (aggregated) value. A data stream aggregate can also be applied on a data stream window $\mathbb{O}_x^{i,j}$. Generally, a data stream aggregate function is defined as

$$\text{aggr}(\mathbb{O}_x^n) = X,$$

where X is the aggregated value of the provided observations. The most common data stream aggregate functions are (a comprehensive list is available in [57,58]):

- **Count.** Counts the number of observations in a data stream: $X = n$,
- **Maximum.** Returns the maximum value in a data stream: $X = \max\{w_1, \ldots, w_n\}$,
- **Minimum.** Returns the minimum value in a data stream: $X = \min\{w_1, \ldots, w_n\}$,
- **Sum.** Sums up all values in a data stream: $X = \sum_{i=1}^{n} w_i$.

A more complex example of a data stream aggregate is the **moving average** (MA). This aggregate is used to smooth out short-term fluctuations and highlight longer-term trends. It calculates the average of the observations within a data stream window $\mathbb{O}_x^{i,j}$ and is defined as

$$\text{MA}(\mathbb{O}_x^{i,j}) = \frac{1}{j-i+1} \sum_{k=i}^{j} x_k.$$

When the data stream window moves the new MA can be calculated by using the previous MA value:

$$\text{MA}\left(\mathbb{O}_x^{i+1,j+1}\right) = \frac{(j-i+1) \cdot \text{MA}(\mathbb{O}_x^{i,j}) - w_i + w_{j+1}}{j-i+1}$$

The second more complex aggregate function is the **exponential moving average** (EMA). It is similar to MA only that it incorporates a decaying factor; giving the more recent observations greater importance. This aggregate inputs the data stream $\mathbb{O}_x^n$ and is calculated with the following recursive function:

$$\text{EMA}(\mathbb{O}_x^n) = \begin{cases} x_n, & \text{for } n = 1, \\ \alpha(n) \cdot x_n + \big(1 - \alpha(n)\big) \cdot \text{EMA}(\mathbb{O}_x^{n-1}), & \text{for } n \neq 1, \end{cases}$$

where $\alpha(n) = \frac{\Delta t(n)}{T}$, $\Delta t(n) = t_x^{(n)} - t_x^{(n-1)}$ represents the rate of the decay and T is the user defined split time constant.

Once we decide on the aggregate functions to use in processing, we can create an aggregated data stream. An **aggregated data stream** $\mathbb{O}_{x,\text{aggr}}^n$ is a data stream containing the sequence of aggregated values of $\mathbb{O}_x^n$ by using the aggregate function "aggr".

$$\mathbb{O}_{x,\text{aggr}}^n = \left\{ \big(\text{aggr}(\mathbb{O}_x^1), t_x^{(1)}\big), \ldots, \big(\text{aggr}(\mathbb{O}_x^n), t_x^{(n)}\big) \right\}.$$

We will now look at two aggregated data stream examples:

Moving average data stream. This aggregated data stream is created by using the MA aggregate and is described as

$$\mathbb{O}_{x,\text{MA}}^n = \left\{ \big(\text{MA}(\mathbb{O}_x^{1,k+1}), t_x^{(1)}\big), \ldots, \big(\text{MA}(\mathbb{O}_x^{n-k,n}), t_x^{(n)}\big) \right\},$$

where $k < n$ is the user defined data stream window size.

Exponential moving average data stream. This aggregated data stream is created by using the EMA aggregate and is described as

$$\mathbb{O}^n_{x,\text{EMA}} = \left\{ \left(\text{EMA}(\mathbb{O}^1_x), t^{(1)}_x\right), \ldots, \left(\text{EMA}(\mathbb{O}^n_x), t^{(n)}_x\right) \right\}.$$

More complex stream aggregates can require interpolation over the time-series and calculation of values such as: number of extremes in a particular data stream window, highest n-th derivative in the data stream window, duration of the largest maximum, etc. Such derivatives are, for example, useful for modeling of crop types in earth observation scenarios.

3.5. Data Stream Resampler

One of the properties of data streams is that observations might not come at a constant rate. This can cause problems when multiple data streams need to be synchronized. To handle this issue we define a **sampling function** which takes a data stream $\mathbb{O}^n_x$ and a timestamp T as an input and returns a sample value. The function can be described as

$$\text{sampler}(\mathbb{O}^n_x, T) = X_i,$$

where X_i is the sampled value generated from the input parameters. The sample value can be generated using different functions:

- **Last value.** This function returns the last observation that appeared before the provided time: x_k, where $t^{(i)}_x < T$ for all $i \leq k$ and $T \leq t^{(j)}_x$ for $k < j$.
- **First value.** This function returns the first observation that appears after the provided time: x_k, where $t^{(i)}_x < T$ for all $i < k$ and $T \leq t^{(j)}_x$ for $k \leq j$.
- **Linear interpolation.** This function returns the sample value by using a linear interpolation between observations around the provided time, e.g.,

$$\text{lin}(\mathbb{O}^n_x, T) = \frac{x_k - x_{k-1}}{t^{(k)}_x - t^{(k-1)}_x}\left(T - t^{(k)}_x\right) + x_k,$$

  where $t^{(k-1)}_x \leq T \leq t^{(k)}_x$.

Once we decide on the sampling function f and a constant time period T we can create a data stream of sampled data. This type of data streams is defined as a **resampled data stream** containing the sampled values of a provided data stream $\mathbb{O}^n_x$ and is described as

$$\mathbb{O}^m_{x,f} = \{(X_1, T^{(1)}_X), \ldots, (X_m, T^{(m)}_X)\},$$

where $X_i = f\left(\mathbb{O}^n_x, T^{(i)}_X\right)$ is the sampled value returned by the sampling function f performed at time $T^{(i)}_X$. In addition, the difference of consecutive time values is equal to the constant time period T, i.e.,

$$T = T^{(i+1)}_X - T^{(i)}_X,$$

for all $i \in \{1, \ldots, m-1\}$.

3.6. Data Stream Merger

Sometimes we require to merge two or multiple data streams into a single data stream. We define a merger function that is able to do just that. Suppose we have two data streams $\mathbb{O}^n_x$ and $\mathbb{O}^m_y$ where $t^{(i)}_x = t^{(i)}_y$ for all $i \leq \min\{n, m\}$. A **merger function** takes $\mathbb{O}^n_x$ and $\mathbb{O}^m_y$ as an input and returns the merged data stream, i.e.,

$$\text{merger}(\mathbb{O}_x^n, \mathbb{O}_y^m) = \{(x_1, y_1, t^{(1)}), \ldots, (x_k, y_k, t^{(k)})\},$$

where $k = \min\{n, m\}$ and $t^{(i)} = t_x^{(i)} = t_y^{(i)}$ for all $i \in \{1, \ldots, k\}$. If we adopt the view provided in Remark 1, the output of the merger function is a data stream. Indeed, if we write the values x_i, y_i as entries of a vector then the output will follow the generalized definition of a data stream.

3.7. Forecast Data Stream

Forecast data streams provide forecasted values for the future. The difference between the time of the forecast and the time of the forecast generation is called a prediction horizon. A simple forecasting model provides forecasts for a constant prediction horizon and can be described simply by $\mathbb{O}_p^n$, where p_i is the forecasted value and $t_p^{(i)}$ refers to a timestamp in the future. More complex forecasting streams (i.e., weather forecasts) provide predictions for multiple time horizons at the same time. For example, every hour new weather forecasts are being generated for the next 48-hour interval. This implies that a forecast for a particular hour in a day gets updated 48-times during this process. At forecast generation a new data stream window $\mathbb{O}_p^{i,j}$ is generated, thus the forecast data stream is defined as

$$\mathbb{F}_x^n = \{(\mathbb{O}_p^{1,k}, t_x^{(1)}), (\mathbb{O}_p^{2,k+1}, t_x^{(2)}), \ldots, (\mathbb{O}_p^{n,k+n-1}, t_x^{(n)})\}, \tag{1}$$

where k is the number of time horizons and $\mathbb{O}_p^{i,k+i-1}$ is the data stream window generated at time $t_x^{(i)}$. In the case of $k = 1$, the forecast data stream contains a simple forecasting model, which generates only one prediction of a constant prediction horizon.

3.8. Feature Vector

In machine learning, a *feature vector* is a vector that contains data important for description and modeling of a particular system. Together with a label of a particular data instance, it is used for training of a machine learning model. If the label is unknown, the vector can be used to derive predictions (or other results) from a trained model. An element in the vector is called a feature (in some literature it is referred to as an attribute). A feature vector which includes contextually rich information, derived from a set of data streams, will allow a machine learning model to achieve the best possible results in modeling particular phenomena. Such a vector ϕ_{full} should be the final result of a streaming data fusion framework.

A feature vector ϕ_{full} is a representation of a set of observation, static data and forecast data streams $\mathbb{O}_x^n$, $\mathbb{S}_y^n$ and $\mathbb{F}_z^n$, where $x \in \{x_1, \ldots, x_i\}$, $y \in \{y_1, \ldots, y_j\}$ and $z \in \{z_1, \ldots, z_k\}$. We will focus on feature vectors of the following form:

$$\phi_{\text{full}} = \begin{bmatrix} F^{(1)} \\ F^{(2)} \\ \vdots \\ F^{(q)} \end{bmatrix},$$

where q is the size of the feature vector and each feature $F^{(r)}$ is defined with:

$$F^{(r)} = \begin{cases} x_i^{(j)}, & \text{a (resampled) measurement extracted from } \mathbb{O}_{x_i}^n \\ y_i^{(j)}, & \text{a (resampled) forecast extracted from } \mathbb{F}_{y_i}^n \\ z_i^{(j)}, & \text{a (resampled) forecast extracted from } \mathbb{S}_{z_i}^n \\ X_i^{(j)}, & \text{a (resampled) stream aggregate extracted from } \mathbb{O}_{x_i}^n \\ Y_i^{(j)}, & \text{a (resampled) stream aggregate extracted from } \mathbb{F}_{y_i}^n \\ Z_i^{(j)}, & \text{a (resampled) stream aggregate extracted from } \mathbb{S}_{z_i}^n \end{cases}$$

Note that the index $j \in \{1, \ldots, n\}$ can refer to current or historical values of the measurements or the aggregates. Additionally, each feature vector reflects the state of the observed system at resampled timestamp $T_x^{(n)}$.

4. Architecture and Methods

4.1. Architecture

The architecture of the proposed framework (http://github.com/klemenkenda/iot-fusion/) is depicted in Figure 2. The framework is designed to be easily integrated into a speed layer of a standard big data processing lambda architecture [51]. The proposed framework consists of three main building blocks: pre-processing, fusion and modeling. Pre-processing includes data adapters for various data streams. Data streams are enriched with stream aggregates, resampled to a common timestamp (see Algorithm 1) and partial feature vectors are extracted in the partial fusion component (see Algorithm 2). The data fusion block accepts partial feature vectors from a set of data sources and merges them together into a full feature vector (see Algorithm 3). In this component, potentially different timestamps are compensated and additional derivatives are calculated from partial feature vectors (i.e., the difference between current and yesterday's daily average electricity consumption).

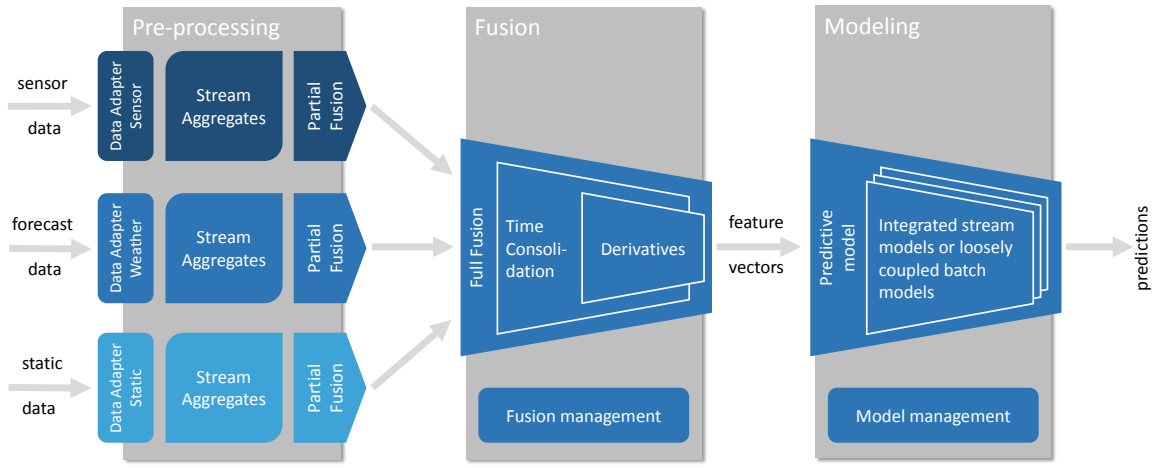


Figure 2. Data fusion framework architecture with added modeling component. The framework consists of three main components: pre-processing, fusion and modeling. Pre-processing is dedicated to the independent transformation of particular data streams, fusion merges them together into full feature vectors, whereas modeling provides predictions from either generated batch models or from incremental learning models.

The modeling component is also included in the framework since data-driven modeling methods perform a kind of data fusion by mapping feature vectors into predictions. Two different modeling components have been implemented: (a) based on a rich ecosystem of batch learning techniques and (b) based on a sparse implementations of stream learning techniques. The latter component is useful in the edge scenarios, where computational efficiency is paramount.

The whole process is controlled via a single configuration structure, which defines the input data sources, data enrichment (a set of stream aggregates attached to a particular data stream), resampling time, the particular elements of a full feature vector and even meta-data relevant for predictive analytics (prediction horizon, selected modeling method and corresponding parameters).

4.2. Complex Forecast Transformation into a Coherent Data Stream

Forecasts usually represent important contextual information regarding the process we are trying to model and can drastically improve the accuracy of our models. We have defined the forecast data

stream $\mathbb{F}_x^n$ in Equation (1). An attentive reader might observe that such a data stream is breaking the basic rules of a coherent data stream [36]. A forecasted value p_j at time $t_p^{(j)}$ gets updated with each new received data stream window $\mathbb{O}_p^{i,k+i-1}$, as long as $j \in \{i, i+1, \ldots, i+k-1\}$. For such streams it is impossible to use the majority of the stream mining algorithms, including stream aggregators.

We propose decomposing the complex stream from Equation (1) into a set of k streams, where each stream represents forecast for a constant prediction horizon h and can, therefore, be simply denoted as $\mathbb{O}_{p_h}^n$. For example, a weather forecast stream which contains 48-hourly forecasts in each update would decompose into 48 separate coherent data streams. With this transformation complex forecast streams can be simply integrated into our framework.

4.3. Data Flow

Data flows within our system reflect the proposed architecture and the particularities of different stream types. On the farther left side of Figure 3 is the forecast complex stream $\mathbb{F}_x^n$. As this stream breaks the definition of a coherent time series it is conformed to a set of simple streams within the data adapter as described in the previous subsection. The second column of Figure 3 represent the initial phase of data fusion. The depicted data streams share the same form but do not have a common time denominator. Afterwards, all data streams are enriched with stream aggregates. Common time denominator is established in streams $\mathbb{EO}_x^n$ with resampling the data streams to to the master time interval T. Data fusion is performed in two steps. Firstly, the enriched data streams are transformed into partial feature vector streams $\mathbb{PO}_x^n$ by adding additional historical values or derivatives to the data stream. Finally, these feature vector streams are merged into final feature vector data stream $\mathbb{FO}_x^n$.

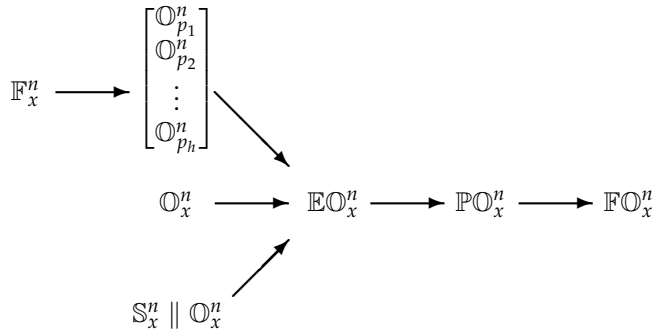


Figure 3. Hierarchy of data streams in the stream fusion framework. Heterogeneous data streams are consolidated and merged with every step of the stream fusion (depicted from left to right). Raw data streams ($\mathbb{F}_x^n$, $\mathbb{O}_x^n$ and $\mathbb{S}_x^n$) are transformed, enriched and resampled into coherent data streams $\mathbb{EO}_x^n$ and then fused through partial feature vector streams $\mathbb{PO}_x^n$ into a final full feature vector data stream $\mathbb{FO}_x^n$.

4.4. Enrichment of a Data Stream

Enrichment of a data stream is a process which takes a data stream $\mathbb{O}_x^n$ as an input and adds a set of additional stream aggregates $\mathbb{O}_{x,\text{aggr}}^n$ to it. Our enrichment procedure is described in Algorithm 1. Alongside the data stream $\mathbb{O}_x^n$ the algorithm also requires information about the configuration of the stream aggregates (`config`). Based on the `config` the algorithm initiates a set of stream aggregates and attaches them to the data stream, which enables stream aggregate operators to update their values with every new element in stream $\mathbb{O}_x^n$. Additionally, we initiate a resampler operator in this step. The task of the resampler operator is to put the data stream $\mathbb{O}_x^n$ to a common time $T_x^{(i)}$, which is shared among all the data streams in the stream fusion process. Enrichment algorithm listens to the data stream and triggers an action after every measurement is received. Firstly, all the attached stream aggregates are updated with the new value. Next, an enriched vector b_n consisting of the original measurement, all of the stream aggregate values and the measurement timestamp is created. Finally, the enriched

vector b_n is inserted into the resampler and if a new resampled vector b'_n is available it is pushed into the resulting enriched resampled data stream $\mathbb{EO}_x^n$. Note: the data stream $\mathbb{O}_x^n$ only utilizes last values, which means that historical values are not stored by the algorithm. Historical values needed for the calculation of window based data stream aggregates are stored in buffers within the aggregate mechanism [49]. The data stream $\mathbb{EO}_x^n$ is stored as a buffer $\mathbb{EO}_x^{i,j}$, where the window's right limit j is increasing with each new measurement, and left limit i is increased after the cleanup of the obsolete data in Algorithm 3. After the initial phase, when old measurements are being collected in order to satisfy the needs of a feature vector generation, the size of the $\mathbb{EO}_x^{i,j}$ buffer remains constant.

Algorithm 1: Enrichment of a data stream with stream aggregates.

Data: Data stream $\mathbb{O}_x^n$; configuration of stream aggregates `config`
Result: Enriched resampled data stream $\mathbb{EO}_x^n$ based on $\mathbb{O}_x^n$
init a set of j stream aggregators $X^{(j)}$ from `config`$[\mathbb{O}_x^n]$;
init resample (= tick) interval T from `config`;
start listening to stream $\mathbb{O}_x^n$;
while *stream is active* **do**
  wait for next instance in $(\mathbb{O}_x^n)$;
  update stream aggregates with observation $(x_n, t_x^{(n)})$;
  create enriched stream record $b_n \longleftarrow (x_n, X^{(1)}, \ldots, X^{(j)}, t_x^{(n)})$;
  if *stream will yield next resampled value* $b'_n = \text{sampler}(\mathbb{O}_x^n, T)$ **then**
    push b'_n to stream $\mathbb{EO}_x^n$;
  end
end

The new stream $\mathbb{EO}_x^n$ satisfies all the properties of a data stream from Section 3 and can be further used in any stream mining algorithms within the pipeline. The new stream contains new enriched data as well as conformed timestamp.

4.5. Data Fusion Algorithms

Streaming data fusion consists of two separate algorithms. The partial fusion algorithm (see Algorithm 2) and full fusion algorithm (see Algorithm 3). The partial fusion algorithm transforms an enriched data stream $\mathbb{EO}_x^n$ into a partial feature vector based on the values from this data stream. The full fusion algorithm merges all partial feature vectors together and forms the final full feature vector, which is ready to be used for learning or prediction in any machine/stream learning method.

The purpose of the partial fusion algorithm is to add additional historical and derived features (out of a single data stream) by using a data stream window $\mathbb{EO}_x^{i,j}$ of recent values. Additionally, the algorithm takes current feature generation time T_c and configuration of features ($\mathbb{A}$) as input data. Each feature is identified by an `elementPosition` (column) in an item of $\mathbb{EO}_x^{i,j}$ as well as by its `relativeOffset` from feature generation time in the stream (row). For example, a relevant feature for most of the models, which predict human behaviour (i.e., energy consumption), is the value of the phenomena we are trying to predict from a day before. For instance, if our resampling time T is equal to 1 h, the relative offset would be -24. Partial fusion is an algorithm that is called by Algorithm 3. The algorithm initiates an empty partial feature vector p. Then it transverses all the features from a set of partial feature vector features $\mathbb{A}$ and inserts the appropriate values (according to position and offset) into the partial feature vector. Algorithm 2 can easily be extended with additional feature generators (i.e., time differences or averages over a set of features), which we have demonstrated in the real-world use cases (see Section 6). If the data in the current enriched resampled stream window $\mathbb{EO}_x^{i,j}$ is inadequate (i.e., some historical data is missing), the algorithm throws an exception. Final partial feature vector $p \in \mathbb{PO}_x^n$ is returned to the full fusion algorithm.

Algorithm 2: Partial fusion algorithm in pre-processing step.

Data: Enriched data stream window $\mathbb{EO}_x^{i,j}$; current timestamp for feature generation T_c; list of features $\mathbb{A}$
Result: Partial feature vector stream $p \in \mathbb{PO}_x^n$
init empty feature vector p;
set feature vector timestamp $\longleftarrow T_c$;
for $f \in \mathbb{A}$ **do**
  $e \longleftarrow f$.`elementPosition`;
  init `currentOffset` in $\mathbb{EO}_x^n$ that corresponds to T_c;
  `relativeOffset` $\longleftarrow f$.`offset`;
  $k \longleftarrow$ `currentOffset` + `relativeOffset`;
  if $k \notin [i, j]$ **then**
    throw error feature vector can not be generated;
  end
  push $x_k^{(e)} \in \mathbb{EO}_x^{i,j}$ to p;
end
push p to data stream $\mathbb{PO}_x^n$;

Algorithm 3: Full data fusion algorithm.

Data: a set of data streams $\mathbb{A}$ relevant for full feature vector generation; master sensor $m \in \mathbb{A}$, which dictates the generation of the feature vectors
Result: Feature vector data stream $\mathbb{FO}_x^n$ for master node m
start listening to data streams $\mathbb{EO}_A^n$, $A \in \mathbb{A}$;
$T_{lc} \longleftarrow 0$; `// init last time, when feature vector was generated`
while *all streams are active* **do**
  trigger the new fusion after receiving next instance from stream $\mathbb{EO}_A^n$;
  if *not all streams are available* **then continue**;
  set T_c to the smallest $T_m > T_{lc}$ in master node enriched resampled data stream $\mathbb{EO}_m^n$;
  if *such T_c exists* **then**
    if *partial feature vector generation is not possible for all streams from* $\mathbb{A}$ **then continue**;
    init empty feature vector ϕ_m with timestamp T_c;
    for $s \in \mathbb{A}$ **do**
      append partial feature vector $p \in \mathbb{PO}_s^n$ from sensor s to ϕ_m; `// see Algorithm 2`
      cleanup obsolete data from stream s;
    end
    push ϕ_m to full feature vector data stream $\mathbb{FO}_m^n$;
    $T_{lc} \longleftarrow T_c$;
  end
end

The full data fusion algorithm collects all the partial feature vectors from set $\mathbb{A}$ and merges them together in a single full feature vector, which is suitable for usage in various stream/machine learning algorithms. The system requires initialization from a configuration of streams $\mathbb{A}$, relevant for feature vector generation, and information on the master sensor $m \in \mathbb{A}$, which dictates the generation of feature vectors. The result of the algorithm is a full feature vector $\phi_m \in \mathbb{FO}_m^n$. The algorithm initiates with the smallest possible time $T_{lc} = 0$, which indicates the last time of a feature vector generation. Upon each new record from any of the data streams $\mathbb{EO}_A^n$ the system checks if there is data already available from all streams $A \in \mathbb{A}$. The system tries to identify the smallest possible next feature

generation time from master sensor m. This is the smallest timestamp for which a full feature vector has not yet been successfully generated. If such a timestamp T_c exists, then the algorithm checks whether there is enough data (historical and for the future) within the $\mathbb{EO}_A^n$ buffers for successful generation of partial feature vectors. Then the algorithm instantiates an empty feature vector p for timestamp T_c and fills it with partial feature vectors from all the relevant sensors. In order to keep the memory usage as low as possible, the algorithm checks for any obsolete data records in buffers $\mathbb{EO}_A^{i,j}$ and removes them. Finally, the final full feature vector ϕ_m for the master node m is pushed to the appropriate stream $\mathbb{FO}_m^n$, where it is made available for any interested consumers, i.e., stream or batch machine learning algorithms.

The methodology implements a fully streaming data fusion algorithm. The requirement for historical data is satisfied with the usage of smallest possible buffers (i.e., internal buffers of stream aggregates based on sliding window or resampled enriched data stream window $\mathbb{EO}_x^{i,j}$). The methodology is generic and can be initiated for any stream modeling scenario with a configuration structure, which includes (a) a set of fusion meta-data like fusion id, tick time interval T and others, (b) a set of stream aggregates for a particular sensor s, where each stream can implement a set of *tick-based* (last value only) or *sliding window-based* aggregates, and (c) the definition of a feature vector, which consists of partial feature vectors for a particular sensor. Each partial feature vector can include an arbitrary set of features, which include current or historical sensor values, current and historical aggregated values over different sliding windows and even derivatives of current and historical values.

5. Integration

Our methodology is suitable to be included in lambda and similar big data architectures [51,52]. These architectures define the role of 2-fold data processing layers: batch and speed. Speed layer is dedicated to the processing of data streams. Quite often the speed layer is reduced to event processing [52], however, we propose to use incremental learning techniques independently in the speed layer (at least for edge processing applications) or to transfer the models from batch layer (where they are learned) to the speed layer (where they are used to provide predictions on an almost real-time data stream).

In the following subsections, we present two different integration scenarios: (1) integration in the cloud infrastructure and (2) integration in the edge/fog infrastructure. Additionally, due to the relatively low computational cost of streaming data fusion and modeling components, the system would be ideal for deployment in dew architecture (low-end servers, deployed near IoT devices), as proposed in [59].

5.1. Integration: Cloud Infrastructure

Integration in the cloud represents the usual implementation of our framework. We have used this setup in smart grid power, power station demand, groundwater level, public buildings power demand and other real-world scenarios. As depicted in Figure 4, such an integration consists of 4 different layers: Analytical Layer, Communication Layer, Data Layer and External Layer. The central component of the system is the message queue system within the Communication Layer. In our implementations, we have used Apache Kafka. Exposed through a single entry point Apache Kafka can distribute extreme amounts of data via elastic, scalable and fault-tolerant infrastructure. As each of the processing components (Preprocessing, Fusion and Modeling) only needs to access the message queue, they can be distributed over the computational infrastructure and thus ensure scalability even in the processing end.

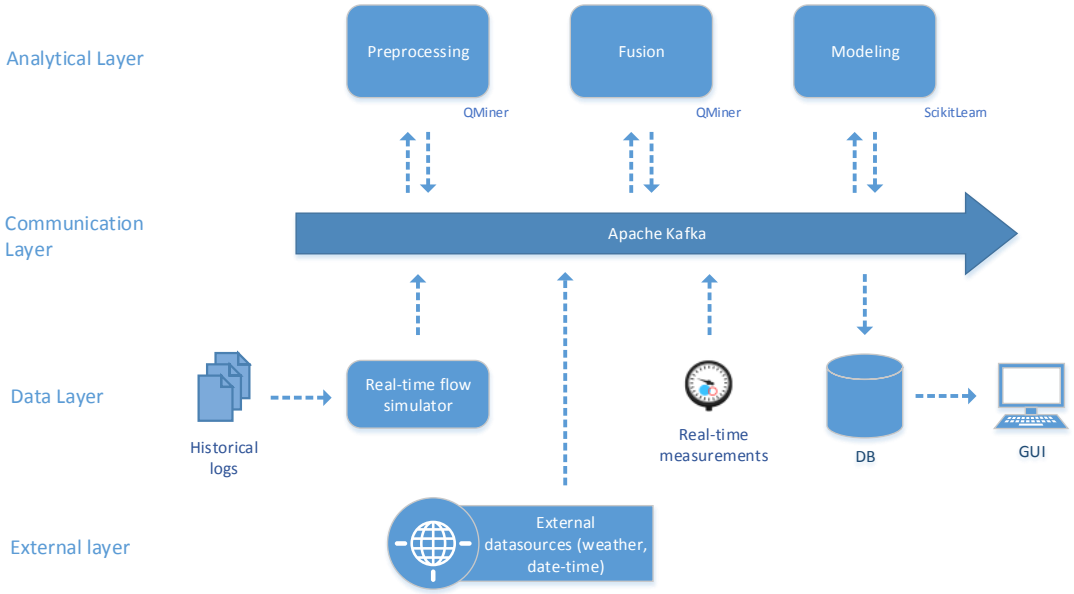


Figure 4. Integration in the cloud infrastructure is based on the message queue in the Communication Layer (in our applications this was Apache Kafka). Each component in the architecture is loosely coupled to the system and receives/sends data to the system via the message queue using a predefined data format. The system can receive simulated or real-time data (from sensors and from external data sources). Results are stored in the monitoring database (DB) and finally shown to the user (GUI).

Data is provided in the Data and External Layers. The External Layer includes external data sources, such as weather data, weather predictions and static data (data-time features, human behaviour data, etc.) while the Data Layer includes the essential IoT data infrastructure. Before real-world systems are deployed to the production, extensive testing is needed. This is ensured by real-time flow simulator component, which is able to simulate conditions in the real-time (or faster) based on historical log data files. The platform provides three types of results: (i) pre-processed feature vectors from the pre-processing component, (ii) final feature vectors from the fusion component and (iii) predictions from the modeling component. All the intermediate, as well as final results, are stored in the monitoring database (DB) for further analysis and visualization in the Graphical User Interface (GUI).

We have implemented our stream fusion system in QMiner [49] stream processing engine, which enables fast prototyping as well as production grade framework deployment and already a rich ecosystem of implemented stream aggregate operators.

5.2. Integration: Edge/Fog Infrastructure

Stream mining systems are suitable for the implementation in the edge/fog infrastructure due to their low computational demand. We have used this setup in energy demand prediction on a public train. As depicted in Figure 5, the integration is simpler than in the cloud scenario. In this integration, the loosely coupled architecture provides only additional overhead since the implementation is to be achieved within a single node. A messaging queue system can be omitted in this scenario. Data adapters can connect directly to the data sources (i.e., via HTTP API). Data is transferred between pre-processing, fusion and modeling components directly—via internal interfaces. All the components, including modeling, are implemented using QMiner framework. Incremental learning algorithms, like recursive linear regression and VFDT, are used. Predictions are exposed via lightweight WebSocket protocol and made available in the GUI. In another setup, we have used a lightweight message queue solution based on the MQTT protocol.

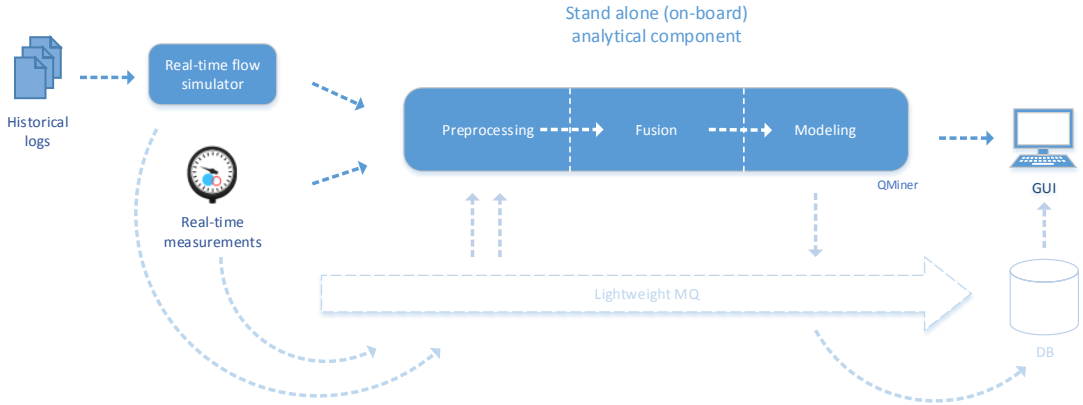


Figure 5. Data fusion framework integration in the edge/fog scenario. Message queue system can be completely omitted in this scenario and all communication is achieved via HTTP API or WebSockets (on the GUI part). Components are tightly coupled to ensure faster data transfer between components. Modeling is included in the analytical component and implements lightweight incremental learning algorithms.

6. Experimental Results

In this section we present experimental results on a selection of three different use cases from smart cities domains, demonstrating integrations in cloud and edge scenarios. For each use case, we have obtained a real-world dataset, i.e., sensor measurements from the actual testbed. We used non-parametric, linear models (e.g., ridge regression), as well as nonlinear models (e.g., k-nearest neighbours, decision trees, gradient boosting regression and random forests) of different complexities. In the attempt to reach the best possible model performance, we added additional data sources to the model and enriched it with various autoregressive derivatives. These additional data sources are: seasonable date-time related variables (e.g., hour of the day, day of the week), meteorological variables (current and forecasted weather), and additional static variables related to the use case (such as holidays status). To incorporate also different short term trends from the sensor values, we added various autoregressive features, computed for different rolling/sliding windows sizes. Relevant aggregate functions are *mean*, *minimum*, *maximum*, *sum*, and *variance*. Time windows that we used are: *1 h*, *6 h*, *1 day*, *1 week* and *1 month*. Due to the cyclic behaviour, we know that the features from yesterday, or from the same time in the previous week, can be similar to the current values, and are therefore useful for the model. Therefore, some past feature values (*1 day*, *2 days*, and *1 week back*) were also included as autoregressive features.

With respect to the data sources, there are the following universal denominations that we use in the text (for example, data set with name `M_AR_WC_WF` means that measurements with autoregressive derivatives, current weather and forecasted weather features are included in the dataset):

- **M** — Available sensor measurements (cleaned and resampled)
- **AR**—Autoregressive variables (measurements and their historical and aggregated values)
- **WC** — Current weather
- **WF** — Weather forecasts
- **DT** — Date-time (calendar) properties
- **TOP_20** — 20 most important features (obtained with feature selection process)

Evaluation of data fusion algorithms is a topic that has not yet been addressed well even in the traditional batch scenarios. The issue remains an open challenge [1,32]. The data fusion algorithms are usually tested in a simulation environment with unclear performance benefits in a real-world setting. It has been shown that only 1 in almost 20 research papers focuses on the evaluation, relevant

to practical applications [60]. To the best of our knowledge, no standardized methodology exists for comparing stream fusion techniques and this remains an open research challenge. Potential evaluation methodology should assess the expressiveness of the feature vector description language (i.e., number of available stream aggregates, ability to utilize historical values, derivatives and aggregates, ability to generate new features by transforming existing ones, ability to include different types of data sources, ability to perform feature selection on-line etc.) as well as how does it benefit the real world applications (i.e., by improving the models, ease of use, number of hyper parameters to be defined, ease of connectivity, initialization time, robustness etc.).

In our evaluation, we show that our methodology has helped to improve modeling capabilities, which is an indirect measure of its benefits in the real world real-time systems. Our system includes six different stream aggregate operators, it can use different historical values and aggregates, it is able to generate new features by applying the difference between historical values and it can integrate three different types of data streams. It can be implemented by a single JSON config file of data sources, feature vectors and model parameters, and it supports connectivity via Apache Kafka, MQTT or REST API. Initialization of the system is dependent on historical data needed for feature vector construction. The system can not estimate historical data based on available data, which in the case of a requirement to include a week old value, will not produce a feature vector until 1 week of viable measurements are in the system.

In Section 6.3 we compare our framework against the ISDI framework [8]. Other methodologies that use values and derivatives (i.e., Kalman filter based methods or other machine learning approaches) can (in the best case) achieve the performance of autoregressive features (denoted with `M_AR` in the subsections below).

6.1. Cloud Infrastructure Deployment for Smart Grid

In this section, we present results from two separate use cases related to electricity distribution. The first case shows the application of our methodology in the smart grid, the second case shows the modeling results from power stations supporting public trains.

The **smart grid** use case includes results on predicting measured power from smart meters at five industrial consumer sites. Mean load at each smart meter was 10kW. Testing data set included two full years of data with hourly resolution. We have tested the short-term load forecast scenario (as defined by the energy domain) with a prediction horizon of 10 h. Results are depicted in Figure 6.

The learning curves compare the performance of a model on training and test data over a varying number of training instances. In Figure 6, the red learning curves represent the training score and the great learning curves represent the test score in terms of R^2. Training score is calculated on a training data set and test score (or cross-validation score) is calculated on a testing set by using cross-validation. In the experiments, we use cross-validation with 10 iterations to get smoother mean test and train score curves, each time with 20% data randomly selected as a testing set (i.e., 80:20 train-test ratio). The shaded area around each curve represents the standard deviation from mean test scores of each step in the cross-validation.

Learning curves offer a better overview of the trained models and allow a data analyst to diagnose, whether the model is trained well or if it has some weakness. The latter can be improved by optimizing method parameters, including more data or reducing the number of features. Such adjustments can prevent overfitting or help models achieve optimal accuracy with the given data set. A tight fit between training and cross-validation scores can indicate that the model suffers from high bias (is under-fitted). Horizontal curve and a consistent gap between the scores indicate that a model has learned as much as it can about the data (additional data would not help). We can see that this is the case for ridge regression and gradient boosting regression. In such cases, one of the standard ways to improve the performance of a model that is suffering from a high bias, is by adding additional informative features or by optimising model parameters. Indeed we can observe that the score has increased using more features (comparison between feature sets from first to the fourth column in Figure 6). A wide

gap between training and cross-validation test scores usually indicates that the model is dealing with high variance (over-fitting) problem. This is clear for decision trees, which completely reflect the training data (an almost perfect score is depicted by the red curve), but do not generalize well. In such cases, we might improve our model by obtaining more training examples, or by decreasing model complexity (by decreasing number of features, or model parameter optimisation—i.e., by using shallower trees). Ideally, we want to find a sweet spot that minimizes bias and variance, by finding the right number of features and the right level of model complexity. Among all the possibilities we chose random forests model. Its test scores are the highest (for both, training and cross-validation) and they converged to a constant training score, with more or less all data sets. A tight gap between the testing and training set also indicates that the model generalizes well with new data. Regarding the features, we can observe that for most of the algorithms adding auto-regressive features of the input measurements increased the model performance the most. Static date-time features additionally increased the performance for most algorithms, while weather forecast features don't seem to affect the modeling results. Nevertheless, we have to take into account that these scores are averaged over the entire testing set. Cases where special features such as bad weather or holidays help are rare. The improvement of the test scores is modest in this case, but correct prediction in these rare cases is valuable as it exposes a deviation from the normal behaviour.

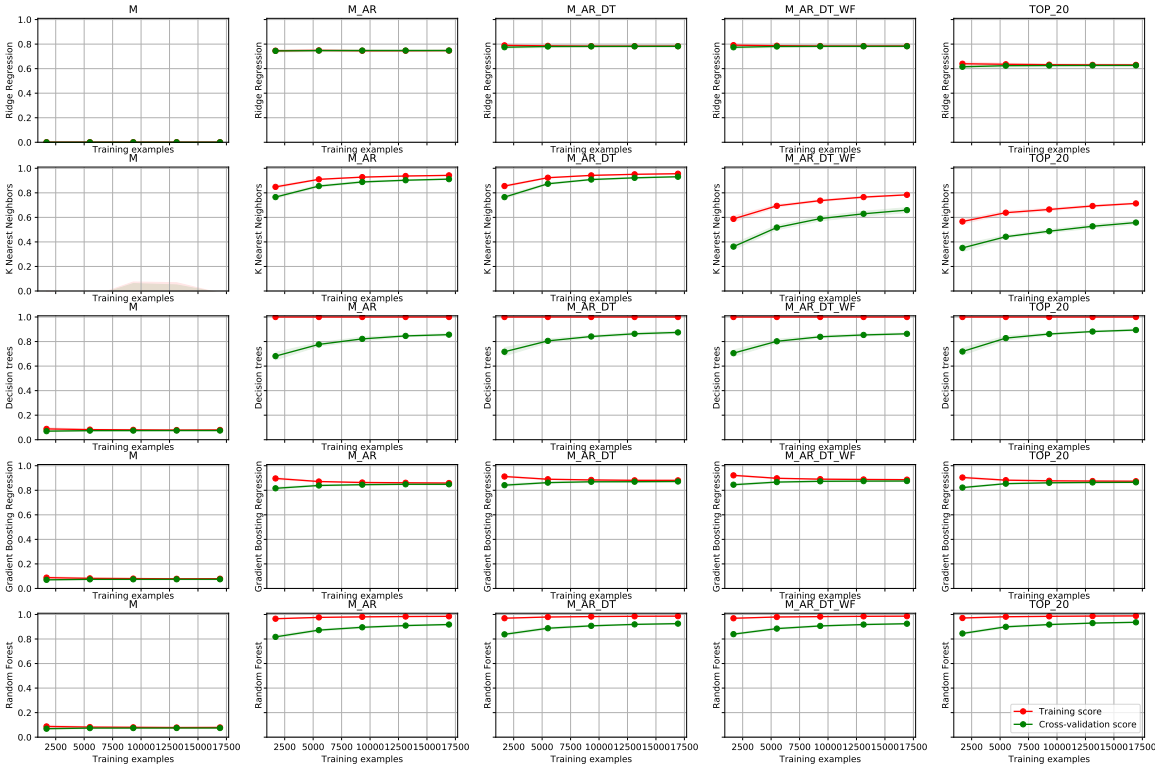


Figure 6. Learning curves on the smart grid use case. Columns depict five different feature vector definitions: `M`—measurements, `M_AR`—measurements and autoregressive features, `M_AR_DT`—measurements, autoregressive features and data-time features, `M_AR_DT_WF`—all of the features from `M_AR_DT` and weather forecasts, `TOP_20`—best 20 features. The rows include results from different learning algorithms: ridge regression, k-nearest neighbours, decision trees, gradient boosting and random forest, respectively. Each sub-figure in the matrix presents a number of training examples on x axis and R^2 score in the y axis. The green line represents the learning curve on the test data. The red line represents the learning curve on the train data. The darker band around the curves depicts standard deviations of the R^2 score.

The second use case presents experiments with train substation feeder with a mean load of approximately 50kW. The data set includes 2 months of measurements with hourly resolution. Again, the prediction horizon has been set at 10 hours. Results are depicted in Figure 7.

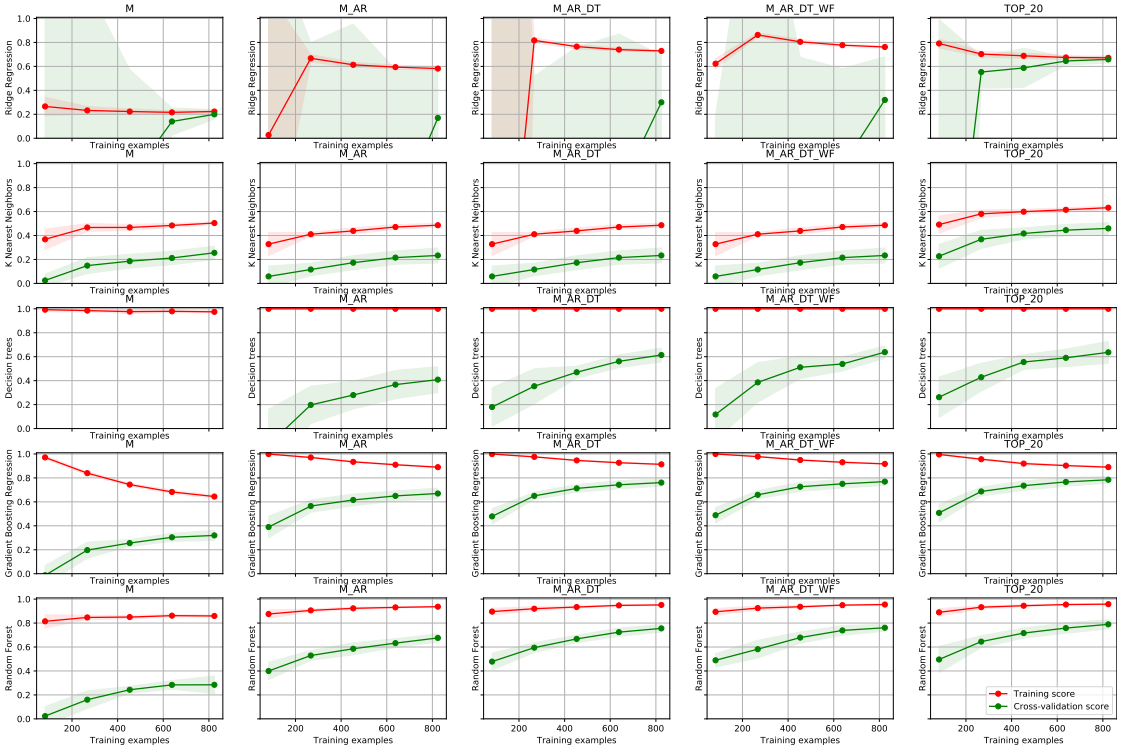


Figure 7. Learning curves on the train substation feeder use case. The structure of the figure mirrors the one from Figure 6.

Figure 7 clearly depicts that the ridge regression model was the worst model in this use case, with high standard deviation and a large gap between training and cross-validation test scores, which indicates high variance (over-fitting). A high gap between training and cross-validation test scores can be observed in results for decision trees. We can observe that more training examples improve the overall performance of the model, however, the gap does not converge. This indicates that the model can still be improved using more training data. The more or less consistent gap between training and testing scores can be observed with k-nearest neighbors (KNN) and random forest models, but the score is still increasing with more data, which again shows the more data could improve the overall results. But since KNN converged to much lower R^2 score than random forest, the latter would, of course, be the best choice. Regarding the features, we can observe the same pattern as with previous experimental results in Figure 6. Each additional feature set (data source) slightly improved the performance of the model, which suggests the benefits of our data fusion methodology for the modeling. On the other hand, keeping only the 20 most important features worked very well, since more or less all of the models deal with high variance (poor generalization).

Our framework provides out-of-the-box capabilities for the inclusion of different data sources (see Algorithm 3) and of the corresponding historical (see Algorithm 2), aggregated and aggregated historical values (see Algorithms 1 and 2). Inclusion of new features is possible with a single line in the use-case's configuration structure. Without extensive additional work, which would implement particular aggregating functions and book-keeping capabilities, in most other systems the modeling results would not exceed the ones, achieved with `M` or `M_AR` datasets. ISDI framework, which is the closest to ours in terms of functionality, provides windowing and fusion of multiple data streams, however, extraction of aggregated values is achieved with a custom user-defined function, which is batch-based and not optimized to the incremental nature of the IoT data. Additionally, ISDI's

performance deteriorates drastically with larger time windows. Our framework already provides built-in mechanisms for the enrichment and generalized inclusion of historical values (without any limitations on the size of windows).

6.2. Edge/Fog Infrastructure Deployment on Public Trains

The third use case demonstrates the usage of our methodology for stream data fusion, as well as for incremental learning, directly on an edge device where the measurements were taken (we have used Raspberry Pi 3) **on board a public train**. The main reason for this choice was because of the large amount of streaming data at a relatively fast pace. The experiments have been conducted on a data set containing 2 months of data with 1 second time resolution. The task was to predict the power consumption of a train with a very short term prediction horizon (10 s). The results are depicted in Figure 8.

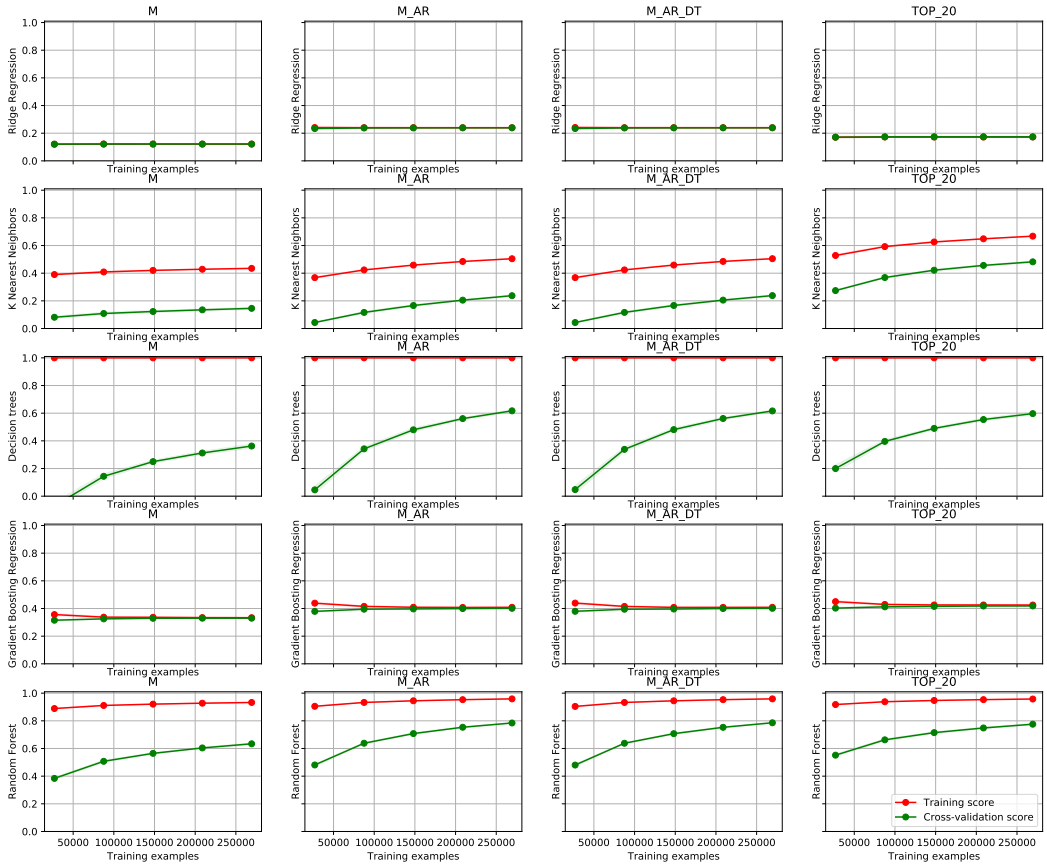


Figure 8. Learning curves on the train autonomous node use case. The structure of the figure mirrors the one from Figure 6, with the exception that weather data set (`M_AR_DT_WF`) is not included.

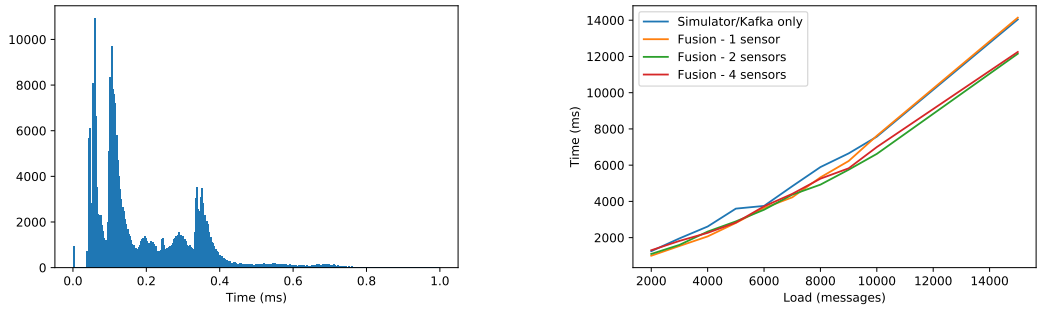
From Figure 8, we can observe that the positive trend of a test set score is rising in every subfigure. It is obvious, that the learning data set has been too small in this case. The benefits of our methodology are, however, apparent and can be observed in a general improvement of the test score with the number of used features (see differences in columns 1 through 3). With the KNN learning method, one can also observe drastic improvement with the data set with selected top 20 features. This exposes the need for a good feature selection methodology, which we have not yet implemented. Decision trees (again) show obvious overfitting to the training data set, however, the improvement of modeling is apparent. In this case, gradient boosting (we used the implementation in `scikit-learn`) shows poor adaptation to the data and exhibits that it can not be improved with more training data, only with better feature

engineering. The method of choice in the experiments is random forest, which achieves the highest test scores with any selected data set.

With the application of Algorithms 1 and 2 we improved the modeling results from the ones depicted in the first (`M`) to the ones depicted in the second (`M_AR`) column of Figure 8. With Algorithm 3 we included additional sources and achieved even better modeling results (see column `M_AR_DT` in Figure 8). With the simple configuration capabilities (https://github.com/klemenkenda/iot-fusion/blob/master/conf/train.js) of the framework, custom feature sets like `TOP_20` were implemented and deployed to the production in a matter of minutes.

6.3. Performance Tests

Applicability of the stream fusion framework has been tested from the performance perspective. The results are depicted in Figure 9. The first performance test has been conducted on a real-world smart grid use case data set with 10^6 messages. The setup included streaming fusion of three different types of sources (sensor, static and weather forecasts) and has generated a single feature vector (with 96 features based on current and historical aggregated values on 1-h to 1-month sliding windows) for 24-h prediction horizon. Response times of the fusion component (without modeling) have been measured and the results are depicted in the histogram in Figure 9a. We can observe three major peaks in the histogram. Each of the peaks represents a data source. The peak with the lowest response time corresponds to static data (the simplest data source), the middle one corresponds to weather data (long message with more complex integration subtasks) and the last peak corresponds to sensor data, which in some cases trigger the most computationally demanding fusion process. Median response time of the fusion component is approximately 0.21 ms, which means that the system is able to process approximately 5000 messages per second with a single thread process on an older high-end server (Intel Xeon CPU E5-2667 v2—3.3 GHz, 128 GB RAM, Windows Server 2012 R2). Each process could support the same performance. The measured throughput of Raspberry Pi 3 was approximately eight times slower (700–800 messages per second).



(**a**) (**b**)

Figure 9. Performance results of data fusion in the smart grid scenario. (**a**) The histogram represents the time used for processing a single message in the data fusion system. A message can be sensor, static or weather forecast data. (**b**) Load graph represents measured processing time of the data fusion system integrated into the Apache Kafka pipeline at different loads and numbers of sensors.

A typical frequency in the smart grid scenario is 15 min. This means that data fusion itself could support a smart grid with up to 18×10^6 messages per hour, which corresponds to 3×10^6 smart grid nodes per logical processor. ISDI [8] reports (in the best possible scenario) on the throughput of approx. 27×10^6 messages per hour per logical processor. The reported throughput is 50% faster than of our framework but it is computed for a single fixed time window (and the feature extraction function is unknown in their evaluation). Also, ISDI encounters major throughput breakdown with larger time windows (>10 days), which is not the case for our methodology. On the other hand, our methodology

computes multiple window sizes at the same time (from 1 h up to 1 month) as well as uses multiple aggregate functions. Window sizes can easily be expanded to 1 year with a very little performance cost.

The obvious bottleneck in such a setting is, however, not the fusion algorithm but the ML prediction algorithm. Using the usual method of choice in energy/environment related scenarios (RandomForest with 10 estimators) on the high-end server resulted in one prediction generated per approx. 0.5 s, which roughly corresponds to 7000 smart grid nodes. According to presented integration architectures, the capabilities can be scaled horizontally (over server cores and over other servers) and the final throughput is limited by the state-of-the-art message distribution frameworks such as Apache Kafka, which is apparent in Figure 9b.

7. Conclusions and Future Work

The paper describes a novel generic framework for building feature vectors for machine learning from heterogeneous streaming data sources on-line. The main benefit of this methodology is its universal applicability in real-world use cases. It also enables easy configuration of streaming data fusion and modeling pipelines as well as horizontal scalability due to the design patterns used in the architecture.

We have developed and used this methodology in a plethora of use cases, related mostly to efficient energy use, where regressive predictive models have been implemented at the end of the analytical pipeline. In the paper, we have demonstrated the usability of the methodology with three applications in use cases related to smart grids and transport, which are implemented in the cloud and in the edge computing device. The experimental results show that the proposed methodology improves modeling capabilities of the real-world IoT systems.

Future work in this domain should be dedicated firstly to a definition of a thorough evaluation methodology for streaming data fusion frameworks. In this paper, we have relied on an indirect approach with modeling. However, as discussed at the beginning of Section 6, a more thorough approach should be developed, which would take different aspects of data fusion into account.

We see several lines of possible extensions of the proposed framework as follows. Further simplification of data streams might contribute to the easier implementation and faster computation of results (i.e., static data, which is currently included as a stream, might be encoded with a function). Feature selection, dimensionality reduction, instance selection, instance reduction and concept drift are important research topics in the field and should be addressed in the framework. Some of the presented algorithms offer further optimization in terms of performance and expressiveness of the language for describing feature vectors. Stream discretization techniques (implemented via stream aggregates) could be further expanded to match the richness of their batched counterparts. Deployment into a large scale real-world use case states additional challenges related to the management of a large number of data fusion components and models, communications, etc. An efficient deployment system is crucial to ensure the practical scalability of the methodology. In many heterogeneous environments access control is of the utmost importance. Integration of the framework with state-of-the-art methods could be beneficial in various use cases (i.e., healthcare). Finally, the system's usage depends on the implemented incremental learning algorithms. As mentioned in the introduction, the implementations of the state-of-the-art non-linear incremental learning algorithms are scarce.

Taking into account the shortcomings mentioned above, the framework should find its place in the real-world applications within the IoT and bridge the current gap between academic achievements and practice.

Author Contributions: Conceptualization, K.K. and D.M.; methodology, K.K., E.N.; software, B.K. and K.K.; validation, B.K., D.M. and K.K.; formal analysis, K.K., E.N and B.K.; data curation, K.K. and B.K.; writing—original draft preparation, K.K.; writing—review and editing, K.K., B.K., E.N. and D.M.; visualization, B.K. and K.K.; supervision, D.M.; project administration, K.K.; funding acquisition, D.M. and K.K.

Funding: This research was funded by European Union's Horizon 2020 programme projects Water4Cities (Research and Innovation Staff Exchange) grant number 734409 and PerceptiveSentinel (Research and Innovation) grant number 776115.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
DB	Database
EMA	Exponential Moving Average
GUI	Graphical User Interface
IoT	Internet of Things
ISDI	IoT Streaming Data Integration
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbors (algorithm)
MA	Moving Average
MQTT	Message Queuing Telemetry Transport
REST	Representational State Transfer
RF	Random Forest
SPE	Stream Processing Engine
VFDT	Very Fast Decision Trees (also Hoeffding trees)
VHT	Vertical Hoeffding Trees

References

1. Rodríguez-Mazahua, L.; Rodríguez-Enríquez, C.A.; Sánchez-Cervantes, J.L.; Cervantes, J.; García-Alcaraz, J.L.; Alor-Hernández, G. A general perspective of Big Data: Applications, tools, challenges and trends. *J. Supercomput.* **2016**, *72*, 3073–3113. [CrossRef]
2. Ahmed, E.; Yaqoob, I.; Hashem, I.A.T.; Khan, I.; Ahmed, A.I.A.; Imran, M.; Vasilakos, A.V. The role of big data analytics in Internet of Things. *Comput. Netw.* **2017**, *129*, 459–471. [CrossRef]
3. Aggarwal, C.C. *Data Streams: Models and Algorithms (Advances in Database Systems)*; Springer: Secaucus, NJ, USA, 2006.
4. Gama, J.; Žliobaitė, I.; Bifet, A.; Pechenizkiy, M.; Bouchachia, A. A Survey on Concept Drift Adaptation. *ACM Comput. Surv.* **2014**, *46*, 44:1–44:37. [CrossRef]
5. Gepperth, A.; Hammer, B. Incremental learning algorithms and applications. In Proceedings of the European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, 27–29 April 2016.
6. Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. MOA: Massive Online Analysis. *J. Mach. Learn. Res.* **2010**, *11*, 1601–1604.
7. Manyika, J.; Chui, M.; Bisson, P.; Woetzel, J.; Dobbs, R.; Bughin, J.; Aharon, D. *Unlocking the Potential of the Internet of Things*; McKinsey Global Institute: New York, NY, USA, 2015.
8. Tu, D.Q.; Kayes, A.; Rahayu, W.; Nguyen, K. ISDI: A New Window-Based Framework for Integrating IoT Streaming Data from Multiple Sources. In Proceedings of the International Conference on Advanced Information Networking and Applications, Matsue, Japan, 27–29 March 2019; Springer: Berlin, Germany, 2019; pp. 498–511.
9. Babcock, B.; Babu, S.; Datar, M.; Motwani, R.; Widom, J. Models and Issues in Data Stream Systems. In Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Madison, WI, USA, 3–5 June 2002; ACM: New York, NY, USA, 2002; pp. 1–16. [CrossRef]
10. Kandel, S.; Heer, J.; Plaisant, C.; Kennedy, J.; van Ham, F.; Riche, N.H.; Weaver, C.; Lee, B.; Brodbeck, D.; Buono, P. Research directions in data wrangling: Visualizations and transformations for usable and credible data. *Inf. Vis.* **2011**, *10*, 271–288. [CrossRef]

11. Fan, W.; Bifet, A. Mining Big Data: Current Status, and Forecast to the Future. *SIGKDD Explor. Newsl.* **2013**, *14*, 1–5. [CrossRef]
12. Krempl, G.; Žliobaite, I.; Brzeziński, D.; Hüllermeier, E.; Last, M.; Lemaire, V.; Noack, T.; Shaker, A.; Sievi, S.; Spiliopoulou, M.; et al. Open challenges for data stream mining research. *ACM SIGKDD Explor. Newsl.* **2014**, *16*, 1–10. [CrossRef]
13. Yang, Q.; Wu, X. 10 Challenging Problems in Data Mining Research. *Int. J. Inf. Technol. Decis. Mak.* **2006**, *5*, 597–604. [CrossRef]
14. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* **2017**, *239*, 39–57. [CrossRef]
15. Zhang, L.; Xiao, N.; Yang, W.; Li, J. Advanced Heterogeneous Feature Fusion Machine Learning Models and Algorithms for Improving Indoor Localization. *Sensors* **2019**, *19*, 125. [CrossRef]
16. Bouguelia, M.R.; Karlsson, A.; Pashami, S.; Nowaczyk, S.; Holst, A. Mode tracking using multiple data streams. *Inf. Fus.* **2018**, *43*, 33–46. [CrossRef]
17. Kong, J.L.; Wang, Z.N.; Jin, x.b.; Wang, X.Y.; Su, T.L.; Wang, J.L. Semi-Supervised Segmentation Framework Based on Spot-Divergence Supervoxelization of Multi-Sensor Fusion Data for Autonomous Forest Machine Applications. *Sensors* **2018**, *18*, 61. [CrossRef]
18. Wu, J.; Feng, Y.; Sun, P. Sensor Fusion for Recognition of Activities of Daily Living. *Sensors* **2018**, *18*, 4029. [CrossRef]
19. Ma, M.; Song, Q.; Gu, Y.; Li, Y.; Zhou, Z. An Adaptive Zero Velocity Detection Algorithm Based on Multi-Sensor Fusion for a Pedestrian Navigation System. *Sensors* **2018**, *18*, 3261. [CrossRef]
20. Zhou, Y.; Xue, W. A Multisensor Fusion Method for Tool Condition Monitoring in Milling. *Sensors* **2018**, *18*, 3866. [CrossRef]
21. Shi, P.; Li, G.; Yuan, Y.; Kuang, L. Data Fusion Using Improved Support Degree Function in Aquaculture Wireless Sensor Networks. *Sensors* **2018**, *18*, 3851. [CrossRef]
22. Zhou, F.; Hu, P.; Yang, S.; Wen, C. A Multimodal Feature Fusion-Based Deep Learning Method for Online Fault Diagnosis of Rotating Machinery. *Sensors* **2018**, *18*, 3521. [CrossRef]
23. Lu, K.; Yang, L.; Seoane, F.; Abtahi, F.; Forsman, M.; Lindecrantz, K. Fusion of Heart Rate, Respiration and Motion Measurements from a Wearable Sensor System to Enhance Energy Expenditure Estimation. *Sensors* **2018**, *18*, 3092. doi:10.3390/s18093092. [CrossRef]
24. Hu, J.; Huang, T.; Zhou, J.; Zeng, J. Electronic Systems Diagnosis Fault in Gasoline Engines Based on Multi-Information Fusion. *Sensors* **2018**, *18*, 2917. [CrossRef]
25. Wu, B.; Huang, T.; Jin, Y.; Pan, J.; Song, K. Fusion of High-Dynamic and Low-Drift Sensors Using Kalman Filters. *Sensors* **2019**, *19*, 186. [CrossRef]
26. Akbar, A.; Kousiouris, G.; Pervaiz, H.; Sancho, J.; Ta-Shma, P.; Carrez, F.; Moessner, K. Real-Time Probabilistic Data Fusion for Large-Scale IoT Applications. *IEEE Access* **2018**, *6*, 10015–10027. [CrossRef]
27. Kayes, A.; Rahayu, W.; Dillon, T.; Chang, E.; Han, J. Context-aware access control with imprecise context characterization for cloud-based data resources. *Future Gener. Comput. Syst.* **2019**, *93*, 237–255. [CrossRef]
28. Colombo, P.; Ferrari, E. Fine-Grained Access Control Within NoSQL Document-Oriented Datastores. *Data Sci. Eng.* **2016**, *1*, 127–138. [CrossRef]
29. Kayes, A.S.M.; Rahayu, W.; Dillon, T. Critical situation management utilizing IoT-based data resources through dynamic contextual role modeling and activation. *Computing* **2018**. [CrossRef]
30. Colombo, P.; Ferrari, E. Access Control Enforcement Within MQTT-based Internet of Things Ecosystems. In Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies, Indianapolis, IN, USA, 13–15 June 2018; ACM: New York, NY, USA, 2018; pp. 223–234. [CrossRef]
31. Zhang, K.; Li, X.R.; Zhu, Y. Optimal update with out-of-sequence measurements. *IEEE Trans. Signal Process.* **2005**, *53*, 1992–2004. [CrossRef]
32. Khaleghi, B.; Khamis, A.; Karray, F. Multisensor Data Fusion: A Data-Centric Review of the State of the Art and Overview of Emerging Trends. In *Multisensor Data Fusion: From Algorithms and Architectural Design to Applications*; Fourati, H., Ed.; CRC Press: Boca Raton, FL, USA, 2015; pp. 15–33.
33. Lahat, D.; Adali, T.; Jutten, C. Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects. *Proc. IEEE* **2015**, *103*, 1449–1477. [CrossRef]
34. García, S.; Ramírez-Gallego, S.; Luengo, J.; Benítez, J.M.; Herrera, F. Big data preprocessing: Methods and prospects. *Big Data Anal.* **2016**, *1*, 9. [CrossRef]

35. Zliobaite, I.; Gabrys, B. Adaptive Preprocessing for Streaming Data. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 309–321. [CrossRef]
36. Stonebraker, M.; Çetintemel, U.; Zdonik, S. The 8 Requirements of Real-time Stream Processing. *ACM Sigmod Rec.* **2005**, *34*, 42–47. [CrossRef]
37. Gaber, M.M.; Zaslavsky, A.; Krishnaswamy, S. Mining Data Streams: A Review. *ACM Sigmod Rec.* **2005**, *34*, 18–26. [CrossRef]
38. Domingos, P.; Hulten, G. Mining high-speed data streams. In Proceedings of the KDD 2000—Sixth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Boston, MA, USA, 20–23 August 2000; Volume 2, p. 4.
39. Manapragada, C.; Webb, G.I.; Salehi, M. Extremely Fast Decision Tree. In Proceedings of the KDD 2018—24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; ACM: New York, NY, USA, 2018; pp. 1953–1962. [CrossRef]
40. Kourtellis, N.; Morales, G.D.F.; Bifet, A.; Murdopo, A. VHT: Vertical Hoeffding Tree. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 915–922.
41. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Netw.* **2019**, *113*, 54–71. [CrossRef]
42. Gama, J.; Sebastião, R.; Rodrigues, P.P. On evaluating stream learning algorithms. *Mach. Learn.* **2013**, *90*, 317–346. [CrossRef]
43. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache spark: A unified engine for big data processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
44. Kleppmann, M.A.; Kreps, J. Kafka, Samza and the Unix philosophy of distributed data. *IEEE Data Eng. Bull.* **2015**, *38*, 4–14.
45. Carbone, P.; Katsifodimos, A.; Ewen, S.; Markl, V.; Haridi, S.; Tzoumas, K. Apache flink: Stream and batch processing in a single engine. *Bull. IEEE Comput. Soc. Tech. Comm. Data Eng.* **2015**, *36*.
46. Pathak, H.; Rathi, M.; Parekh, A. Introduction to Real-Time Processing in Apache Apex. *Int. J. Res. Advent Technol.* **2016**, 19.
47. Bifet, A.; Zhang, J.; Fan, W.; He, C.; Zhang, J.; Qian, J.; Holmes, G.; Pfahringer, B. Extremely Fast Decision Tree Mining for Evolving Data Streams. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; ACM: New York, NY, USA, 2017; pp. 1733–1742. [CrossRef]
48. Montiel, J.; Read, J.; Bifet, A.; Abdessalem, T. Scikit-Multiflow: A Multi-output Streaming Framework. *J. Mach. Learn. Res.* **2018**, *19*, 1–5.
49. Fortuna, B.; Rupnik, J.; Brank, J.; Fortuna, C.; Jovanoski, V.; Karlovcec, M.; Kazic, B.; Kenda, K.; Leban, G.; Mladenić, D.; et al. QMiner: Data Analytics Platform for Processing Streams of Structured and Unstructured Data. In Proceedings of the Software Engineering for Machine Learning Workshop, Neural Information Processing Systems, Montreal, QC, Canada, 8–12 December 2014.
50. Yi, W.; Teng, F.; Xu, J. Novel Stream Data Mining Framework Under the Background of Big Data. *Cybern. Inf. Technol.* **2016**, *16*, 69–77. [CrossRef]
51. Marz, N.; Warren, J. *Big Data: Principles and Best Practices Of Scalable Real-Time Data Systems*; Manning Publications Co.: New York, NY, USA, 2015.
52. Ta-Shma, P.; Akbar, A.; Gerson-Golan, G.; Hadash, G.; Carrez, F.; Moessner, K. An Ingestion and Analytics Architecture for IoT Applied to Smart City Use Cases. *IEEE Internet Things J.* **2018**, *5*, 765–774. [CrossRef]
53. Kolomvatsos, K.; Anagnostopoulos, C.; Hadjiefthymiades, S. Data Fusion and Type-2 Fuzzy Inference in Contextual Data Stream Monitoring. *IEEE Trans. Syst. Man, Cybern. Syst.* **2017**, *47*, 1839–1853. [CrossRef]
54. Wu, X.; Zhu, X.; Wu, G.Q.; Ding, W. Data Mining with Big Data. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 97–107. [CrossRef]
55. Kenda, K.; Škrjanc, M.; Borštnik, A. Modelling of the complex data space: Architecture and use cases from NRG4CAST project. In Proceedings of the 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA), Corfu, Greece, 6–8 July 2015; pp. 1–4.

56. Tekin, C.; Canzian, L.; van der Schaar, M. Context-adaptive big data stream mining. In Proceedings of the 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 30 September–1 October 2014; pp. 483–490. [CrossRef]
57. Christ, M.; Kempa-Liehr, A.W.; Feindt, M. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv* **2016**. arXiv:1610.07717.
58. Christ, M.; Braun, N.; Neuffer, J.; Kempa-Liehr, A.W. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh—A Python package). *Neurocomputing* **2018**, *307*, 72–77. [CrossRef]
59. Gusev, M. A dew computing solution for IoT streaming devices. In Proceedings of the 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 22–26 May 2017; pp. 387–392.
60. van Laere, J. Challenges for IF performance evaluation in practice. In Proceedings of the 2009 12th International Conference on Information Fusion, Seattle, WA, USA, 6–9 July 2009; pp. 866–873.

Chapter 4

Feature Selection Using Multi-Objective Optimization

> Natural selection is the preservation of favored races in the struggle for life.
>
> *Charles Darwin*

This chapter presents the paper titled *FASTENER feature selection for inference from earth observation data* by Filip Koprivec, Klemen Kenda and Beno Šircelj published in Entropy journal (IF 2020: 2.524) [5]. Filip Koprivec and Klemen Kenda share the first authorship of the paper. Klemen Kenda contributed to conceptualization and methodology, validation, writing and also to project administration and funding acquisition. Klemen Kenda was supervising the work of the other co-authors.

Feature selection represents a crucial step in optimizing the performance of the model. Based on the evaluation criterion, the feature selection methods are divided into filter and wrapper methods. Filter methods utilize the statistical performance of the data it trains for feature evaluation and do not have a relation to the learning algorithm used. On the other hand, wrapper methods use the training accuracy of learning algorithms. Algorithms can, therefore, capture more specific behavior at the expense of computational complexity. For wrapper algorithms, feature selection represents a search in the feature space. The task of finding a minimal feature set that yields the best possible model results is proven to be an NP problem [22].

Wrapper algorithms employ different search algorithms for feature selection: complete search, heuristic search, random search [23]. A complete search can find the optimal solution; however it often demands relatively large computational requirements. To find the optimal feature subset, the M-feature combinations of N original features must be searched. Therefore, in most practical situations, an exhaustive search cannot be achieved [23].

FASTENER algorithm belongs to the family of random search approaches. It was originally developed to be used in land-use classification based on earth observation data, however, we have shown that the algorithm performs very well also on several other use cases, including environmental and biological datasets. FASTENER exploits entropy-based measures, such as mutual information in the crossover phase of the iterative genetic approach. FASTENER outperforms other multi-objective wrapper methods in terms of converging to a (near) optimal subset of features more quickly.

We evaluate the performance of wrapper methods with an area under (Pareto) front (AUF), where the front is represented with F_1 scores of optimal k selected features for a particular classification problem. FASTENER outperforms the POSS [24] and FS-SDS [25] algorithms in Earth observation data sets with an average AUF of 0.71, compared to 0.66 and 0.62 for POSS and FS-SDS, respectively. FASTENER also produces surprisingly good results in the benchmark datasets for feature selection [26], where its average AUF score is better than the FS-SDS AUF score in almost all, except one out of 25 datasets. On average, the difference between AUF scores between FS-SDS and FASTENER in 0.08, where FASTENER needed more than 2.5-times less model evaluations. Compared to DT-forward algorithm [27], FASTENER yields better results in nine out of 25 benchmark datasets. It is worth mentioning that FASTENER requires on average 2 to 5 times fewer model evaluations than DT-forward, while achieving comparable or better results.

Finally, FASTENER demonstrates superior classification accuracy when compared with similarity and information theory-based techniques typically applied in Earth observation scenarios. FASTENER was also tested on open feature selection datasets and compared with state-of-the-art methods. It achieves comparable results, however, with fewer model evaluations. FASTENER can be used in any supervised machine learning scenario.

entropy

MDPI

Article

FASTENER Feature Selection for Inference from Earth Observation Data

Filip Koprivec [1,2,3,†], Klemen Kenda [1,4,*,†] and Beno Šircelj [1]

[1] Jožef Stefan Institute, 1000 Ljubljana, Slovenia; filip.koprivec@ijs.si (F.K.); beno.sircelj@ijs.si (B.Š.)
[2] Faculty of Mathemathics and Physics, University of Ljubljana, 1000 Ljubljana, Slovenia
[3] Institute of Mathematics, Physics, and Mechanics, 1000 Ljubljana, Slovenia
[4] Jozef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia
* Correspondence: klemen.kenda@ijs.si
† These authors contributed equally to this work.

Received: 19 May 2020; Accepted: 21 October 2020; Published: 23 October 2020

Abstract: In this paper, a novel feature selection algorithm for inference from high-dimensional data (FASTENER) is presented. With its multi-objective approach, the algorithm tries to maximize the accuracy of a machine learning algorithm with as few features as possible. The algorithm exploits entropy-based measures, such as mutual information in the crossover phase of the iterative genetic approach. FASTENER converges to a (near) optimal subset of features faster than other multi-objective wrapper methods, such as POSS, DT-forward and FS-SDS, and achieves better classification accuracy than similarity and information theory-based methods currently utilized in earth observation scenarios. The approach was primarily evaluated using the earth observation data set for land-cover classification from ESA's Sentinel-2 mission, the digital elevation model and the ground truth data of the Land Parcel Identification System from Slovenia. For land cover classification, the algorithm gives state-of-the-art results. Additionally, FASTENER was tested on open feature selection data sets and compared to the state-of-the-art methods. With fewer model evaluations, the algorithm yields comparable results to DT-forward and is superior to FS-SDS. FASTENER can be used in any supervised machine learning scenario.

Keywords: feature selection; machine learning; earth observation; genetic algorithm; information theory

1. Introduction

Land cover classification based on satellite imagery is one of the most common and well-researched machine learning (ML) tasks in the Earth Observation (EO) community. In Europe, the Copernicus Sentinel-2 missions provide large amounts of global coverage EO data. With 5-day revisit times, Sentinel-2 generated a total of 9.4 PB of satellite imagery by April 2020 [1]. The total amount of EO data available through the Copernicus services is estimated to exceed 150 PB. The data represent an opportunity for building solutions in various domains (agriculture, water management, biology or law enforcement); however, computationally efficient methods that yield sufficient results are needed to deal with this big data source. The most important issue when trying to maximize the accuracy of statistical learning methods is effective feature engineering. With many features, the algorithms become slower and consume more resources. In this paper, we present FASTENER (FeAture SelecTion ENabled by EntRopy, https://github.com/JozefStefanInstitute/FASTENER) genetic algorithm for efficient feature selection, especially in EO tasks.

EO data sets are characterized by a large number of instances (millions or even billions) and a fair number of features (hundreds). These characteristics differ from the usual feature selection data sets that often contain up to several hundred instances and usually thousands of features. FASTENER

exploits entropy-based measures to converge to a (near) optimal set of features for statistical learning faster than other algorithms. The algorithm can be used in many scenarios and can, for example, complement multiple-source data fusion and extensive feature generation in the fields of natural language processing, DNA microarray analysis or the Internet of Things [2]. FASTENER has been tested on a land cover classification EO data set from Slovenia. Additionally, we have tested the algorithm against 25 general feature selection data sets.

The main evaluation use case presented in this paper is based on our previous contributions [3,4] regarding crop (land cover) classification using Sentinel-2 data from the European Space Agency (ESA). The scientific contributions of our work are as follows:

1. A novel genetic algorithm for feature selection based on entropy (FASTENER). Such an algorithm reduces the number of features while preserving (or even improving) the accuracy of the classification. The algorithm is particularly useful for data sets containing a large number of instances and hundreds of features. A reduced number of features reduces learning and inference times of classification algorithms as well as the time needed to derive the features. By using an entropy based approach, the algorithm converges to the (near) optimal subset faster than competing methods.
2. Improvement of the state-of-the-art in feature selection in the field of Earth Observation. FASTENER yields better result than current state-of-the-art algorithms in remote sensing. The algorithm has been tested and compared to other methods within the scope of the land-cover classification problem.
3. Usage of pre-trained models for information gain calculation for feature selection in Earth Observation scenarios. Usually, it is computationally expensive to train a machine learning model. The inference phase is much faster. FASTENER exploits the pre-trained models in order to estimate the information gain of certain features by inferring from data sets with randomly permuted values of the evaluated feature. This approach improves the convergence speed of the algorithm.

The paper is structured as follows. In the next section, related work regarding feature selection algorithms as well as state-of-the-art in feature selection in Earth Observation is described. Section 4 contains a thorough description of the data used in the experiments and describes our algorithm. Section 5 describes while Section 6 discusses the experimental results and compares FASTENER to other algorithms based on NSGA-II. Finally, our conclusions are presented in Section 7.

2. Related Work

When tackling problems related to EO, quite often the older and simpler algorithms (ReliefF) have given the best results. This section presents the current state-of-the-art in feature selection and particularly its applications within the EO community. It also describes supporting fields of the work presented in this paper, such as the current status in land-cover classification and the multi-objective optimization approach to feature selection.

2.1. Multi-Objective Optimization and Its Usage in Feature Selection

Data science often reduces its evaluation to a single score (or fitness function) that explains the accuracy or efficiency of a particular methodology. There are other limitations that should be considered. On the one hand, there are the available resources (e.g., memory, processors, space), while on the other hand the response time should be considered (i.e., the preferred response time for a particular methodology). Quite often, the most appropriate approach should minimize the need for resources and response time while maintaining sufficient accuracy. The multi-objective optimization [5] strives to find a set (or Pareto front) of solutions (in our case feature sets) that achieve the best possible classification accuracy with a limited number of features.

Multi-objective optimization methods have already been used for feature selection purposes, although their usage in the field is limited due to their computational complexity [6]. The classical

NSGA-II algorithm [5] was used for feature subset optimization in [7]. NSGA-II uses fast non-dominated sorting for selecting viable candidates for the next generation parent population. The methodology has been improved by the usage of Reduced Pareto Set Genetic Algorithm with Elitism (RPSGAe) [8] for achieving a more efficient directed search in the feature space [9]. The elite-preserving operator suppresses the deterioration of the population fitness along with the successive generations. RPSGAe reduces the number of solutions on the efficient (Pareto) front while maintaining its characteristics intact, similarly to the purge function in the FASTENER algorithm. NSGA-II feature selection approach has also been extended in [10] with six different importance measures, including representation entropy which is based on information theory. The research shows that there is no preferred importance measure that would be a good fit for all feature selection problems. FASTENER uses importance measures that are based on real model evaluations and are therefore more reliable than estimates based on the features themselves.

Pareto-optimal subset selection (POSS) [11] is also a genetic algorithm, but it only uses mutation techniques (while NSGA-II also uses cross-over methods). Other methods explore several variants of directed searches within the feature space to converge to the sub-optimal data set faster (e.g., FS-SDS [12] uses stochastic diffusion search and extreme learning machine [13] as an embedded classifier). Other methods, such as forward selection and backward elimination use an exhaustive step-by-step search to find the most optimal feature set [14]. These methods are much slower (and quite often do not converge to a small-enough data set within a reasonable time) and could even converge to a local minimum.

Our approach extends POSS and NSGA-II algorithms and suggests the usage of entropy-based indices based on previously trained models to steer the iterative feature selection process. The parameterized mutation strategy introduces additional features to the candidate feature sets with respect to changes in the Pareto front. The Pareto front is periodically purged in order to eliminate non-suitable candidates from the loop. With these improvements, FASTENER converges to a (nearly) optimal solution faster than the compared methods.

2.2. Feature Selection and Dimensionality Reduction for EO Tasks

The automatic spectro-temporal feature selection (ASTFS) [15] workflow provides a search over the feature space by incrementally extending the feature set with effective features (those that improve classification scores) ordered by their global separability index [16]. A similar approach is implemented in [17], where the authors use mutual information (MI) [18] and Fisher's criterion to select the k most important features. Mutual information based on entropy estimates from k-nearest neighbour distances [18] is data-efficient and has a minimal bias. The problem of estimating MI between discrete and continuous features has been solved in [19], which can be applied to classification tasks (discrete) with continuous features. Although the separability index and mutual information provide good heuristics for adding different features, the non-iterative (non-wrapping) algorithms do not necessarily yield the optimal feature set. Our algorithm offers a more thorough search, which is controlled by the genetic nature of the algorithm. The approach has been tested using timeless features for land cover classification [20,21].

A revision and test of multiple feature selection algorithms (similarity-based, statistical, sparse learning, information theoretical, and wrapper methods) [22] showed that ReliefF (a similarity-based approach) [23], although not among the recent favourites, yielded the best results in a particular scenario of parthenium weed infestation detection.

Our algorithm can be applied to any feature selection scenario as it is expected to converge to the (near) optimal subset selection. FASTENER is expected to perform at least as good as any other similarity-based approaches; however, several iterations would be required during the learning process to find the (near) optimal subset of features.

Genetic algorithms have been used in EO scenarios for feature selection of hyper-spectral EO images [24]. In contrast to multi-spectral (as in our case), hyper-spectral images contain a significantly

larger number of bands. The approach in [24] does not use any entropy-based features to optimize the search and is, in that respect, similar to POSS. The latter has been tested in land cover classification [4] and yielded good results.

In an EO environment, where data indices are abundant, the calculation of some features is computationally very intensive and external data are difficult to acquire (weather data, soil samples). Reducing the number of features has a significant positive impact on the overall algorithm performance in a real-world scenario. As the production of global EO products sometimes take thousands of computing hours, efficient feature selection algorithms could reduce the costs by 50–95%.

2.3. Land Cover Classification and Feature Engineering

There are essentially three approaches to land cover classification based on EO data [25]. The first approach is object-based and focused on a single image, while the other two approaches are pixel-based. From the latter, a simpler approach is based on single-image data, while the other is based on a time-series of images. Single image approaches depend on the current situation (e.g., cloud cover), whereas time series based approaches inherently overcome this shortcoming. Sentinel-2 mission from the Copernicus programme is dedicated to agriculture and their satellites re-visit times are 5 days. It is possible to construct an interpolated time-series of different EO sensors throughout the entire vegetation period [3]. A simple feature extraction involves taking particular sensor values and derived features at pre-selected points in time and combining them into a feature vector. A more intelligent approach involves the extraction of timeless features such as the maximum value of a particular time series, length of the maximum interval, steepness of the steepest slope and others [20,21]. Our evaluation of the FASTENER algorithm is based on the latter approach [3].

The best accuracy scores in land cover classification were achieved with deep learning [26]. These approaches reduce the value of feature engineering that originates mostly from domain experts and/or extensive experimental knowledge. Deep neural networks are able to derive the important features exclusively from raw data. Of course, the data must be extensive, which is the case in EO. However, this comes with a price; namely, extreme computational requirements reduce the usability of these methods.

Our goal is to develop lean EO machine learning models that are capable of capturing most of the information with limited resources. Intelligent feature selection, together with smart feature engineering and a pragmatic choice of the learning models is one of the three pillars of smart EO.

3. FASTENER—A Genetic Algorithm for Feature Selection

The presented algorithm is based on the POSS genetic algorithm and the Pareto ensemble pruning proposed in [27,28]. The main objective of the algorithm is to select an optimal subset of available features from a large feature set ($N > 100$). The problem could be represented mathematically as finding the optimal (or near-optimal) binary vector x of an indexed set $\{0,1\}^N$. Finding the optimal subset is NP-complete. Each element in x corresponds to a particular feature that is included or excluded in our decision model. The optimality of such a set is measured by a scoring function, which we will denote by $A(M;x)$ (accuracy measure of model M on a subset of features x). Some standard examples of such measures in machine learning classification problems include precision, recall, accuracy and F_1 score. In our experiments, the F_1 score was used. We also adopt the notation $|x|$ to denote the number of bits set in x that directly corresponds to the number of selected features. Mathematically, for a fixed classification model M and the number of features k, we search for such an x where the following maximum is reached:

$$\max_{\substack{x\in\{0,1\}^N \\ |x|=k}} A(M;x).$$

As in POSS, the function $x \mapsto (|x|, A(M;x))$ creates a two-dimensional Pareto front. In one dimension, we evaluate k, the number of selected features, and in the other dimension, we present A score, which represents the optimality (e.g., accuracy) of such a subset. In this space, the ordering of instances is defined as follows. The pair (k_1, s_1) dominates the pair (k_2, s_2) if and only if $k_1 \leq k_2$ and $s_1 \geq s_2$. Intuitively, the smaller set (by cardinality) of features provides classification results, which are at least as good as for the larger set. We denote such a relation by $(k_2, s_2) \preceq (k_1, s_1)$. The relation is obviously transitive but does not induce linear ordering. Pairs that are not comparable through such a relation are of particular interest, as they lie on the Pareto front. Feature subsets that lie on the Pareto front (not strongly dominated by other subsets) are the desired subsets. They provide the best classification power using the smallest number of features. The final result of a feature selection algorithm is the final Pareto front. The final Pareto front can be seen as an accumulation of the best results for a fixed k. The features with the best performance can then be selected from the front automatically (with a certain cut-off threshold), manually or by using additional information (e.g., time to calculate the features). The results clearly show that the Pareto front "plateaus" after the inclusion of some features and this fact can easily be used to automatically select the best number of features.

Our modification of the POSS algorithm combines Pareto front searching with a genetic algorithm to incorporate additional statistical information when recombining genes. The main ingredient in the algorithm is the concept of an *item*; a subset of features and *scored item*, which is an item with an assigned score corresponding to such a subset of features. The size of an item denoted by $|\text{ITEM}|$ corresponds to the number of features selected. The algorithm works by successively evaluating items, combining them, and updating the Pareto front.

Each subset of features can be represented directly as a binary number, where set bits correspond to the included (selected) features and unset bits correspond to excluded ones. This allows for a natural representation of genes for such an item with a binary string or bit-set and also gives a natural human-readable representation as an arbitrary sized integer. Apart from this, many later needed operations can be quickly implemented as simple operations on a binary string. The size of an item corresponds to the number of set bits, the bit-wise AND operation between two items corresponds to the genes that are common to both of them, while the bit-wise XOR returns all different selected features.

In our implementation, the integer representation of item genes is treated in a little-endian way, as this allows easier extension of feature sets and keeps integer representations valid even when adding new features. For example, in a set of six features with 32 possible feature combinations, binary string `11010` represents a subset containing features with indices 0, 1 and 3 and is represented by a decimal number 11. Even if the number of features is increased, such a subset would still be represented by the number 11, albeit its binary string representation would be extended by zeros on the right side.

An example of the incremental Pareto front update is shown in Figure 1. Each subsequent generation of the FASTENER algorithm produces a Pareto front that dominates the Pareto fronts from previous generations (in rare cases, where no improvement is made, the Pareto fronts of two subsequent generations may be the same). The main objective of the algorithm is to converge as quickly as possible to a theoretically limiting front (in terms of fitness function evaluations).

Figure 1. Iterative improvements of Pareto front from the first generation. Each next generation's Pareto front achieves better F_1 score.

3.1. Algorithm

The algorithm stores the current Pareto-optimal items and progresses in generations. For each $k \leq N$ it keeps an item with the best score A. If there is no such item this means either that it is strictly dominated by another item or that the algorithm has not yet evaluated an item this size. In our implementation, the Pareto front is implemented as a simple dictionary. Apart from the Pareto front, the current population is kept in a separate set. The decoupling of the current population from the Pareto front allows the algorithm to "explore" different combinations more easily, while the Pareto front continues to be updated at each iteration.

As usual, in each generation, the current population of items is mated, a random genetic mutation is introduced and newly acquired items are evaluated. For each evaluation, the Pareto front is updated. If a new item is strictly better than an item on the front with the same size, the newly evaluated item is placed on the Pareto front. After all new items have been evaluated, the front is purged by removing items that are strictly dominated by other items. The pseudo-code for the basic algorithm is presented in Algorithm 1. The algorithm's flow diagram is depicted in Figure 2.

To prevent the search phase of the algorithm from diverging (the population as a whole is not controlled by the scoring function), the population is periodically reset to the current Pareto optimal items. In the presented algorithm, this decision is based only on the generation number; however, our implementation also considers the rate with which the running Pareto front is being updated between generations.

Algorithm 1 Basic algorithm

Require:
Initial population POP
Evaluation function EVALF
Number of iterations K
Mating pool selection strategy SELECTPOOL
Crossover strategy CROSSOVER
Mutation strategy MUTATE
Predicate RESETTOFRONT
Predicate PURGEFRONTPREDICATE
Front purging procedure PURGEFRONT

```
1:  function FeatureSubsetSelection
2:      front ← pop
3:      for gen = 1 to K do
4:          pool ← selectPool(pop)
5:          pool ← crossover(pool)
6:          pop ← mutate(pop ∪ pool)
7:          pop ← evaluate(pop, evalF)
8:          front ← front ∪ pop                ▷ Add evaluated population to Pareto front
9:          front ← removeDominated(front)     ▷ Remove unoptimal items from Pareto front
10:         if purgeFrontPredicate(gen) then
11:             front ← purgeFront(front)
12:         end if
13:         if resetToFront(gen) then
14:             pop ← front
15:         end if
16:     end for
17:     return front
18: end function
```

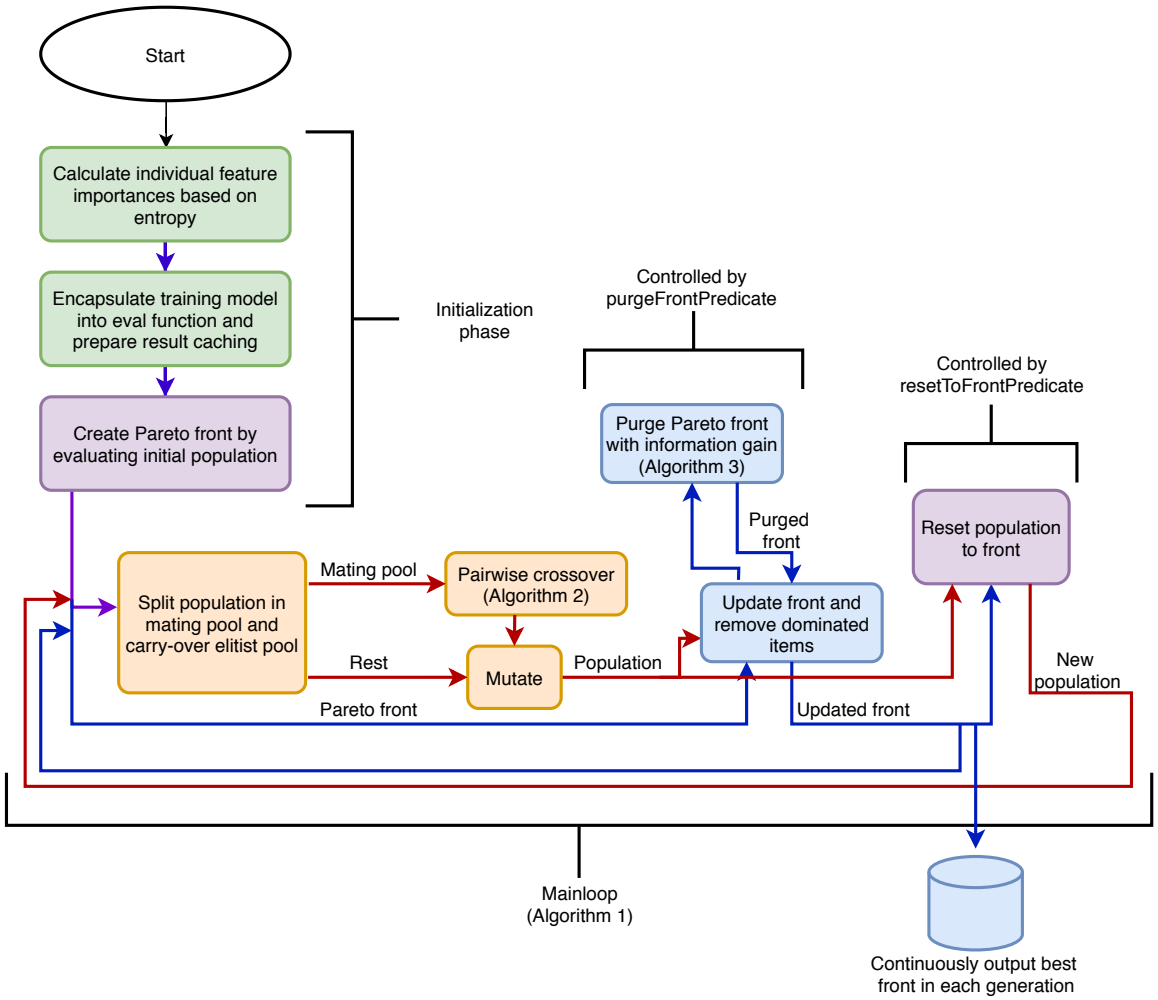


Figure 2. The flow diagram of the FASTENER graphically represents Algorithm 1 and puts Algorithms 2 and 3 into context. In the figure, red/orange colour objects represent population-based operations, while blue colour objects represent Pareto front related operations. The violet colour represents both, the population and the Pareto front. Green boxes refer to technical details of the algorithm. The algorithm includes the initialization phase and the main loop. In the initialization phase, the initial population is prepared and evaluated and the technical prerequisites for the algorithm are created. In the main loop, the population is first split into a mating pool and a (gene) carry-over elitist pool. The mating pool first enters the crossover phase based on Algorithm 2 and is being mutated together with the elitist pool. Then the new Pareto front is updated and purged (Algorithm 3). Finally, the main loop is closed by registering the new population as the next generation Pareto front.

3.2. Mutation

An important part when considering gene manipulation and modification is to keep the overall objective in mind. We want to find the optimal subset of features with good accuracy characteristics. The ultimate goal is to obtain a subset with a size much smaller than N. In preliminary experiments, the number of features with a sufficiently predictive power was around $\sqrt{N}$, which means a significant reduction in both training time and in data preparation time (which is true especially for EO data sets). Mutation and crossover methods should therefore be tailored to help maintain, reduce or only slightly increase the size of items.

The most interesting parts of the algorithm are the CROSSOVER and MUTATE procedures. The procedure for the mutation is adopted from the POSS algorithm and is a simple random bit mutation. Each random bit in the gene is flipped independently with a probability $\frac{1}{N}$. Thus a new

item is generated from each of the items selected for mating and, most importantly, the expected value of set bits is kept approximately the same. Formally, the expected value of the number of flips is 1, and therefore the mutations gravitate slowly toward x-s where $|x| = \frac{N}{2}$. Since the desired range of features in which we can expect satisfactory score results is much lower than the asymptotic behaviour of mutations, this type of mutation serves as a size increaser. Mutation procedures therefore slowly increase and add new features to be considered. Furthermore, varying the probability of flipping is an easy way to control the speed of the introduction of new features. Setting the mutation rate to $\frac{a}{N}$ for $a > 1$ and varying a with generation number and changes in the Pareto front updates can significantly increase the search space. In addition, all further analyses are still valid even if the mutation is set as an arbitrary function $f : \mathbb{N}^+ \to (0, 1]$ only dependent on N and heuristic data about mutations (independent of bit-index and bit status), called *mutation function*.

3.3. Crossover

The crossover procedure is designed to reduce the size of the newly produced item. The mating pool selection used for the algorithm is a simple random mating pool with a fixed size. Two different items from the mating pool are combined using the CROSSOVERPAIR procedure. The CROSSOVERPAIR procedure itself is presented in Algorithm 2 and requires 4 input parameters: two items to be mated, a scaling function for the number of non-intersected genes, and a set of entropy information for each individual feature. Without loss of generality, we may assume that $|\text{ITEM1}| \geq |\text{ITEM2}|$.

Lemma 1. *Let* ITEM1 *and* ITEM2 *be a random independent subset of* $\{0, 1\}^N$ *where the probability of each gene being set is some positive function* $f : \mathbb{N}^+ \to (0, 1]$. *The expected size of an intermediate gene conditional on the sizes of its parent genes is*

$$\mathbb{E}\left[|\text{ITEM1} \cap \text{ITEM2}| \mid |\text{ITEM1}| = a, |\text{ITEM2}| = b\right] = \frac{ab}{N} = \frac{|\text{ITEM1}||\text{ITEM2}|}{N}$$

We can similarly derive conditional expectation for size change.

Lemma 2. *The probability of an* ITEM *having size k, assuming that each feature is selected independently with probability* $f : \mathbb{N}^+ \to (0, 1]$ *is*

$$P(|\text{ITEM}| = k) = \frac{\binom{N}{k}(1 - f(N))^{N-k} f(N)^k}{Nf(N)}.$$

Lemma 3. *The probability of an intermediate item having size k conditional on sizes of its parents is similarly*

$$P(|A \cap B| = k \mid |A| = a, |B| = b) = \frac{\binom{N}{k}\binom{N-k}{a-k}\binom{N-a}{b-k}}{\binom{N}{a}\binom{N}{b}}.$$

Since all features are equally likely to be selected, conditional probability is independent of scaling function f.

Lemma 4. *By linearity of expectation and assumption that* $|\text{ITEM1}| = a$ *and* $|\text{ITEM2}| = b$ *we can easily derive*

$$\mathbb{E}\left[max\{|\text{ITEM1}|, |\text{ITEM2}|\} - |\text{ITEM1} \cap \text{ITEM2}| \mid |\text{ITEM1}| = a, |\text{ITEM2}| = b\right] = |\text{ITEM1}|\left(1 - \frac{|\text{ITEM2}|}{N}\right).$$

Remark 1. *The assumption that each feature is selected independently with probability* $f(N)$ *should not be confused with each subset of features selected with equal probability. In our case, this is beneficial because smaller sets are intrinsically better, which are easily controlled by the mutation function* f.

Algorithm 2 Randomized crossover with information gain weighting

Require:
First item ITEM1
Second item ITEM2
Scaling function for number of genes ONGENESCALING
Individual feature entropy INFORMATIONENTROPY

```
1:  function CROSSOOVERPAIR
2:      if |ITEM1| < |ITEM2| then
3:          SWAP(ITEM1, ITEM2)
4:      end if
5:      intermediate ← ITEM1 & ITEM2                          ▷ Intersection of genes
6:      rest ← ITEM1 ⊕ ITEM2                  ▷ Bitwise XOR, features present in exactly one item
7:      addNum ← ONGENESCALING(ABS(|ITEM1| - |ITEM2|))
8:                          ▷ Scale the number of new features according to provided function
9:      additional ← RSELECT(addNum, informationEntropy, rest)
10:                     ▷ Randomly select addNum features, weighted by precalculated entropy
11:     mated ← intermediate | additional                     ▷ Add selected features
12:     return mated
13: end function
```

After crossover, the resulting genes initially contain features that were selected in both parents. The intersection of features is a good starting point, as it seems to produce good results during the running of the algorithm. Since the implementation of the genes is a simple bit string, the intermediate result is simply bitwise logical AND of its parents' genes. Another way to combine the parents would be to produce offspring with the union of genes. The latter approach produces offspring that is too large (contains too many selected features) and is thus unsuitable for our objective, where we want to select a sufficiently small feature set. The size of the intermediate result obtained by the intersection (line 5) is smaller than or equal to the size of ITEM2 (smallest of the parents), but can be much smaller, depending on the input sets (see Lemma 3). With the intersection, the number of genes decreases too rapidly (see Lemma 4) and disposes of useful information. We therefore use additional genes presented in exactly one of the parents (generated by element-wise exclusive OR) to construct an additional set of features with good information gain. The visualization for the first part of CROSSOVERPAIR procedure on an example case is shown in Figure 3. The genes of two items to be mated are split into two indexed sets: an intermediate set containing genes from both parents and a set called *rest* containing genes to be used for the enrichment of intersected genes.

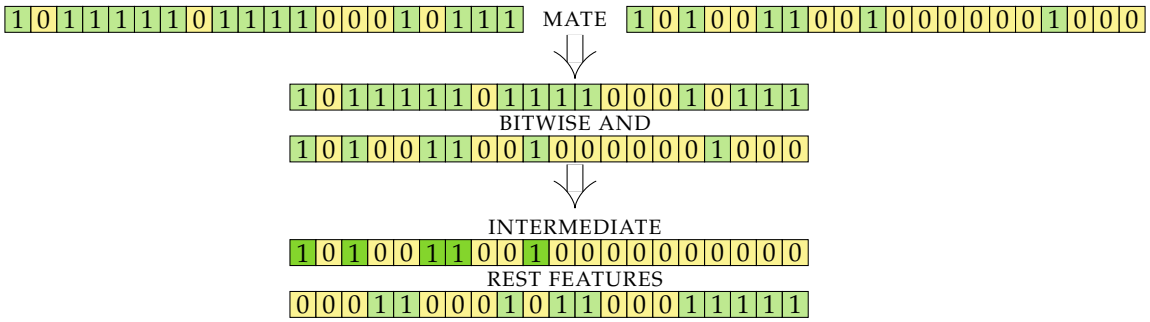


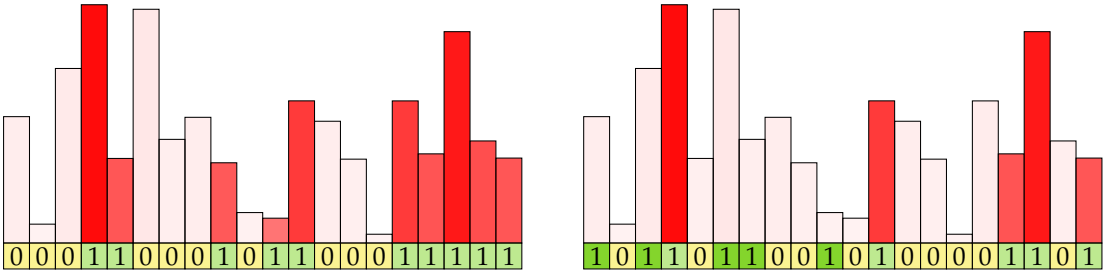
Figure 3. Schema of mating of 2 genes. With bit-wise AND operation we produce the intermediate set and with bit-wise exclusive or XOR we produce the rest (features) set.

Enrichment of the intermediate set of genes is done as follows. First, the number of enrichment genes is calculated using the ONGENESCALING function. This function is used to control the size of the

final offspring. From Lemma 4, it follows that the expected size of the rest (features) set, with the size of the largest parent being constant, is linear to the size of the smaller parent (ITEM2). Adding all genes from the rest set will result in an offspring too large (and diverge toward the whole feature space being selected), selecting none of the genes will diverge towards very small sets. In our experiments, we used the function $x \mapsto \left\lfloor \frac{x}{2} \right\rfloor + 1$ (for ONGENESCALING) that scaled the amount of features appropriately for our data set. For a larger number of features, the functions $\sqrt{x}$ and $\log_a x$ with a sufficiently small base a are a good choice. As with most parameters, ONGENESCALING can be swapped during the run-time and made dependent on the conditions of front updates and statistics.

The second part of the mating algorithm selects the best performing features using heuristics based on information gain. From all features in the rest set, *addNum* are randomly selected (using a weighted random selection, where the weight of each feature is its individual precalculated information gain). The final result is an item with genes from the intersection set and selected genes from rest set. In the implementation, the results are simply combined with element-wise OR. The heuristics for the information gain in our algorithm is based on mutual information gain. The `scikit-learn` implementation [29] of mutual information gain is used for each feature in the train set.

Figure 4a shows the continuation of the example mating algorithm from the previous figure. Each of the initial features is represented with its information gain, which is shown by the height and intensity of the shade of the column above it. Columns for features in the rest set are shown in red (and the corresponding genes are green in the bottom row). Of the eight features in the rest features row, five are selected using weighted random function. Figure 4b shows the visualization of the result (offspring) of the mating procedure. The genes selected by intersection are marked with dark green shading, while the genes selected by information enrichment are shown in light green. The corresponding information gain statistics in shown with column color intensity. In this example, two genes of size 14 and 6 were combined and they produced a result with size 10, which may be further increased by the mutation strategy. The resulting item contains genes that are present in both parents and were therefore rated as good in the previous evaluation. It is enriched with a limited number of genes that are present in either parent, weighted according to their information gain.



(**a**) Rest genes with feature importances (**b**) Final crossover results

Figure 4. Visualization of the rest set and final mating result. Information gain of a particular feature is depicted by the height of the column above a particular feature.

3.4. Purging Features from Pareto Front Using Information Gain

An initial version of the algorithm presented in Algorithm 1 has been further improved to exploit pre-trained models for the calculation of the information gain and to purge non-relevant features from items on the Pareto front. For each item added to the front, a model trained on its genes is stored. Since the number of elements is small compared to the learning data, storing the models results in virtually no additional effort in run-time and is a useful tool for evaluating the optimization over time. In the same way as when the population is reset to the Pareto front, another predicate is introduced to indicate when the Pareto front should be purged using the information criterion. The gene purging procedure is presented in Algorithm 3.

Algorithm 3 Gene purging algorithm

Require:
Current Pareto front FRONT
Evaluation function with ability to randomly shuffle features EVALF

```
1: function PURGEPARETOFRONT
2:     newFront = ← Dict.empty                ▷ Empty dictionary representing new Pareto Front
3:     for item in front do
4:         if |ITEM| == 1 then
5:             continue                       ▷ Items with only one feature cannot be reduced
6:         end if
7:         baseResult ← item.result           ▷ Score acquired by using all features from item
8:         scoreDecrease ← []                 ▷ Score decrease for each feature
9:         for geneInd in item.genes WHERE ISSET(item.genes[geneInd]) do  ▷ Check only set genes
10:            newScore ← EVALF(item.model, item.genes, geneInd)
   ▷ Evaluate existing trained model on test data set, where values of feature geneInd are randomly
   shuffled
11:            scoreDecrease[geneInd] ← baseResult - newScore
12:        end for
13:        bestInd ← ARGMIN(scoreDecrease)
                   ▷ Take the gene which showed smallest score decrease when shuffled
14:        newItem ← ITEM(genes=UNSETBIT(item.genes, bestInd))
                   ▷ Create new item where the gene with the smallest score decrease is unset
15:        newFront[|newItem|] ← EVALUATE(newItem, EVALF)
                   ▷ Evaluate newly created item (train and test model)
16:    end for
17:    front ← front ∪ newFront               ▷ Update current front with newly calculated items
18:    front ← REMOVEDOMINATED(front)         ▷ Remove unoptimal items from Pareto front
19:    return front
20: end function
```

The main idea of purging the items on the Pareto front with information criterion is to discard unneeded features from items that already perform well. For each item on the Pareto front, we try to discard a feature that makes the least contribution to the item's score. The obvious way to do this would be to transverse through all the features used in the prediction model, remove superfluous features one by one, and train and re-evaluate such a model. For each item, a new model must be trained and re-evaluated. This is time-consuming and offers only minor improvements. Our proposal is to use a heuristic approach to select a feature to be removed. For each such feature, an existing model corresponding to the item is used (saving time by avoiding the training of a new model) and evaluated according to the previously used test data set. For each selected feature, its values are randomly shuffled across the test data set. The motivation for such heuristics is that the shuffling values of a feature with strong predictive power has a much stronger effect on the resulting score than shuffling values of a feature with only low predictive power. In this way, we can provide heuristics for estimating the predictive power of features in a model-agnostic way using only a linear number of model calls. Such heuristics provide a noticeable acceleration and can be used for any type of model. The approach targets black box models with a high ratio between training and inference time/resource consumption—e.g., gradient boosting or complex neural networks. For each newly shuffled feature in an item, a change in the base score is recorded (line 11). Interestingly, sometimes the change can even be negative if the included features are detrimental to the overall prediction procedure. This is proving to be an effective method for quickly sorting out bad features introduced by mutations at the end of simulations when the exploration is limited to the feature space previously explored.

The feature whose shuffling has the most detrimental effect is taken and a new item is constructed whose genes are the same as those of the original element, except that the resulting feature is omitted (lines 13 and 14). This procedure can be further generalized. For example, first *scoreDecrease* could be scaled with some function (for example, x^3) and then the feature to be removed could be sampled by these new *scoreDecrease* weights from the set. Special care should be taken that the scaling function is monotonous (and preferably a bijection) since negative values can occur and should not be bundled with positive ones. If a random selection is used, the final probabilities must also be re-scaled correctly due to possible negative weights. After the feature to be removed is selected, the new item is evaluated and all new items are brought to the front. The front should also be purged to take into account possible new features on the front.

The most important heuristic optimization used by the front purging functions is based on the fact that only a new model is built with the features with the most promising importance score (as inferred by a model when feature values are shuffled). The time required for this operation is linear in the number of features. For each feature, the model requires one evaluation (inference) on the training set. The implementation of the gene purging can be further optimized so that each feature is selected only once. Under the reasonable assumption of sufficiently deterministic training and evaluation, it is easy to see that evaluating an item and (potential) addition of a feature to the front is an idempotent operation. Removing the same feature from the same item on the front several times is not a reasonable change and therefore another feature could be considered for removal. An appropriate threshold for reducing the score may also be introduced so that features with very high predictive power are never removed.

4. Data

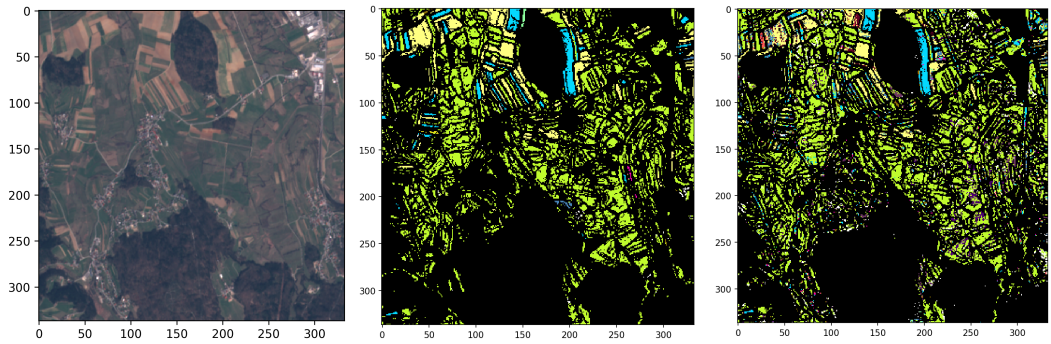
FASTENER is focused on improving the state-of-the-art in EO land-cover classification. The description of the EO data set is given in Sections 4.1 and 4.2. Apart from that, the algorithm was tested on 25 additional feature selection benchmark data sets with a varying number of features, label classes and instances, in order to be compared against the existing state-of-the-art methods.

4.1. EO Data

Earth Observation data were provided by the EU Copernicus program's Sentinel-2 mission, whose main objectives are land observation, land use and change detection, support for generating land cover, disaster relief support and climate change monitoring [30]. The data comprise 13 multi-spectral channels in the visible/near-infrared (VNIR) and short wave infrared (SWIR) spectral range with a temporal resolution of 5 days and spatial resolutions of 10 m, 20 m and 60 m (the latter is used for diagnostics only) [3]. Sentinel's Level-2A products (surface reflectances in cartographic geometry) were retrieved via the SentinelHub (https://www.sentinel-hub.com/) services and processed using the `eo-learn` (https://github.com/sentinel-hub/eo-learn) library. In addition, a digital elevation model for Slovenia (EU-DEM) with 30 m resolution was used.

The ground truth data in our experiments were collected from the Slovenian land parcel identification system (LPIS). The original LPIS data consist of 177 different vegetation classes. These classes were grouped into 23 more general classes proposed by domain experts. The final data set includes 23 separate classes describing the type of farmland and one class describing all non-agricultural surfaces. Data have been collected for the year 2017.

A classification scenario is depicted in Figure 5. The figure shows a subset (true colour) of input EO data, the manually acquired ground truth data and the final automatic classification result of our algorithm. Based on the (easily obtainable EO data), the classification algorithm predicts ground-truth label (which is difficult to obtain and often contains incorrect data). When comparing Figure 5b,c, we observe the similarity between ground truth and the classification result.



(**a**) EO true color (**b**) Ground truth—LPIS data (**c**) Classification result

Figure 5. Partial input data, ground truth data and classification results.

4.2. Feature Engineering and Sampling

The EO data were collected for the entire year. Four raw band measurements (red, green, blue—RGB and near-infrared —NIR) and six relevant vegetation-related derived indices (normalized differential vegetation index—NDVI, normalized differential water index—NDWI, enhanced vegetation index—EVI, soil-adjusted vegetation index—SAVI, structure intensive pigment index—SIPI and atmospherically resistant vegetation index—ARVI) were considered. The derived indices are based on extensive domain knowledge and are used for assessing vegetation properties. In Figure 6d, an example of NDVI index is depicted, which is an indicator of vegetation health and biomass. Its value changes during the growing season of the plants and differs significantly from other non-planted areas. The NDVI is calculated as:

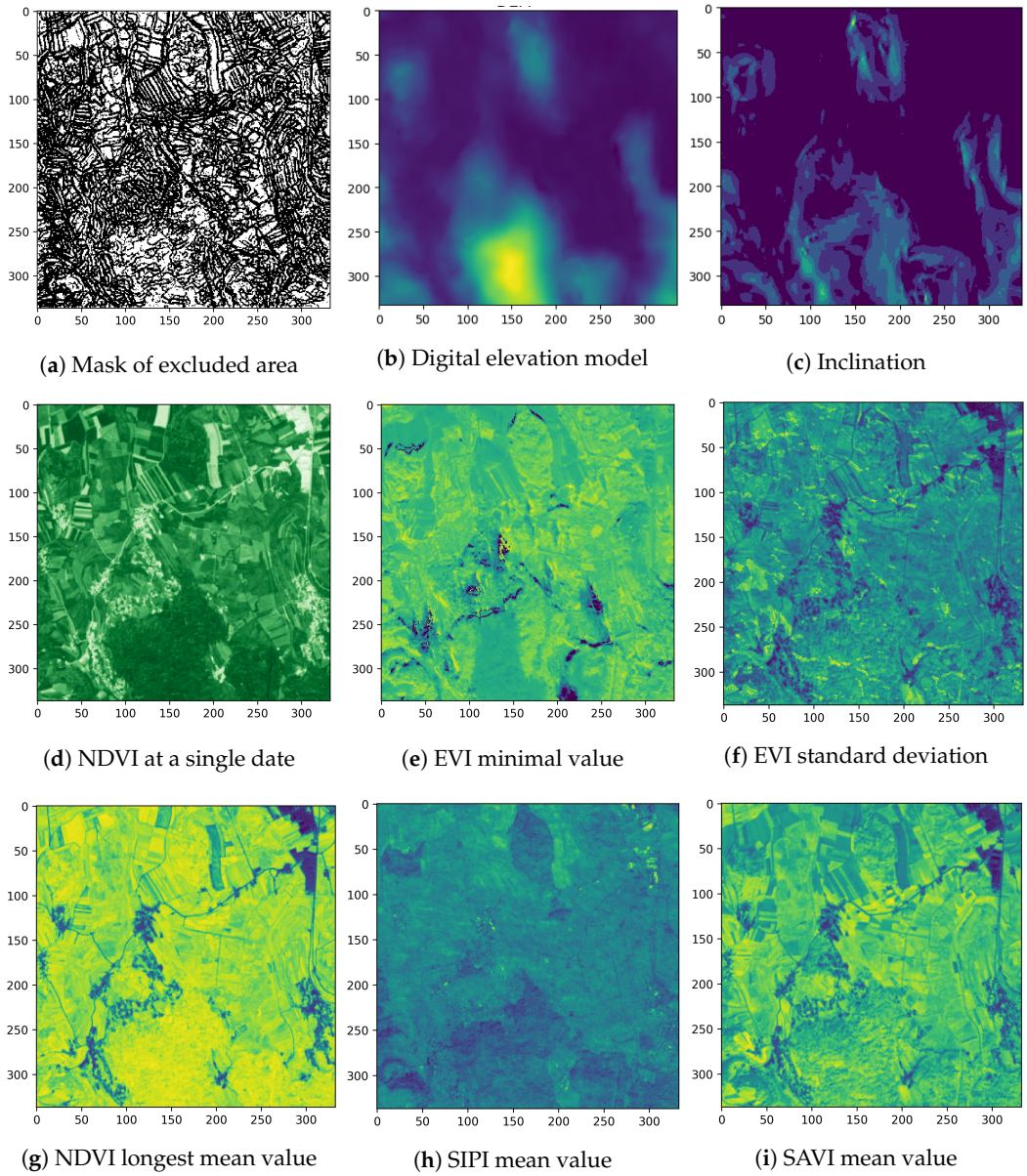
$$NDVI = \frac{NIR - red}{NIR + red}$$

Timeless features were extracted based on Valero et al. [20]. These features can describe the three most important crop stages: the beginning of greenness, the ripening period and the beginning of senescence [20,21]. Annual time series have different shapes due to the phenological cycle of a crop and characterize the evolution of a crop. With timeless features, they can be represented in a condensed form.

For each pixel, 18 features per each of the 10 time-series were generated. The raw value and maximum inclination for a given pixel were calculated from the elevation data as 2 additional features. In total, 182 features were used in the experiments.

The examples of learning features are depicted in Figure 6: EVI minimal value (Figure 6e), EVI standard deviation (Figure 6f), NDVI maximum mean value in a sliding temporal neighborhood of size 2 (Figure 6g), SIPI mean value (Figure 6h) and SAVI mean value (Figure 6i).

Prior to the experiments, edge detection [31] was performed on EO data, excluding the pixels at the borders of land plots. These pixels are potential mixed-class instances that can have a negative effect on the learning process. An example of an edge detection mask is depicted in Figure 6a. A balanced learning set was sampled from the entire data set with 20×10^3 data points (pixels) representing each class. The classes with the lowest frequency were oversampled in the vicinity of sampled instances. The entire learning data set [32] consists of 480×10^3 samples.



(**a**) Mask of excluded area (**b**) Digital elevation model (**c**) Inclination

(**d**) NDVI at a single date (**e**) EVI minimal value (**f**) EVI standard deviation

(**g**) NDVI longest mean value (**h**) SIPI mean value (**i**) SAVI mean value

Figure 6. Examples of features derived from EO time series. Each feature represents a potentially significant parameter for land cover classification [20].

4.3. Feature Selection Benchmark Data Sets

Benchmark data sets for feature selection were taken from [6]. The description of all the data sets is given in Table 1. It is apparent from the table that our main data set, EOData [32] in the last row, differs from other data sets with its very high number of instances and relatively low number of features.

Table 1. Description of the data sets used for testing the FASTENER with number of instances, features and classes. $I_{C\downarrow}$ and $I_{C\uparrow}$ represent the percentage of instances representing the smallest and the largest class, respectively.

Data Set	Instances	Features	Classes	$I_{C\downarrow}$	$I_{C\uparrow}$
ALLAML	72	7129	2	35	65
arcene	200	10,000	2	44	56
BASEHOCK	1993	4862	2	50	50
CLL_SUB_111	111	11,340	3	10	46
COIL20	1440	1024	20	5	5
colon	62	2000	2	35	65
gisette	7000	5000	2	50	50
GLIOMA	50	4434	4	14	30
GLI_85	85	22,283	2	30	70
Isolet	1560	617	26	4	4
leukemia	72	7070	2	35	65
lung	203	3312	5	3	68
lung_discrete	73	325	7	7	29
lymphoma	96	4026	9	2	48
nci9	60	9712	9	3	15
ORL	400	1024	40	3	3
orlraws10P	100	10,304	10	10	10
PCMAC	1943	3289	2	50	50
Prostate_GE	102	5966	2	49	51
RELATHE	1427	4322	2	45	54
TOX_171	171	5748	4	23	26
USPS	9298	256	10	8	17
warpAR10P	130	2400	10	10	10
warpPIE10P	210	2420	10	10	10
Yale	165	1024	15	7	7
EOData	480,000	182	24	4	4

5. Results

5.1. Experimental Setup

To assess the quality of the selected feature subsets, a *frontSurface* metric was introduced to measure the surface under the Pareto front produced by the resulting subsets. With the analogy to the *area under a curve* (AUC), which is often used in the evaluation of classification methods, the *frontSurface* measure can also be called the *area under a front* (*AUF*). The area under a front can be interpreted as a measure of the efficiency of the feature selection algorithm. The surface is calculated as a simple sum of the best scores for each number of features k. In order to simplify the presentation and later evaluation, the surface is calculated only for a fixed maximum number of features K and then normalized. This makes it easier to compare fronts of different sizes and keeps them resistant to outliers with a large number of features. The measure is defined as

$$AUF := \frac{\sum_{k=1}^{K} opt(k)}{K} \qquad opt(k) := \max_{|\text{ITEM}|\leq k} \{item.result\},$$

where $opt(k)$ represents the item with the highest score on the Pareto front, with a size of, at most, k (i.e., the best performing subset of features for a fixed number of features). In our case, the maximum number of features $K = 20$ was selected because the selected feature subsets rarely have more than 20 features. Intuitively, the *AUF* measures the average of the best scoring items with a size of less than K.

All experiments on benchmarking data sets were performed with the same setup. The data for feature selection were first split into training and test subset (80:20) with stratified random split (`sklearn.model_selection.train_test_split` with a random state 2020 was used). The selected

algorithm was run on a training subset, where the accuracy score was calculated as the F_1 score of the resulting model on a test set. For the FS-SDS algorithm, which uses internal data set splitting, the full data set was used. EO-data set was further split into two equal parts. The first part was used as a test set in the evaluation phase. The second part was used in the later analysis of the algorithm's generalization in Section 6.2.

The resulting statistics were calculated using the best reported feature subset for a data set and feature selection algorithm. To partially avoid possible feature selection bias [33] (test data set is not separate from the data set used for feature selection), a set of 80:20 training/test balanced splits were done using random seeds from 20 to 50. For wrapping algorithms (which typically use a significant amount of time in the learning phase), it is often impossible to validate according the nested cross-validation loop strategy.

The final results were obtained as the *AUF* of F_1 score, produced by the models trained on training data sets using the selected feature subset and evaluated on corresponding test subsets. The results are used for comparative purposes between the different algorithms.

For FASTENER algorithm, the decision tree classifier provided by the `scikit-learn` library was used as the base classification algorithm (using Gini index for measuring quality of a split and a requirement of keeping, at a minimum, two examples in a split). A random mating pool of size 3 was selected and all item pairs were mated using information gain crossover. All items from the Pareto front were also carried over onto the next generation and mutated along with the newly created ones. For DT-forward, we used the same decision tree hyper parameters as with FASTENER. FS-SDS uses Extreme Learning Machine (ELM) for classification. The latter was configured with 160 hidden neurons and the sigmoid activation function.

5.2. Comparison with Similarity-Based Methods on EO Data

FASTENER results were compared with the results of the KBEST algorithm and the RELIEFF algorithm. Although outdated, they are presented in the literature as the currently best-performing methods in the field of land cover classification [22]. To no surprise, FASTENER achieves better results. Similar to FASTENER, the best feature subset for a fixed number of features was represented as an item in a Pareto front. The Pareto front visualization for reported optimal feature subsets is shown in Figure 7. The optimal feature subsets reported by the FASTENER algorithm outperform KBEST and RELIEFF. Since the tested implementation of RELIEFF uses Python's native KD tree, only 20% of the original training data was used to speed up the feature selection process in RELIEFF and compared with the subset selected by FASTENER on the same reduced training data.

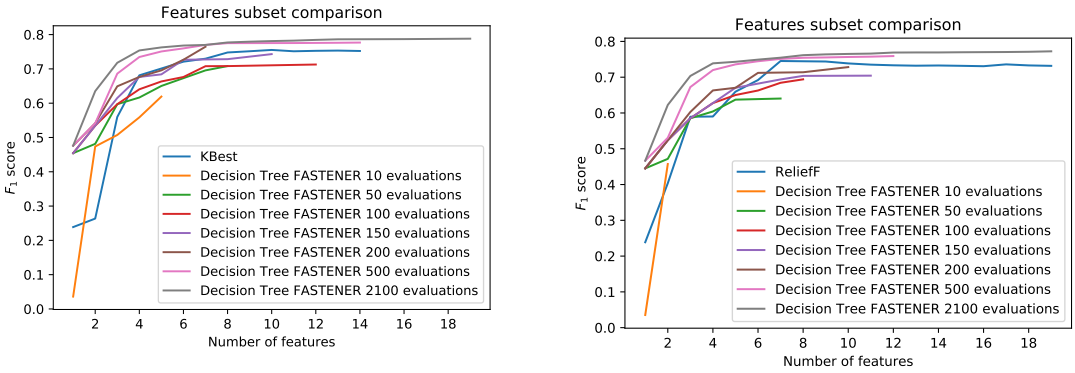
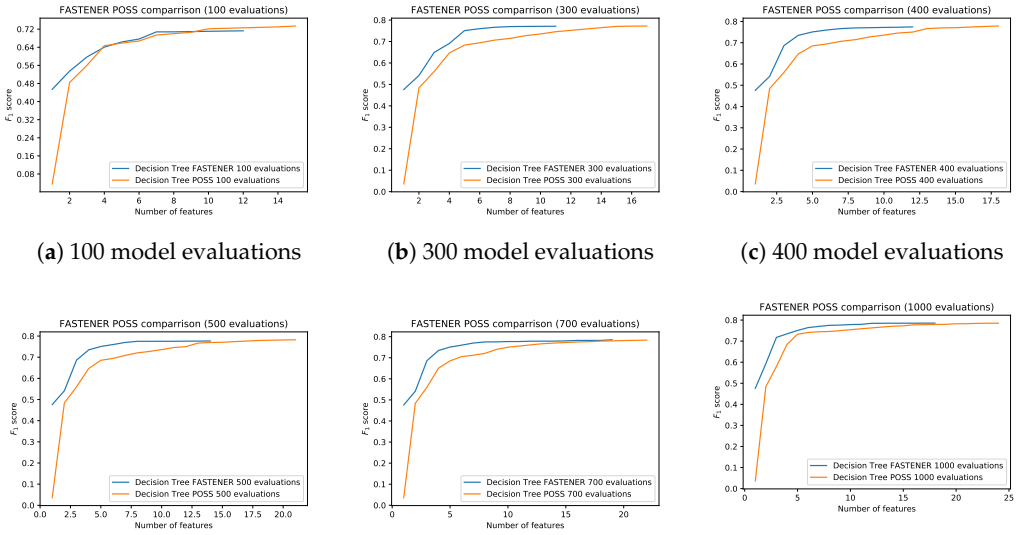


Figure 7. Pareto front comparison.

5.3. Detailed Comparison with Wrapper Methods for EO Validation

Due to the nature of the EO data set (high number of instances), the evaluation of DT-forward, DT-backward, SVM-forward and SVM-backward [14] was not possible due to extremely long computation times. Along with FASTENER and POSS, we were also able to test the EO data set using FS-SDS [12]. The following evaluation presents a detailed comparison of POSS and FASTENER algorithms, and FS-SDS results are given at the end of the subsection.

Figure 8 shows the progression of the Pareto front produced by POSS and FASTENER algorithms with the same number of model evaluations. It can be clearly seen that the convergence speed of the FASTENER algorithm outperforms the convergence speed of a comparable POSS implementation. Further analysis shows that the FASTENER algorithm converges about three-times faster than the POSS algorithm.

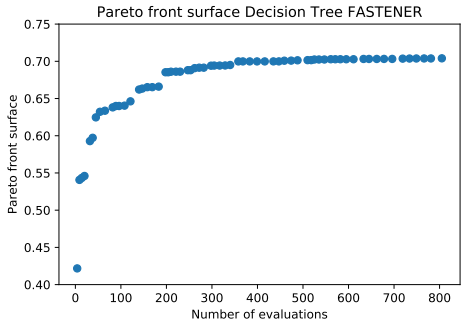


(**a**) 100 model evaluations (**b**) 300 model evaluations (**c**) 400 model evaluations

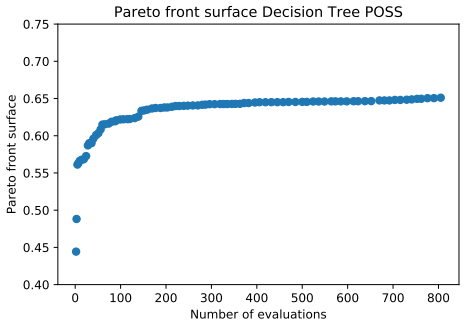
(**d**) 500 model evaluations (**e**) 700 model evaluations (**f**) Pareto front after 1000 model evaluations

Figure 8. Comparison of the Pareto fronts produced by FASTENER and POSS algorithms after different numbers of iterations.

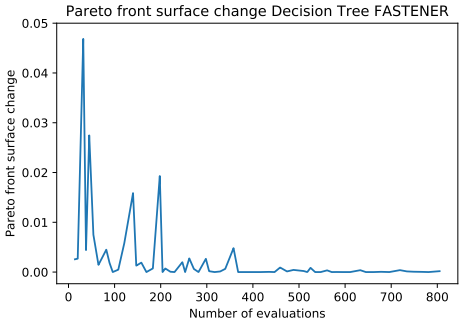
Comparisons of Pareto *AUF* and *AUF* changes for POSS and FASTENER algorithms during the first 800 iterations are shown in Figure 9. A quick visual comparison of the *AUF* graphs between POSS and the FASTENER results shows that the surface with the FASTENER algorithm is on average of 5% larger. The FASTENER *AUF* also exhibits much larger jumps. Larger jumps indicate a strong improvement in the F_1 score. This can either be a large improvement in the score for an existing k number of features or a slightly smaller improvement in the score for many smaller feature sets that dominate a large part of an existing Pareto front. Periodic larger jumps correspond to the invocation of PURGEPARETOFRONT routine, which tries to improve the results with additional information gain.



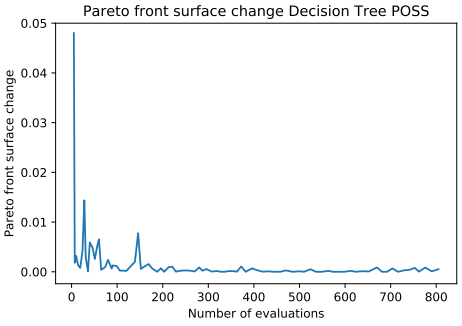
(**a**) Progression of *AUF* with FASTENER algorithm



(**b**) Progression of *AUF* with POSS algorithm



(**c**) *AUF* change with FASTENER algorithm



(**d**) *AUF* change with POSS algorithm

Figure 9. The comparison of the Pareto fronts generated by FASTENER and POSS algorithms after a different number of iterations. FASTENER exhibits better *AUF* scores and the discovery of several jumps, indicating the discovery of a distinct new best combination of features.

Another useful tool in convergence analysis is to compare the change in *AUF* presented in Figure 9c,d. The changes in the FASTENER algorithm are an indication of a higher convergence ratio. The computational complexity of *AUF* is minimal (linear in the size of the front) and can be easily computed while purging the Pareto front after each generation. The continuous calculation of the *AUF* (and its change) can be used as a helpful marker to measure the course of convergence in several of the above methods. In particular, the change in the area under a front can be used to modify the probabilities in the crossover and mutate methods to control the feature space search.

Detailed analysis of wrapper methods tested on EO-data is presented in Table 2. All the tested methods produce the area under a front with low variation, which can be attributed to a data set size. It can clearly be seen that FASTENER outperforms both POSS and FS-SDS.

Table 2. Results of wrapper feature selection algorithms on EO data set. FASTENER yields the highest average *AUF* and lowest number of model evaluations. Standard deviation of the *AUF* is small for all methods. Best values accross different methods are bolded (in all tables).

Method	avg. *AUF*	sd	eval_n
POSS	0.66	0.0008	**1000**
FS-SDS	0.62	0.0008	120,000
FASTENER	**0.71**	0.0008	**1000**

5.4. Benchmark Data Set with Wrapper Methods

Along with FASTENER, forward decision tree selection method (DT-forward) [14] and FS-SDS algorithm [12] were tested for comparison on an open feature selection data sets [6] used for benchmarking.

Tables 3 and 4 present comparisons of the FS-SDS algorithm, FASTENER and forward selection with decision tree (DT-forward). For each of the algorithms, 10 different data splits were used for feature selection and each feature subset was tested on 30 different test/train data splits. FASTENER was run with an exploration setting (larger generations) for 200 generations and the number of evaluations is a lot higher than in the EO setting. Each resulting Pareto front was purged, since adding more features might hurt performance on new data and the area under the Pareto front was calculated. Tables 3 and 4 present mean, max, median and standard deviation of thus obtained surfaces under Pareto fronts, along with the number of model evaluations.

Table 3. Mean, max, median value, standard deviation *AUF* and number of model evaluations. FS-SDS uses 30,000 model evaluations per each *k* (whereas for FASTENER the number of iterations produces the whole Pareto front) in the first iteration. Number of evaluations is reduced with the next iterations and stays in the range between (10,000–30,000).

	FS-SDS					FASTENER				
Data Set	**Mean**	**Max**	**Medi**	**sd**	**eval_n***	**Mean**	**Max**	**Medi**	**sd**	**eval_n**
ALLAML	0.643	0.717	0.65	0.039	30,000	**0.776**	**0.894**	**0.78**	0.062	11,238
arcene	0.608	0.673	0.607	0.037	30,000	**0.707**	**0.785**	**0.713**	0.034	12,083
BASEHOCK	0.58	0.618	0.58	0.018	30,000	**0.742**	**0.78**	**0.744**	0.019	13,847
CLL_SUB_111	0.537	0.671	0.534	0.053	30,000	**0.626**	**0.79**	**0.623**	0.067	11,911
COIL20	0.655	0.673	0.657	0.01	30,000	**0.687**	**0.714**	**0.687**	0.012	14,214
colon	0.658	0.77	0.652	0.069	30,000	**0.763**	**0.9**	**0.757**	0.076	11,256
gisette	0.672	0.682	0.671	0.005	30,000	**0.786**	**0.802**	**0.786**	0.006	13,213
GLIOMA	0.549	0.687	0.561	0.067	30,000	**0.618**	**0.827**	**0.617**	0.095	11,674
GLI_85	0.659	0.746	0.665	0.063	30,000	**0.755**	**0.894**	**0.77**	0.065	11,238
Isolet	0.365	0.392	0.364	0.012	30,000	**0.396**	**0.432**	**0.395**	0.012	14,572
leukemia	0.62	0.77	0.618	0.06	30,000	**0.824**	**0.9**	**0.833**	0.057	11,248
lung	0.663	0.75	0.656	0.031	30,000	**0.727**	**0.828**	**0.731**	0.038	11,802
lung_discrete	**0.533**	0.666	**0.532**	0.068	30,000	0.511	**0.716**	0.516	0.086	11,753
lymphoma	0.539	0.619	0.548	0.058	30,000	**0.59**	**0.757**	**0.58**	0.07	11,719
nci9	0.331	0.513	0.311	0.097	30,000	**0.405**	**0.61**	**0.406**	0.088	12,081
ORL	0.347	0.395	0.351	0.029	30,000	**0.39**	**0.488**	**0.389**	0.04	13,540
orlraws10P	0.675	0.77	0.686	0.053	30,000	**0.694**	**0.821**	**0.695**	0.062	13,571
PCMAC	0.596	0.624	0.594	0.012	30,000	**0.71**	**0.736**	**0.708**	0.013	15,768
Prostate_GE	0.676	0.734	0.691	0.041	30,000	**0.748**	**0.837**	**0.749**	0.05	14,002
RELATHE	0.52	0.554	0.519	0.017	30,000	**0.667**	**0.707**	**0.667**	0.018	15,541
TOX_171	0.421	0.509	0.427	0.046	30,000	**0.553**	**0.647**	**0.55**	0.039	14,759
USPS	0.561	0.569	0.561	0.005	30,000	**0.583**	**0.593**	**0.583**	0.005	16,670
warpAR10P	0.502	0.588	**0.525**	0.052	30,000	**0.523**	**0.672**	0.518	0.065	13,840
warpPIE10P	0.582	0.646	0.589	0.037	30,000	**0.633**	**0.741**	**0.634**	0.041	13,630
Yale	0.38	0.454	0.376	0.035	30,000	**0.436**	**0.594**	**0.438**	0.052	15,307

Table 3 shows that FASTENER outperforms FS-SDS on almost all data sets, except on the lung_discrete data set, where the mean and median *AUF* score are better with FS-SDS. Apart from outperforming score-wise, FASTENER also produces results faster (with fewer model evaluations) due to additional statistical data, while producing only marginal overhead due to initial entropy calculation.

Table 4. Mean, max, median value, standard deviation of *AUF* and number of model evaluations.

	DT Forward					FASTENER				
Data Set	**Mean**	**Max**	**Medi**	**sd**	**eval_n**	**Mean**	**Max**	**Medi**	**sd**	**eval_n**
ALLAML	**0.826**	0.893	**0.833**	0.044	106,830	0.776	**0.894**	0.78	0.062	11,238
arcene	0.659	0.76	0.665	0.054	149,895	**0.707**	**0.785**	**0.713**	0.034	12,083
BASEHOCK	**0.753**	**0.781**	**0.751**	0.015	72,825	0.742	0.78	0.744	0.019	13,847
CLL_SUB_111	0.54	0.655	0.54	0.049	169,995	**0.626**	**0.79**	**0.623**	0.067	11,911
COIL20	**0.707**	**0.729**	**0.706**	0.011	15,255	0.687	0.714	0.687	0.012	14,214
colon	0.751	0.888	0.745	0.066	29,895	**0.763**	**0.9**	**0.757**	0.076	11,256
gisette	**0.8**	**0.815**	**0.799**	0.007	74,895	0.786	0.802	0.786	0.006	13,213
GLIOMA	0.592	0.743	**0.626**	0.103	66,405	**0.618**	**0.827**	0.617	0.095	11,674
GLI_85	0.724	0.82	0.733	0.066	334,140	**0.755**	**0.894**	**0.77**	0.065	11,238
Isolet	**0.409**	**0.445**	**0.408**	0.014	9150	0.396	0.432	0.395	0.012	14,572
leukemia	**0.866**	**0.9**	**0.873**	0.03	105,945	0.824	0.9	0.833	0.057	11,248
lung	**0.76**	0.827	**0.761**	0.038	49,575	0.727	**0.828**	0.731	0.038	11,802
lung_discrete	**0.526**	0.689	**0.532**	0.08	4770	0.511	**0.716**	0.516	0.086	11,753
lymphoma	0.586	**0.785**	**0.581**	0.077	60,285	**0.59**	0.757	0.58	0.07	11,719
nci9	**0.419**	**0.64**	**0.414**	0.115	145,575	0.405	0.61	0.406	0.088	12,081
ORL	**0.407**	0.464	**0.41**	0.032	15,255	0.39	**0.488**	0.389	0.04	13,540
orlraws10P	**0.76**	**0.835**	**0.765**	0.055	154,455	0.694	0.821	0.695	0.062	13,571
PCMAC	**0.722**	**0.75**	**0.721**	0.013	49,230	0.71	0.736	0.708	0.013	15,768
Prostate_GE	**0.787**	**0.851**	**0.793**	0.041	89,385	0.748	0.837	0.749	0.05	14,002
RELATHE	**0.684**	**0.724**	**0.681**	0.016	64,725	0.667	0.707	0.667	0.018	15,541
TOX_171	0.469	0.536	0.466	0.035	86,115	**0.553**	**0.647**	**0.55**	0.039	14,759
USPS	0.564	0.574	0.565	0.005	3735	**0.583**	**0.593**	**0.583**	0.005	16,670
warpAR10P	**0.533**	0.659	**0.527**	0.06	35,895	0.523	**0.672**	0.518	0.065	13,840
warpPIE10P	**0.669**	**0.753**	**0.676**	0.033	36,195	0.633	0.741	0.634	0.041	13,630
Yale	0.407	0.505	0.406	0.048	15,255	**0.436**	**0.594**	**0.438**	0.052	15,307

Table 4 presents a comparison of features selected by FASTENER with features selected by forward selection with decision tree (DT-forward). DT-forward works by successively adding the best performing features, which can be suitable for small data sets, but the computational complexity explodes, as the number of features increases. As can be seen from Table 4 that FASTENER outperforms the DT-forward in a slightly less than half of the data sets. The difference between the maximum in DT-forward selection and FASTENER data in most of other cases is a few percentage points. The important improvement brought in by FASTENER is the efficiency of obtaining better (or just marginally worse) *AUF* with a lot fewer evaluations in comparisons to the forward selection. On average, the FASTENER uses between 2–5-times fewer model evaluations, depending on the number of features in the data set, while obtaining comparable or sometimes even better results. Additional insight presented by the experiment is the standard deviation when using different data splits. Standard deviation in smaller data sets is larger than in EO-data set, and a bit larger than the standard deviation using DT-forward since FASTENER includes additional random part during the algorithm run, while forward feature selection is deterministic.

Another FASTENER strength compared to the FS-SDS or DT-forward is the non-parametricity concerning the number of features. Both FS-SDS and DT-forward are rigid when selecting the feature subset size and increasing (in the case of forward selection) or changing (FS-SDS) subset size is computationally expensive. FASTENER automatically explores available search space unconstrained by the number of features, but they can be additionally constrained when reporting the results.

6. Discussion

6.1. Comparisons with other Methods

FASTENER builds on the idea of having multiple non-optimal items simultaneously considered for selection. The front in the main loop of the algorithm is a list of items for a fixed number of selected

features. The additional parameter controls the size of the buckets, but initial testing concluded that increasing the bucket size to more than 2 decreased the performance and a more elitist parent selection (bucket sizes 1 and 2) was used. The exploration phase, where the population is allowed to deviate from the currently optimal front and only the items in the currently selected population are used as possible parents for a few rounds can be seen as a generalization of multiple fronts used in NSGA-II. The front purging after a selected number of rounds (controlled by a parameter) then returns the front to the current best and maintains an elitist gene selection.

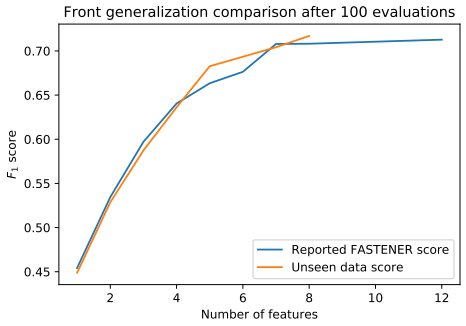
Due to a large number of features and data points in the EO domain, smaller population sizes are more suitable for feature selection. In this case, the increase in bucket sizes (and non-optimal fronts) greatly increases the population size, which produces slower convergence as the elitism part does not kick in quickly enough.

Another specific thing about feature selection is the fact that the feature subset size dimension is discrete and mostly fully dense. Since the optimal sizes of subsets quickly converge even for a small number of selected features, the clustering optimization performed by NSGAA-II and RPSGAe does not improve the efficiency of convergence.

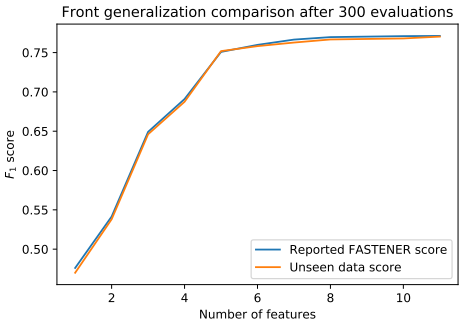
6.2. Generalization

An important aspect of feature subset optimization is further generalization to unseen data. As mentioned at the beginning of this section, an additional part of the data was kept unseen to the FASTENER algorithm during the phase of selecting the optimal subset of features. This hidden data set was used to analyze the overfitting of the FASTENER algorithm to the combination of training/test data. For each generation of the FASTENER algorithm, the features from the optimal items were used as features to train the model using the training data. However, this time the models were evaluated using the 10% of data not seen during FASTENER iterations. The overfitting was minimal and ranged from 0.5 to approximately 0.3 percentage points compared to the results on the test data set reported by the FASTENER algorithm. An important piece of information obtained from additional testing on the unseen data was the fact that some items from the reported optimal Pareto fronts were strictly dominated by feature subsets with fewer features. This was not surprising for subsets with a larger number of features, as the Pareto front "levels off" after a larger number of features. Results of the generalization analysis are presented in Figure 10.

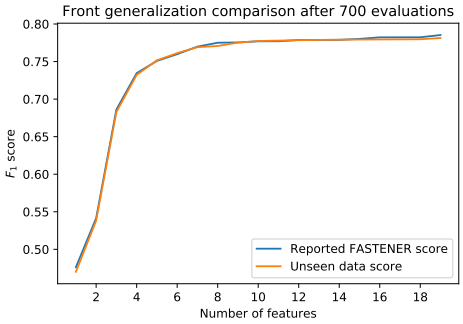
The generalization on unseen data is quite good, which is at least partly due to a large data set compared to the number of algorithm iterations. The algorithm did not converge to some local optimum imposed by training data. Another way to analyze training performance on unseen data is to check the difference between *AUF* values on the reported front and unseen data. The plot of the *AUF* difference is shown in Figure 11. The difference levels off after approximately 60 iterations of the algorithm and appears to be constant with some variation.



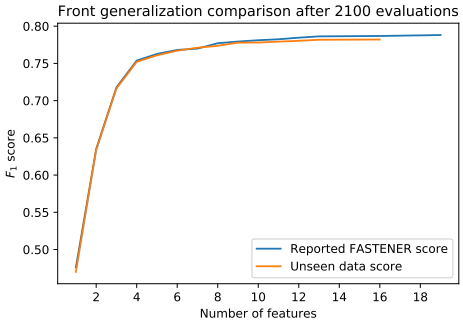
(**a**) Pareto front after 100 model evaluations



(**b**) Pareto front after 300 model evaluations



(**c**) Pareto front after 700 model evaluations



(**d**) Pareto front after 2100 model evaluations

Figure 10. The generalization of results on unseen data presents small performance discrepancies between test data and previously unseen data.

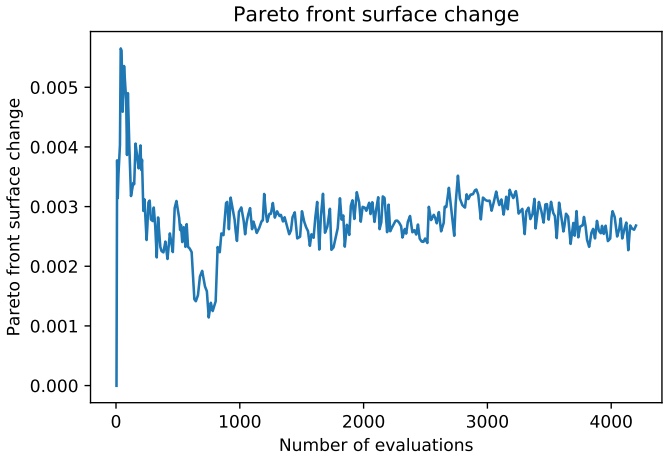
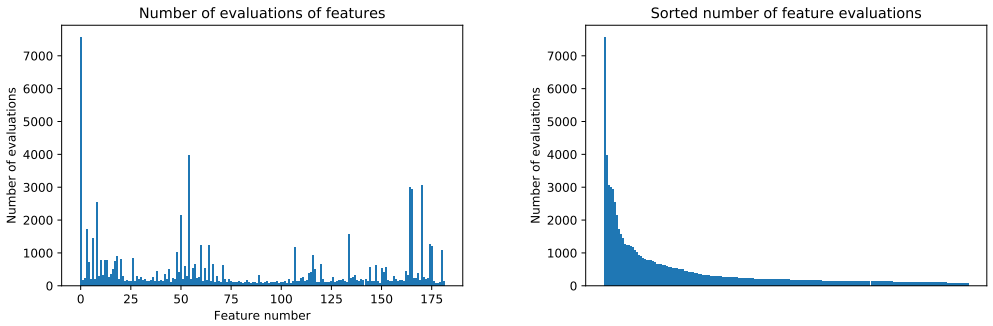


Figure 11. Low *AUF* difference between the test and unseen data shows good generalization abilities of the FASTENER algorithm.

Apart from the generalization, it is interesting to observe the statistics for selected items and the statistics of individual genes during the execution of the algorithm. Figure 12 visualizes optimal selected features after the termination of the FASTENER algorithm. The yellow squares represent selected optimal features as genes. With y axis progression, new features are added. Interestingly, the algorithm does not simply extend the feature sets of k for the higher k's, but rather examines previously excluded feature combinations that, with additional features, perform better.

Figure 12. Visualization of optimal features for different feature subset sizes. The x axis represents the feature index, while the y axis depicts the number of features k (increasing with each row). FASTENER does not simply add new features as k is increased, but rather finds the best possible combination of features that gives the best possible classification result for a given k.

Apart from selecting the optimal subsets, one should also look at the density of the different features, as they were involved during the iterations. The diagrams in Figure 13 show the number of evaluations of features during the run time of the algorithm. If the feature subset [1, 5, 7, 100] was evaluated, each of the features shown is considered to be evaluated once. Figure 13a shows the number of feature evaluations indexed by feature indices. The most evaluated feature is 0, since this was the only item in the starting Pareto front for the algorithm. Looking at other features, there are some areas of higher activity, but interestingly, while features between indices 50 and 75 seem to be fairly highly valued, they are very rarely included in the final optimal Pareto fronts. It seems that they are favored by heuristics. On the contrary, towards the end, features are fairly highly valued and are often present in optimal subsets.



(**a**) Number of feature evaluations (**b**) Sorted number of feature evaluations

Figure 13. Number of feature evaluations. Although some features are not represented in Figure 12, they have often been evaluated by the algorithm.

7. Conclusions and Future Work

With the FASTENER algorithm, we have combined a wrapping methodology with information-theoretical measures based on entropy. The resulting feature selection method has proven to be very promising (superior in EO scenario), with its fast convergence (less learning iterations needed) and better accuracy (higher F_1 score and surface under the Pareto front) than the currently tested methods in the field. In the EO literature, feature selection methodologies have not been thoroughly explored and thus very basic algorithms, such as ReliefF and POSS have been reported to give the best results so far. We have also repeated experiments using FS-SDS, which FASTENER consistently outperformed. FASTENER was compared to FS-SDS and DT-forward algorithms on 25 open feature selection data sets, which are significantly different from the EO data set (a few instances and a lot of features). In terms of accuracy, FASTENER is comparable with DT-forward but generally

achieves the same result with far fewer evaluations. Although the method was originally developed for applications within the EO scenarios, its usage in other domains seems promising. Any supervised machine learning problem (classification or regression) that requires optimization of the accuracy measure (either F_1 score or *RMSE*) with respect to the number of used features would benefit from the implementation of FASTENER.

Several aspects will be addressed in future work. Better theoretical justification of the algorithm is needed as well as the analysis of its convergence and other relevant properties. Possibilities of parallelization of FASTENER (similar as in [34] for POSS) should be examined to speed up the convergence.

Finally, for EO or similar problems, where the calculation of features themselves is computationally challenging, optimization of the final time of the learning process (including feature engineering and data acquisition) need to be performed.

Author Contributions: Conceptualization and methodology, F.K. and K.K.; software, F.K., B.Š.; validation, F.K. and K.K.; writing and editing, K.K., F.K. and B.Š.; visualization, B.Š., F.K.; supervision, project administration, and funding acquisition, K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by European Union's Horizon 2020 programme project EnviroLENS (Innovation Action) grant number 821918 and project PerceptiveSentinel (Research and Innovation) grant number 776115.

Acknowledgments: The authors would like to thank the H2020 Perceptive Sentinel project team members from JSI and Sinergise, who have created an effective and easy to use earth observation Python library `eo-learn` which is based on SentinelHub data access.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ARVI	Atmoshperically adjusted vegetation index
ASTFS	Automatic spectro-temporal feature selection
AUF	Area Under (Pareto) Front
DT	Decision Tree
EO	Earth Observation
ESA	European Space Agency
EVI	Enhanced Vegetation Index
FASTENER	Feature Selection Enabled by Entropy
LDA	Latent Dirichlet Allocation
LPIS	Land Parcel Identification System
NIR	Near Infra Red
NDVI	Normalized differential vegetation index
NDWI	Normalized differential water index
NLPCA	Non-linear Principal Components Analysis
PB	Petabyte
PCA	Principal Components Analysis
POSS	Pareto Optimization for Subset Selection
PPOSS	Parallel Pareto Optimization for Subset Selection
SAVI	Soil-adjusted Vegetation Index
FS-SDS	Feature Selection using Stohastic Diffucion Search
SIPI	Structure Insensitive Pigment Index

References

1. European Space Agency. Mission Status Report 158. Available online: https://sentinel.esa.int/documents/247904/4114743/Sentinel-2-Mission-Status-Report-158-25-Jan-3-Apr-2020.pdf (accessed on 15 May 2020).
2. Kenda, K.; Kažič, B.; Novak, E.; Mladenić, D. Streaming Data Fusion for the Internet of Things. *Sensors* **2019**, *19*, 1955. [PubMed]
3. Koprivec, F.; Čerin, M.; Kenda, K. Crop Classification using Perceptive Sentinel. In Proceedings of the 21th International Multiconference, Ljubljana, Slovenia, 24–28 September 2018; Volume C, pp. 37–40.
4. Koprivec, F.; Peternelj, J.; Kenda, K. Feature Selection in Land-Cover Classification Using EO-learn. In Proceedings of the 22nd International Multiconference, Ljubljana, Slovenia, 7–11 October 2019; Volume C, pp. 37–50.
5. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2001; Volume 16.
6. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R.P.; Tang, J.; Liu, H. Feature selection: A data perspective. *ACM Comput. Surv. (CSUR)* **2018**, *50*, 94. [CrossRef]
7. Khan, A.; Baig, A.R. Multi-objective feature subset selection using non-dominated sorting genetic algorithm. *J. Appl. Res. Technol.* **2015**, *13*, 145–159. [CrossRef]
8. Gaspar-Cunha, A.; Covas, J.A. RPSGAe—reduced Pareto set genetic algorithm: Application to polymer extrusion. In *Metaheuristics for Multiobjective Optimisation*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 221–249.
9. Gaspar-Cunha, A. Feature selection using multi-objective evolutionary algorithms: Application to cardiac SPECT diagnosis. In *Advances in Bioinformatics*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 85–92.
10. Spolaôr, N.; Lorena, A.C.; Diana Lee, H. Feature selection via pareto multi-objective genetic algorithms. *Appl. Artif. Intell.* **2017**, *31*, 764–791. [CrossRef]
11. Qian, C.; Yu, Y.; Zhou, Z.H. Subset Selection by Pareto Optimization. In *Advances in Neural Information Processing Systems 28*; Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; pp. 1774–1782.
12. Alhakbani, H.; al Rifaie, M.M. Feature Selection Using Stochastic Diffusion Search. In Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017; pp. 385–392. [CrossRef]
13. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]
14. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
15. Yin, L.; You, N.; Zhang, G.; Huang, J.; Dong, J. Optimizing Feature Selection of Individual Crop Types for Improved Crop Mapping. *Remote Sens.* **2020**, *12*, 162. [CrossRef]
16. Somers, B.; Asner, G.P. Multi-temporal hyperspectral mixture analysis and feature selection for invasive species mapping in rainforests. *Remote Sens. Environ.* **2013**, *136*, 14–27. [CrossRef]
17. Stromann, O.; Nascetti, A.; Yousif, O.; Ban, Y. Dimensionality Reduction and Feature Selection for Object-Based Land Cover Classification based on Sentinel-1 and Sentinel-2 Time Series Using Google Earth Engine. *Remote Sens.* **2019**, *12*, 76. [CrossRef]
18. Kraskov, A.; Stögbauer, H.; Grassberger, P. Estimating mutual information. *Phys. Rev. E* **2004**, *69*, 066138. [CrossRef]
19. Ross, B.C. Mutual Information between Discrete and Continuous Data Sets. *PLoS ONE* **2014**, *9*, e0087357. [CrossRef][PubMed]
20. Valero, S.; Morin, D.; Inglada, J.; Sepulcre, G.; Arias, M.; Hagolle, O.; Dedieu, G.; Bontemps, S.; Defourny, P.; Koetz, B. Production of a Dynamic Cropland Mask by Processing Remote Sensing Image Series at High Temporal and Spatial Resolutions. *Remote Sens.* **2016**, *8*, 55. [CrossRef]
21. Waldner, F.; Canto, G.S.; Defourny, P. Automated annual cropland mapping using knowledge-based temporal features. *ISPRS J. Photogramm. Remote Sens.* **2015**, *110*, 1–13. [CrossRef]
22. Kiala, Z.; Mutanga, O.; Odindi, J.; Peerbhay, K. Feature Selection on Sentinel-2 Multispectral Imagery for Mapping a Landscape Infested by Parthenium Weed. *Remote Sens.* **2019**, *11*, 1892. [CrossRef]

23. Robnik-Šikonja, M.; Kononenko, I. Theoretical and empirical analysis of ReliefF and RReliefF. *Mach. Learn.* **2003**, *53*, 23–69. [CrossRef]
24. Walton, N.S.; Sheppard, J.W.; Shaw, J.A. Using a Genetic Algorithm with Histogram-Based Feature Selection in Hyperspectral Image Classification. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 1364–1372. [CrossRef]
25. Gómez, C.; White, J.C.; Wulder, M.A. Optical remotely sensed time series data for land cover classification: A review. *ISPRS J. Photogramm. Remote Sens.* **2016**, *116*, 55–72. [CrossRef]
26. Zhu, X.X.; Tuia, D.; Mou, L.; Xia, G.S.; Zhang, L.; Xu, F.; Fraundorfer, F. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE Trans. Geosci. Remote Sens.* **2017**, *5*, 8–36. [CrossRef]
27. Qian, C.; Yu, Y.; Zhou, Z.H. Pareto Ensemble Pruning. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
28. Qian, C.; Shi, J.C.; Yu, Y.; Tang, K.; Zhou, Z.H. Parallel Pareto Optimization for Subset Selection. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, NY, USA, 9–15 July 2016; pp. 1939–1945.
29. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
30. Drusch, M.; Del Bello, U.; Carlier, S.; Colin, O.; Fernandez, V.; Gascon, F.; Hoersch, B.; Isola, C.; Laberinti, P.; Martimort, P.; et al. Sentinel-2: ESA's optical high-resolution mission for GMES operational services. *Remote Sens. Environ.* **2012**, *120*, 25–36. [CrossRef]
31. Canny, J. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **1986**, *6*, 679–698. [CrossRef]
32. Šircelj, B.; Kenda, K.; Koprivec, F. Land Patch Samples. *PANGAEA* **2020**. [CrossRef]
33. Teschendorff, A.E. Avoiding common pitfalls in machine learning omic data science. *Nat. Mater.* **2019**, *18*, 422–427. [CrossRef]
34. Zhou, Z.H.; Yu, Y.; Qian, C. Subset Selection: Acceleration. In *Evolutionary Learning: Advances in Theories and Algorithms*; Springer: Singapore, 2019; pp. 285–293. [CrossRef]

Chapter 5

Big Data Framework Based on Lambda Architecture

> Architecture is not about space, nor about containment. It's about relationships, about connections, about extending our experience into new territories.
>
> *Juhani Pallasmaa*

This chapter describes an enhancement as well as an evaluation of the lambda architecture (see Figure 1.1) in the water management scenario. The first section focuses on the implementation of the architecture, whereas the second section focuses on the evaluation.

5.1 Framework

> In this section, we introduce the paper entitled *Computer Architectures for Incremental Learning in Water Management*, authored by Klemen Kenda, Nikos Mellios, Matej Senožetnik, and Petra Pergar. This paper has been published in the Sustainability (IF 2022: 3.9) [1]. Klemen Kenda contributed significantly to conceptualization and methodology development presented in this paper. He contributed to software development, design and implementation of evaluation. He led the writing of the paper and also contributed to visualizations.

The paper presents an implementation of the platform based on the lambda architecture and modules described in this publication. The platform is coined the Water Management Analytical Platform (WMAP) and serves the data and modeling needs in the water sector. Implementation is steered towards solving real-world issues such as ingestion and wrangling of heterogeneous live data streams, forming model results on the live data and building custom solutions on top of standard predictive/anomaly detection services. The suggested framework builds upon established big data (lambda) architectures and offers an effective approach to address data-driven modeling within the water management sector. The primary enhancement lies in the speed pillar (online analytics) of the architecture. Here, we introduce heterogeneous data fusion across a range of data streams to deliver real-time data-driven modeling and prediction services more efficiently.

The architecture described above has been widely used in the H2020 NAIADES project. Figure 5.1 illustrates the implementation of this architecture. The NAIADES project consisted of three pilot sites, each addressing slightly different issues related to flower beds

watering, saline water intrusion in water distribution networks, water consumption, and water leakage. In this implementation, data enters the system through FI-WARE adapters in the Data Layer on the left side of the figure. Data, as well as various analytics results, such as feature vectors and predictions, are then exchanged between components using a pub/sub mechanism (via Kafka topics). The architecture includes two layers to cater to different analytics needs, namely basic and advanced analytics. The basic analytics layer comprises general-purpose components like data fusion and enrichment (described in Chapter 3), forecasting, and anomaly detection. On the other hand, the advanced analytics layer requires more domain-specific knowledge, and its components are tailored to specific use cases. An instance of this is the detection of leaks at the advanced analytics level, which is activated by services for anomaly detection provided by the basic analytics. The identification of the specific location of leaks is carried out through a leakage group finder that combines the results of physical models produced with EPANET with the capabilities of machine learning. Ultimately, the use of genetic algorithms facilitates the strategic placement of sonic sensors for precise leak detection. In addition to these analytics components, the WMAP implementation also includes other software components necessary for maintaining and troubleshooting the machine learning pipelines. Finally, all analytics results are sent back to the data layer and inserted into the NAIADES platform via FI-WARE adapters, where they are made available to the user.

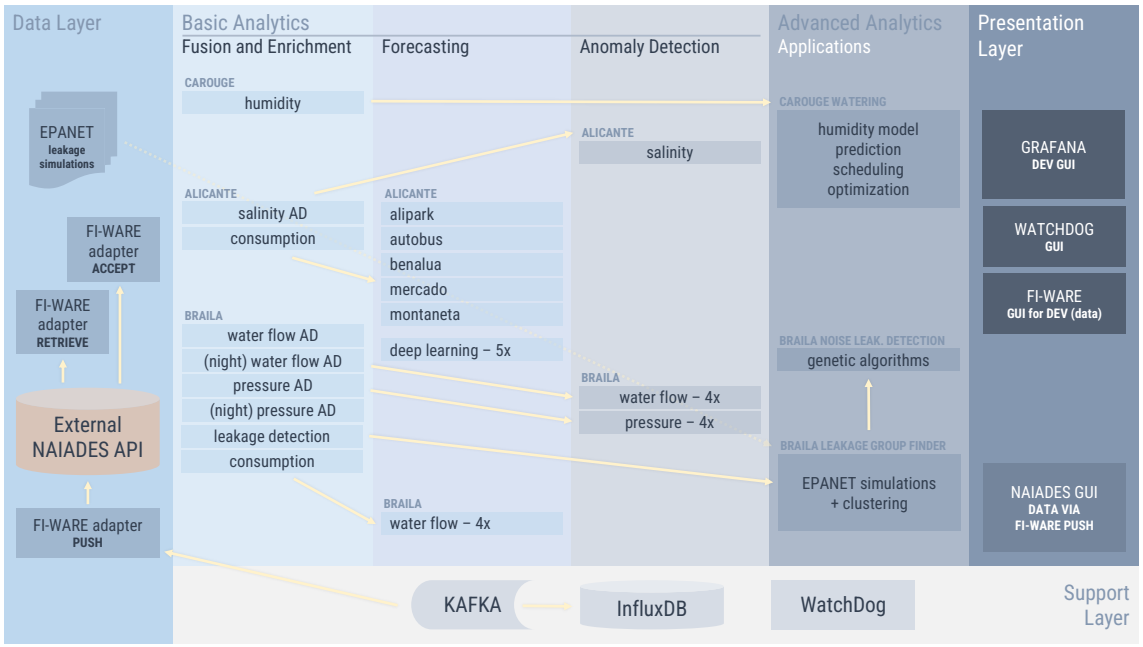


Figure 5.1: Implementation of the WMAP architecture in the H2020 NAIADES project.

sustainability

MDPI

Article

Computer Architectures for Incremental Learning in Water Management

Klemen Kenda [1,2], Nikolaos Mellios [3,4,*], Matej Senožetnik [1] and Petra Pergar [5]

[1] Artificial Intelligence Laboratory, Jozef Stefan Institute, 1000 Ljubljana, Slovenia; klemen.kenda@ijs.si (K.K.); matej.senozetnik@gmail.com (M.S.)
[2] Jozef Stefan International Postgraduate School, Jozef Stefan Institute, 1000 Ljubljana, Slovenia
[3] Department of Civil Engineering, University of Thessaly, 38221 Volos, Greece
[4] Municipal Enterprise for Water Supply and Sewage Treatment of Skiathos, 37002 Skiathos, Greece
[5] Ljubljanski Urbanistični Zavod, 1000 Ljubljana, Slovenia; petra.pergar@luz.si
* Correspondence: nmellios@uth.gr

Abstract: This paper presents an architecture and a platform for processing of water management data in real time. Stakeholders in the domain are faced with the challenge of handling large amounts of incoming sensor data from heterogeneous sources after the digitalization efforts within the sector. Our water management analytical platform (WMAP) is built upon the needs of domain experts (it provides capabilities for offline analysis) and is designed to solve real-world problems (it provides real-time data flow solutions and data-driven predictive analytics) for smart water management. WMAP is expected to contribute significantly to the water management domain, which has not yet acquired the competences to implement extensive data analysis and modeling capabilities in real-world scenarios. The proposed architecture extends existing big data architectures and presents an efficient way of dealing with data-driven modeling in the water management domain. The main improvement is in the speed (online analytics) layer of the architecture, where we introduce heterogeneous data fusion in a set of data streams that provide real-time data-driven modeling and prediction services. Using the proposed architecture, the results illustrate that models built with datasets with richer contextual information and multiple data sources are more accurate and thus more useful.

Keywords: water management; groundwater level; internet of things; data mining; stream mining; machine learning; data cleaning; predictive analytics

Citation: Kenda, K.; Mellios, N.; Senožetnik, M.; Pergar, P. Computer Architectures for Incremental Learning in Water Management. *Sustainability* **2022**, *14*, 2886. https://doi.org/10.3390/su14052886

Academic Editors: Carlo Giudicianni, Armando Di Nardo, Helena M. Ramos, Chrysi S. Laspidou and Manuel Herrera

Received: 26 January 2022
Accepted: 27 February 2022
Published: 2 March 2022

1. Introduction

During recent decades, the issue of water management has gained great interest due to the constantly growing water demand, the limited availability of global water resources, and the effects of climate change. Decision-makers strive towards utilization of optimal water exploitation regimes, realizing that new approaches need to be adopted in order to optimize the use of water resources. Scientific research, based on the latest technological advancements, seeks to establish reliable, effective, and innovative solutions for integrated water management and water resources preservation. These efforts are reflected in the growing smart water management (SWM) paradigm, empowering a variety of information and communication technology (ICT) methodologies. Wireless sensor networks and communication technologies provide a powerful inventory tool to water operators, enabling them to oversee significant water parameters in real time [1,2]. In fact, basic monitoring of the water-related data can be extended with data analysis techniques to gain deeper understanding of the underlying processes and even further with predictive analytics. This contribution might prove crucial to the challenges of future water management.

Managing water relies on heavy physical infrastructure investments and inherently reactive governing attitudes; thus, decision-making constitutes a challenging task for

the operators, especially under the framework of securing systems' sustainability and resilience. Compared to other disciplines, the water domain has not yet witnessed thorough investigation in terms of bringing together real-time monitoring, big data analysis, and machine learning with advanced control systems and the internet of things (IoT). Scientific research focuses on optimizing individual aspects of the water chain, while only recently it turned to the development of cyber-physical systems towards a holistic and digital-oriented approach to water management. Under this realization, the water management analytical platform (WMAP) brings a multifunctional advanced tool to the table which offers robust solutions in combining multiple types of data inputs and conducting analyses using multiple modeling frameworks, especially when there is a goal of producing near-real-time predictions as the basis of decision-making. Additionally, the ability to monitor and model water systems more accurately and respond more quickly to unexpected changes could provide a basis for adaptive management. The hydro-environment community could benefit from the proposed tool by bringing powerful optimization capabilities to practice and opening a wealth of opportunities for water management practitioners.

In many cases, water distribution networks monitor groundwater and surface water bodies with a variety of sensors. Typical measurements include water level, pressure and flow rate, water quality, etc. Related quantities such as precipitation, temperature, evapo-transpiration, pumping water energy, or data emerging from human behavior or remote sensing data are also relevant and are being collected by either water utilities or different environmental agencies [3]. Combining these data enables discovery of causative relations between water demand and demand drivers by means of demographic, socioeconomic, touristic, and infrastructure effectiveness indicators and provides an a priori knowledge on the patterns of water availability and demand [4]. Furthermore, all this knowledge can be used as an input to predictive algorithms and thus, water utilities and municipalities have the opportunity to plan water exploitation better, taking into account a fast-changing environment (mainly due to climate change and population growth). Typical use cases, where users benefit from efficient data-driven analytics, are in prediction of surface or groundwater levels, prediction of water quality parameters, prediction of water demand on household, district or urban levels, predictions of water demand in agriculture, etc. In order to exploit the water management data and implement intelligent solutions, a robust and efficient platform is needed. Some of the required functionalities include: data integration from heterogeneous (streaming) sources, data cleaning, enrichment and fusion, standardization of data access, and data mining and data-driven modeling capabilities that enable batch as well as real-time processing.

In this paper we describe an overall architecture of the WMAP and its integration into a real-world scenario. The presented solution is based on the EU H2020 Water4Cities project [5]. The architecture can ingest large volume of high-velocity data and process it either using online (close to real-time) or batch settings. The outputs of the platform match the needs of the water management stakeholders as identified in Section 2.2. Our main focus is on the online processing. We present data collection, data cleaning, and missing data imputation techniques as well as online contextual heterogeneous data fusion, which enables more accurate predictive analytics. We also introduce stream mining techniques for online processing into water management domain. The latter can effectively address many big data-related issues in an IoT setting. In such a setting, the user is often faced with high-velocity data streams from a large amount of fairly independent sensors. The consequence of the independence is twofold: (1) data streams can be easily processed in parallel (which means that this could be achieved even without the use of modern tools for distributed processing like Apache Hadoop, Spark [6] and others), and (2) stream mining models represent a computationally cheap solution to model the large amounts of data.

The contributions of this paper are as follows. (1) We present a conceptual architecture for WMAP, which we built upon our previous work on the numerous subcomponents. We developed our solution based on lambda [7] and hut [8] architectures. We suggest refinements and particular implementation details of the architectures in order to support

the needs of the water domain. (2) Our implementation supports real-world use case integration. The platform provides the complete pipeline for data analysis from its source (sensor, weather forecast, or other human-behavior data) to the final product (i.e., data-driven prediction for groundwater level or household daily water consumption profile). (3) To the best of our knowledge, we have introduced stream mining methods into the water domain. Stream mining techniques improve the computational performance of the pipeline and provide models that are better at adjusting to the changes (concept drift) in the real-time data than traditional batch models. (4) We integrated a solution for heterogeneous sources data fusion in a stream in the architecture, which enables contextual information to be included in the data-driven models and consequently increases the final accuracy of the models.

The paper is structured as follows. In Section 2.1 we present the relevant related work and explain the benefits of our approach. In Section 2.2 we present typical water domain use cases and corresponding data types. In Section 2.3 we present the architecture, which is able to handle the previously identified data and its integration into a real-world system. We proceed by presenting particular components within the architecture and corresponding results in Section 3 and finally we conclude in Section 4, where we also provide insights into the future challenges.

2. Materials and Methods

2.1. Related Work

The literature proposes a couple of architectures that are suitable for big data processing within the internet of things (IoT) applications. The lambda architecture was proposed by Marz and Warren [7]. The architecture includes two independent pillars for big data processing, the batched processing (batch layer) and the stream processing (speed layer). The serving layer presents a view of the results. The hut architecture [8] extends the lambda architecture by formalizing the data acquisition and message distribution (broker) components on the one hand and reduces the generic nature of the speed layer on the other. It reduces real-time processing to event processing, by which we lose the potential for real-time machine learning techniques. We propose two adjustments to the existing architectures: (1) We propose a concrete framework for realization of the speed layer in the lambda architecture which will serve the needs of water management domain. (2) We propose the extension of the hut architecture with real-time machine learning components within the speed layer (in contrast to only event processing, based on the rules generated in the batch layer). Additionally, we propose the existing hut architecture's principle to deploy models learnt in the batch layer to the speed layer, where predictions are generated on real-time data. Stream mining is a well-researched topic [9]; however, it lacks real-world applications [10]. Many incremental learning algorithms exist. Among them are methods that are based on stochastic gradient descent like recursive linear regression, support vector machines, and neural networks [11], methods based on decision and model trees and their ensembles [12] (i.e., streaming random forests) and incremental deep learning [13]. Many well-known platforms that enable large scale analytics (Apache Spark, Samza, and Flink), which are used in production systems, still lack implementations of more complex stream mining algorithms. Among platforms that do enable incremental learning techniques we can find academy-oriented frameworks such as MOA [14], scikit multiflow [15], and other smaller projects dedicated to a single algorithm only. The infrastructure is sufficient, but a unified production-oriented framework for stream machine learning techniques is needed. We have based our work on QMiner [16], which has been successfully deployed in many industry-grade use cases (from energy management to world news monitoring to anomaly detection in large computer clusters).

Our main contribution is related to the inclusion of advanced heterogeneous data fusion in the streaming setting. To the best of our knowledge, our methodology [17] is the only one that deals with data fusion in an online scenario. A similar platform, IoT streaming data integration (ISDI) [18], also solves real-time data integration using the

generic window-based algorithm paradigm. The platform deals with the time alignment issues and inherent heterogeneity of the incoming streaming data; however, it solves the issue of data fusion with a batched algorithm on top of a relational database. Other methodologies are mainly focused on solving non-heterogeneous sensor fusion, which is not suitable for solving problems in environmental data-driven modeling [17]. The issue has gained a lot of attention from the scientific community in recent years.

In the water domain, the research focus on data analytics ICT systems is spread among various topics. Research has identified the potential of artificial intelligence techniques for deepening the understanding of the acquired IoT data and for aiding decision-making processes. Various architectures and functionalities have already been researched in several areas of water management [19]. For instance, ref. [20] introduces a case study where they deploy an IoT-enabled platform in order to gather data for precision agriculture and ecological monitoring. In another case, ref. [21] presents a web-based platform for water efficient households. The platform enables consumers to monitor and control the water and energy consumption of their households in real time. From a hydrological perspective, a global web-based catchment area (river basin or urban subcatchment area for rainwater) hydrological information platform allows both scientists and non-expert users to easily access and visualize hydrological information for local-level water management and water stewardship in catchments [22]. Although the aforementioned ICT platforms provide advanced and innovative features for water management, their design does not favor easy adaptation to other uses. Their main drawback is the lack of support for standardized data exchange protocols, while they also need to address connectivity with additional data analytics tools. Our platform (WMAP) is implemented as an integrated support tool that has the potential to adjust to different stakeholders' needs, combining fast data acquisition, data fusion capabilities, and low-computational stream mining methods.

On a conceptual level, the usefulness of a big data analytics platform for groundwater management has been recognized by the Southern African Development Community [23].

2.2. Typical Use Cases and Data Description

There is a great range of typical scenarios where ICT systems have the potential to improve efficiency and facilitate critical decision-making in the water domain. Precision irrigation, optimization of water distribution networks, preservation of environmental flows in lake and river ecosystems [24], prevention of extreme events (floods and droughts) along with water-oriented urban planning, present only a few interesting scenarios. Traditionally, water management has been driven by process-based models focusing on revealing the mechanisms of natural water resources and simulating the operation of water-related infrastructure. Although modeling provides reliable and useful insights in relation to critical parameters of the water cycle, its usability is limited and often fails to produce timely solutions. The complexities and extensive detailing involved in the representation of water processes make the development of process-based models extremely challenging. Some of the major drawbacks arise from issues related to miscalibration, over parameterization, high computational requirements and extensive data preparation, while adapting to changes and capturing hidden system dynamics is often impossible. Data-driven methods, on the other hand, bridge some of the shortcomings of the process-based models by relying heavily upon machine learning methods. This gives them the ability to overcome the (in several cases) unknown physics of the modeled system by exploiting the information implicitly hidden in the data. In agriculture, data analytics can reveal the causal relations between irrigation and crop production and assist with preparation of reasonable and precise irrigation schedules in compliance with environmental and socioeconomic aspects. Furthermore, predicting water availability and demand, detecting anomalies throughout the water distribution system, and raising alarms in case of water quality deterioration present an important added value in urban water management. In terms of lake and river ecosystems management, predictive analytics can allow proactive interventions both in regulating environmental flows and preventing water quality degradation [25]. Floods

and droughts, due to their devastating consequences on ecosystems, food supply and economies, present a rather sensitive sector, where the proposed platform could proactively predict such events and reveal the circumstances under which they may take place.

Since water management is often a complex process, encompassing various activities and involving different stakeholders, the optimal use of water resources implies a wide range of actions. Water system monitoring (by in situ sensors or remote sensing) is essential in order to understand the underlying behavior of critical components of the system. Monitoring includes data collection from various sources in (almost) real time. In this paper, we assume that the underlying IoT infrastructure is already in place. Our tools provide a data management and analysis level on top of the sensor layer. Typical data sources in the water domain are (see Table 1):

Table 1. Typical data sources in the water domain and corresponding indicative subcategories.

Typical Data Sources in the Water Domain	Indicative Subcategories
Surface and groundwater bodies data	groundwater level and pressure; permeability and storage capacity; river water level and discharge rates; flood inundation areas
Meteorological data	precipitation; temperature; evaporation; wind speed; radiation
Water repository data	accessible storage volume; water storage bathymetry and level; reservoir or tank water level; storage inflows, outflows and offtakes
Water exploitation data	volume of water taken from groundwater, rivers, lakes, and storage infrastructure; water pumping data
Water quality data	temperature; pH; oxygen; nutrients; chlorine concentration; electrical conductivity
Water pollutant data	heavy metals concentration; fertilizers; pesticides; bacteria; algae
Water distribution data	flow rate; pressure; energy consumption
Human-behavior data	water consumption; migration/tourism; public participation data
Spatial data	infrastructure; future and current land use; water bodies; static data based on previous measurements and process models; surface and topology; geological data; risk maps
Administrative data	water management area boundaries; water prices; water infrastructure inventories

Collected data can be further analyzed and used for modeling. Data-driven forecasting has the potential to reveal system dynamics and produce meaningful and accurate predictions about the state of crucial system components. Thus, water experts and water operators will have the advantage of obtaining the necessary knowledge to proceed in effective water management plans and secure sustainability of water resources.

Possible scenarios for usage of data-driven modeling are:

- Providing water security in agriculture (predicting water availability and demand, regulating irrigation schedules, setting sustainable limits on water allocation);
- Delivering water supply services (predicting supply and demand fluctuations, predicting availability of water resources, securing adequate water quantity and quality, semantic annotation of water demand, detecting anomalies throughout the water distribution network in households or district areas in terms of leakage, theft, etc.) Securing water in aquatic ecosystems (specifying environmental flow regimes to achieve sustainability, identifying water contamination, and regulating the quality);

- Reducing flood and drought risk (in-time storm water "hot spot" localization by operating early warning systems, constructing efficient flood control infrastructure, predicting drought events and taking the necessary preventive measures);
- Promoting integrated urban water management (IUWM) (suggesting possible locations of nature-based solution (NBS) interventions, designing land use change allocation, assessing groundwater levels for urban planning extension).

We present two illustrative examples of typical water management use cases.

Example: Island of Skiathos, Greece. Skiathos Island is a typical Greek island with a high touristic influx during the summer, which exceeds its population of 5000 inhabitants. The influx results in a sharp increase in water demand during summer. This, combined with an aged distribution network with high water loss due to leakage, means the island often faces water shortage issues during the touristic peak. The quantity and quality of groundwater, which serves as the island's water supply, are being increasingly deteriorated; thus, the water operator was forced to take actions towards a more efficient and rational water management plan.

Balancing the water supply by means of water abstraction regulation, effective pressure control schemes, and improvements to ageing infrastructure will help to rationalize the use of the finite water resources. However, the accomplishment of an effective management regime requires a thorough insight in relation to hydrological and hydraulic parameters and other related variables. Aquifer water level, water abstraction rate, pressure throughout the water supply network, seasonal water demand levels, touristic arrivals, weather predictions, etc., are some of the parameters that the water operator needs in order to apply different water supply regimes in accordance with demand [26].

Additionally, in terms of water quality, measurements have shown mercury concentrations above the safety threshold in the groundwater. Thus, water cannot be used for drinking or cooking purposes and people turn to bottled water to cover these needs. Increased mercury concentrations are linked to high salinity in the water; therefore, it becomes important to quantify seawater intrusion, which entails measurements of groundwater level, temperature, and conductivity.

In order to proceed to a smart water monitoring scheme, the operator installed a range of wireless sensors measuring flow, pressure, water level, temperature, and water quality parameters, producing large amounts of real-time data. However, these data are not interoperable, so the water operator is faced with the need to install various different platforms to get access to the data, while any data fusion exercise is an arduous task. From this perspective, Skiathos presents a suitable case for the implementation of our proposed water management analytical platform. The uniform access to the data along with the implementation of data analytics and prediction algorithms will not only serve the efficient real-time data monitoring but will enable new insights into water management planning.

Example: Ljubljana urban region, Slovenia. The goal of the Ljubljana case study is to provide a reference system that will enable integrated urban water management (IUWM). IUWM is based on existing water supply and sanitation principles within an urban area by involving urban water management within the scope of the entire river basin [27]. The conceptual framework and approaches regarding a more efficient IUWM have evolved the past decades to involve new technology solutions, the idea of integration towards the holistic theory, and the public participation through awareness raising and participatory designing. The IUWM system should enable stakeholders to gain deep knowledge of their water systems through the clever visualization of key design parameters, and a valid simulation that will complete and interpolate their information. This way, hidden elements of the urban water cycle will be revealed as well as cause–effect relations. Stakeholders, including citizens, will benefit from optimum and inclusive design. There is a need to build an appropriate system that will enable IUWM decisions on an urban district level.

We found the Ljubljana case study suitable as a reference system because the urban area of the Ljubljana City spreads between two rivers—the Ljubljanica and the Sava rivers, the latter being the main Slovenian river discharging water from the Alpine mountains in the

north-west of the country. The urban and agricultural area between the two rivers is where almost 15% of Slovenian inhabitants live and work and the main groundwater resource at the same time. Two recharging components of the Ljubljanskoaquifer, i.e., the local precipitation and infiltrated Sava River, are exposed to different sources of contamination because they originate from different parts of the hydrological circle. The area of the city of Ljubljana has a long history of various flood protection measures. Nevertheless, many parts of the urban area of the city are still heavily threatened by floods, which are a consequence of intensive urbanization, surface run-off increase, as well as climate change effects.

The use of information and communication technologies (ICT) enables IUWM by weighting the measure not only in comparison with comparable measures, but also against other aspects of planning [28]. It is important that any system results are precise enough to enable IUWM decisions (e.g., investments in infrastructure, city master plan rules); therefore, the big river catchment areas need to be sliced into suitable sub–catchment areas on an urban district level. Freshwater, wastewater, and storm water constitute the parts of the urban water cycle, while the urban surface, the aquifer, and water supply infrastructure constitute the linking mediums that intervene in the natural water cycle, forcing an altered, disturbed urban water cycle. The level of disturbance determines the consequences for water availability, water quality, and the regime of flows that with the increase of extreme events may (or already) constitute a severe threat to social integrity, urban infrastructure, the economy, and the natural environment. The architecture should support ingestion of the identified data sources and provide mechanisms to perform the usage scenarios. The interaction of groundwater with other urban systems, such as infrastructure and surface water networks, is well recognized by expert practitioners and is increasingly important to the everyday city agenda [29]. Therefore, the first step towards the IUWM reference system in Ljubljana is presented with groundwater level sensor data.

2.3. Architecture

Lambda [7] and hut [8] architectures provide a solid basis for building the water management analytical platform (WMAP). According to the identified usage scenarios in the water management domain, we propose a couple of modifications which are depicted in the WMAP architecture in Figure 1.

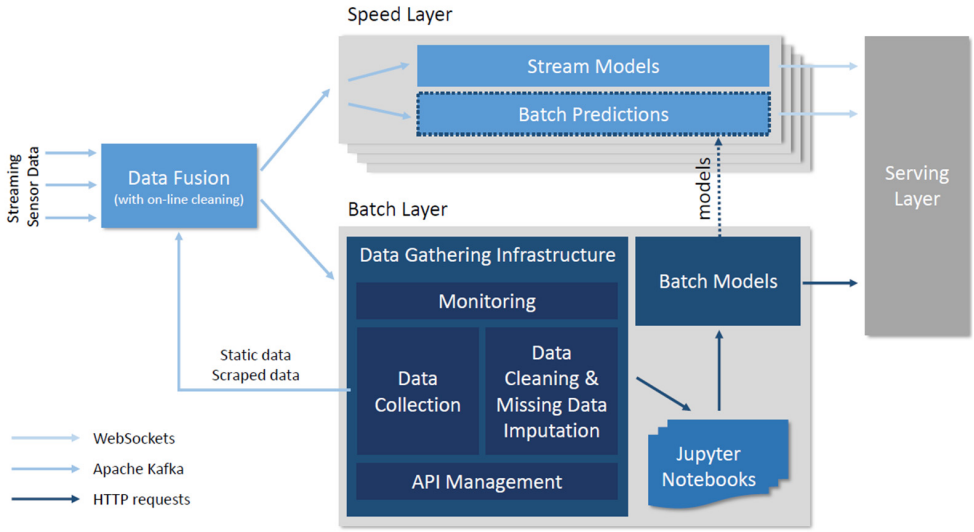


Figure 1. Proposed modification of lambda and hut architectures for usage in the water management domain. Heterogeneous input data streams are fused in the data fusion component and pushed into both speed and batch layers. The speed layer includes two types of models—incremental (being updated with only new data online) and batch. Batch models are updated with all the data from the batch layer.

Arrows in the figure depict data flows. Streaming data is handled by Apache Kafka infrastructure in the back end and with Web Sockets (which can be ingested by HTTP clients) in the serving layer. Static data, web resources, and communication within the batch layer are handled with HTTP requests. The data enters the framework directly from sensors (and corresponding adapters) and from data collection in the data gathering infrastructure (from different web resources that need to be polled for the data). All the data undergoes the initial online data cleaning and fusion. The data fusion component is depicted in more detail in Figure 2. This component provides ingestion of different heterogeneous data streams (including different streams from the internet of things, weather forecasts, and static data on human behavior). All these streams are enriched (with different aggregated values of the stream). In the next step, we join the streams together with special attention to their records' original timestamp. Finally, we compose the feature vectors. Fused (feature vectors) are injected into speed and batch layers and used for predictions of further modeling. Raw data is also pushed into the batch layer for further offline analysis. It is worth noting that we propose the usage of data-driven machine learning models even in the speed layer (we do not limit it to event processing). Technologies like heterogeneous streams data fusion and stream mining provide fast alternatives to traditional data-driven analytics. Finally, results are exposed via the serving layer.

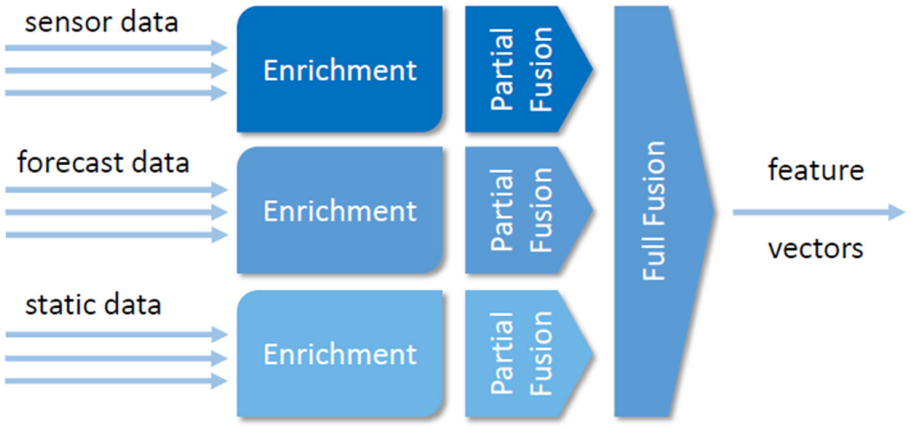


Figure 2. Design of data fusion component in the speed layer. The component is able to ingest three types of data streams and output a stream of feature vectors.

The speed layer consists of two different predictive models: stream models (which are based on incremental learning) and batch predictions, which implement batch models developed in the batch layer. Both components provide similar functionality; however, incremental models are updated with each new measurement whereas batch models need to be updated from the batch layer. As shown in the data description, the uses in water management consist of larger number of contained data-mining problems. Parallelization of the computational tasks in such a setting emerges naturally. Each use (sensor) is independent and requires limited needs for computation power. Therefore, the load can simply be balanced over a set of workers, which are connected by data distribution infrastructure as depicted in Figure 3. The whole streaming pipeline (data fusion and speed layer) is generic and is described in more detail in related work [17].

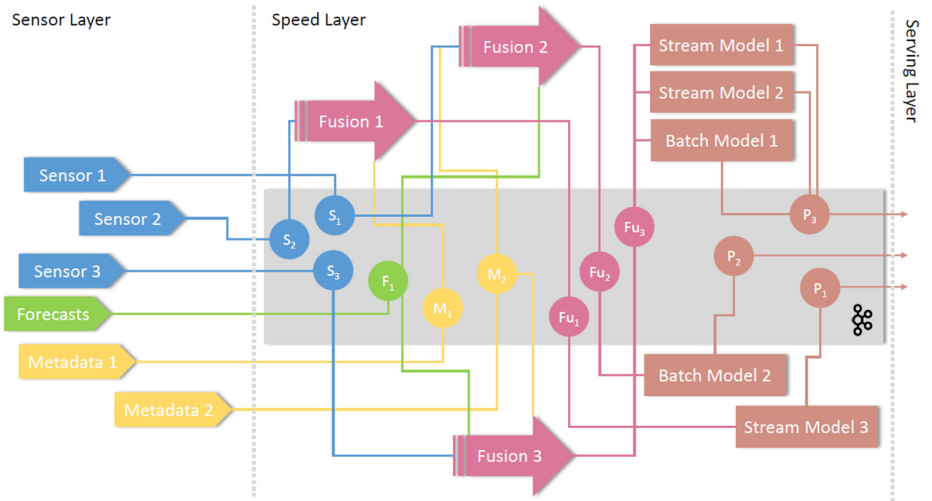


Figure 3. Example of speed layer implementation and interconnection of particular components. Message distribution component is depicted in gray.

The batch layer includes data gathering infrastructure (a data layer that can consume, store, and serve large volumes of data according to the specific use, and is able to perform data cleaning and missing data imputation) and batch models with Jupyter Notebooks to support machine learning on top of these data. The aim of the batch models component is twofold: (1) the learned models is fed into the batch predictions component in the speed layer, and (2) data-mining process results are provided from the batch modeling component, which is useful for on-demand processing. The layer also includes monitoring and API management (to handle availability and access to the data).

2.4. Notes on the Implementation of Speed Layer

An illustrative example of the implementation of the speed layer is depicted in Figure 1. Sensor, speed, and serving layers are positioned from left to right. The gray rectangle depicts the data distribution infrastructure, which takes care of transferring streaming data between components. The blue color depicts components and data flow related to sensors (IoT). The green color represents forecasts (i.e., weather forecasts). The yellow color represents static metadata sources. Data fusion components are depicted in purple, and finally the modeling components and its predictions are depicted in brown.

Data are flowing from left to right, originating from sensors or other sources, being infused into appropriate message channels or topics (depicted as circles) in the data distribution infrastructure and then consumed by different instances of fusion components. These components can consume data from the same or from a different set of sensors, forecasts, and static metadata sources. Here the data are enriched, validated, consolidated, and finally merged into viable feature vectors suitable to be consumed by machine learning models. The models (either batch or stream based) consume the data from fusion message channels and provide results to prediction message channels. Multiple models can consume data from the same fusion message channel, meaning that we can easily provide multiple predictions with heterogeneous properties for the same scenario.

There are many viable platforms to be used for message distribution. We have integrated Apache Kafka (https://kafka.apache.org/, accessed on 1 January 2022), which seems to be the favorite choice in terms of performance and functionality, and RabbitMQ (https://www.rabbitmq.com/, accessed on 1 January 2022) in some cases. The data fusion component has been implemented with QMiner (http://qminer.ijs.si, accessed on 1 January 2022), which enables large-scale data analysis and provides methods (stream aggregates) for enrichment and consolidation of the heterogeneous streaming data. Stream models have been implemented using the same framework, but experiments have been performed also

with Scikit-multiflow (https://github.com/scikit-multiflow/scikit-multiflow, accessed on 1 January 2022) and MOA (https://moa.cms.waikato.ac.nz/, accessed on 1 January 2022), which offer a larger variety of algorithms. However, none of the platforms for stream learning algorithms has reached the maturity that could enable it to be used in production setups with ease and confidence. Batch modeling has been developed using Python scikit-learn (https://scikit-learn.org/, accessed on 1 January 2022) library.

Particular parts of the platform are described in the Section 3. Every subcomponent of our architecture has been described in a separate paper, which is cited in every subsection below.

3. Results and Discussion

3.1. Data Gathering Infrastructure

Data gathering infrastructure was implemented as part of the batch layer and is depicted in the bottom of Figure 2. The layer is described in more detail in related work [30]. It consisted of a data collection component, which controlled subscribing, polling, and preprocessing of external data sources. The latter can include remote devices, sensors, external data access APIs, and other web resources. A built-in feed monitoring component provided the ability to notify different stakeholders about failures and anomalies in the incoming data streams. The data were stored in a MongoDB NoSQL database. The latter allowed handling of data records with flexible schema. We have identified this as a useful feature, since some feed formats may evolve over time and records from different time intervals may contain different data fields. Additionally, the batch layer offered end users secure and uniform access to the data and easy integration with widely used data analysis tools such as Jupyter Python and R notebooks, Matlab scripts, and others. The data gathering infrastructure could also be used to trigger stream simulations, which are helpful for testing and development of streaming models. Loosely coupled data collection components could be scaled horizontally in order to improve performance.

In Water4Cities scenarios, we have stored the information about groundwater levels, pump sensors, and weather data. Special attention was given to the ease of integration of different data sources by providing boilerplate code in various languages (Python, Java, JavaScript). The platform performance has been proven adequate in the traffic domain, where we retrieved and stored approximately 100,000 records per hour (records include images and heterogeneous sensor data for Slovenia), collecting more than 1.2 TB of data per year. With just this amount of data, we could monitor the whole center of Ljubljana with water smart meters with an update interval of 15 min in every household. As data collection components could be distributed to different machines, the bottleneck was represented by data storage [31]. Even with a single high-end server the authors were able to achieve throughput of 1882 records per second, which is equivalent to 1.6 million records in 15 min. Such a setup would be adequate for the whole Ljubljana region and the database could have been scaled horizontally by adding new machines to balance the system load.

3.2. Missing Data Imputation and Data Cleaning

According to the CRISP-DM methodology [32], the process of data mining is devised into six separate stages. Modeling itself represents only one of these stages and the majority of a data scientist's work time is invested in a process of understanding and preparing data. Data preparation includes data transformation, data cleaning, missing data imputation, and also data fusion, which is described in Section 3.3. Algorithms for data cleaning and missing data imputation differ significantly between the speed and batch layers. Within the speed layer, the algorithms should be simple and efficient and should rely only on historic data of a time series in question. They should also be autonomous and should not rely on any expert intervention, as the data stream is continuous. We proposed the usage of the Kalman filter's short-term prediction capabilities in order to address outlier detection (cleaning) and sporadic missing sensor readings in a data stream [33]. In the batch layer it is feasible to use more complex and therefore more effective models that can rely on any

data within the dataset and exploit the power of machine learning to find optimal models for missing data imputation as well as for anomaly detection, as described by [34].

An example of the results of the missing data imputation algorithm on a "Ljubljana polje" groundwater levels dataset [35] is depicted in Figure 4. Nearby sensors were used to predict the missing values of another sensor. High accuracy of the methodology suggests that the data from the sensors were reliable. Accuracy of the sensors maintained by Slovenian Environment Agency is ±0.01 m [36].

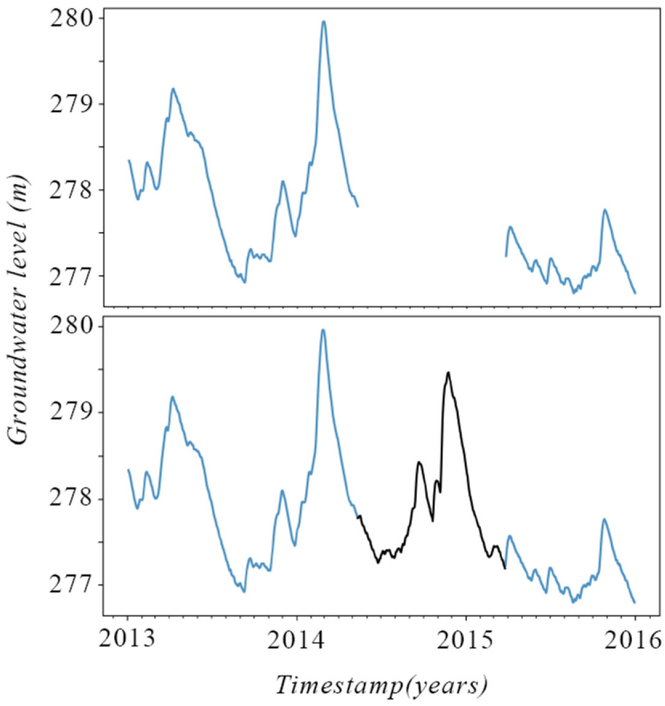


Figure 4. Example of missing data imputation algorithm based on nearby sensors, $R^2 > 0.99$. The dataset with missing data is depicted in the top panel. The imputed part of the time series is depicted in black in the lower panel.

For groundwater level data in the Ljubljana region, we were able to achieve high $R^2 > 0.997$ scores for highly correlated sensors (correlations up to 0.98) and were able to improve scores significantly for sensors with low correlation to nearby sensors. A more detailed description of the dataset, methodology, and results is given in related work [34].

3.3. Real-Time Heterogenous Data Fusion

An intrinsic property of big data is its heterogeneity [37]. In a WMAP system, heterogeneity has been observed regarding data delay (data are submitted via various legacy systems, which bring systematic constant lags, sometimes send data in batches every hour or every day, etc.), data frequency, data type (i.e., weather forecast time series is updated every hour whereas the sensor stream is coherent in regard to its time component). To provide efficient and accurate predictive models, the usage of multiple data sources is essential. Feature vectors that contain additional enriched and contextual data will normally provide additional information to the predictive models and finally result in better prediction accuracies. To the best of our knowledge, our solution [17] is the first to mathematically describe and solve this problem.

The data fusion architecture within the speed layer is depicted in Figure 2. There are three basic tasks to be accomplished within the data fusion: stream enrichment, partial fusion (of sources of the same type), and full fusion. The stream processing component supports ingestion of three different types of streams: sensor data (which might have

various frequencies and delays), forecast data (which is being updated regularly, i.e., weather forecasts are updated every hour for the next 48 h), and other contextual and human-behavior data (usually static pre-generated data, which we include as a stream in our platform).

Every data stream was separately transformed (i.e., we transformed weather predictions into a number of regular data streams) and enriched with stream aggregates (i.e., moving averages, variances, minimums, and maximums in different time windows).

Final feature vectors were generated from partial feature vectors within the fusion component, where time consolidation was done. Time consolidation is a process in which we bring all the partial feature vectors to the same master time (we handled delays and different update frequencies here; we also provided constant sampling; i.e., every 15 min a feature vector was generated, which is essential for most machine learning techniques). Finally, the fusion component expands the feature vector with historic data or even some derivatives (i.e., difference between hourly moving averages in the past hour), which often provide viable information to the data models. Final feature vectors are provided to stream or batch models, which calculate final predictions.

To the best of our knowledge, no methodologies described in the scientific literature can match the expressiveness of our system's feature extraction language and cannot handle heterogeneous streaming data fusion. As argued in the next subsection, such data fusion is beneficial for improved prediction accuracy. There are, however, no direct validation methodologies for data fusion systems, especially not for the online versions' latencies.

When profiling the pipeline, it is evident that the data fusion system works much faster than the optimal prediction algorithm. On a server machine, the average time to process the message in the online data fusion framework is 0.2 milliseconds (Figure 5). However, a prediction step with the random forest method takes on average 63 milliseconds on a server and 740 milliseconds on Raspberry Pi 2 (Figure 5). It is, therefore, important to have an architecture to be able to parallelize the processes (in our case any number of data fusion, stream models, and batch predictions could be running distributed in the network, reachable by message distribution system). The lesson learned in the performance validation of the system was that the message distribution system (i.e., Apache Kafka) has to be optimized within the network. If not, the bottleneck can be in the message distribution and not necessarily in the processing power.

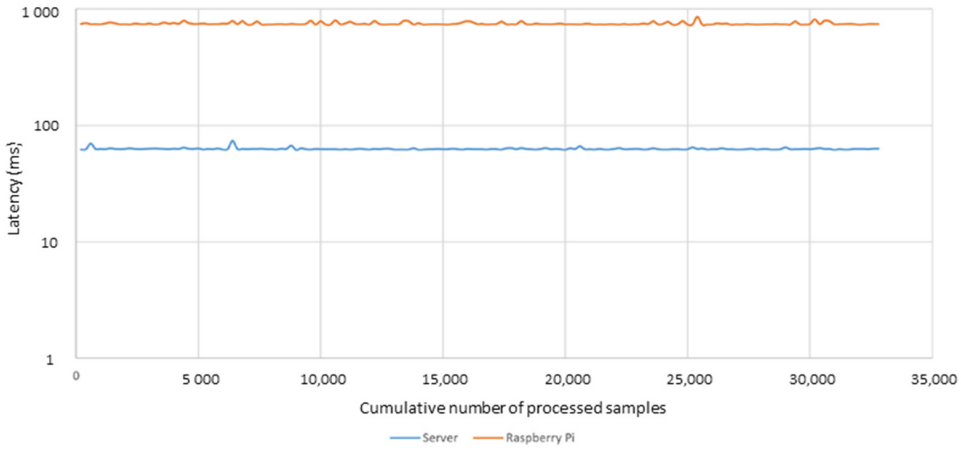


Figure 5. Evaluation of the performance of the speed layer setup (data ingestion, streaming data fusion, and modeling with random forest) on a high-end server (blue) and Raspberry Pi edge device (orange). Apache Kafka was running in a dockerized container on the same high-end server.

3.4. Data-Driven Modeling

Based upon the experience in the other fields that have faced the artificial intelligence breakthrough in recent years (physics, remote sensing and earth observation, energy

management, etc.), we propose to bring the data (for batch analysis) to the users and let them manipulate the data in the fashion that is standardized or is based on the tools they are already familiar with. Our data gathering infrastructure provides a uniform access point for heterogeneous data sources and provides basic boilerplate code to speed up data analysis and development. An expert can access the data in the tool of their preference (i.e., Jupyter notebook), perform analysis, and even deploy their own models to the production by simply updating corresponding configuration structures (feature vectors and models).

As demonstrated in more detail in related work [38], a time series based on groundwater level sensor data can be enriched with various derivatives and with appropriate contextual data. Models will benefit from features such as readings of level change in the past hour, past day, or past week, moving average in the past day or past five days, or even past month. Weather and especially historic weather data (precipitation, snowfall and change of snow blanket) from the area and corresponding river basin, aggregated over a longer period (typically one week) will show good correlations with groundwater level change and will improve the models significantly. In particular areas, the time of the year might expose some typical local dynamics, etc. As stated in the related work section, other generic stream processing platforms do not enable heterogeneous sources data fusion and can therefore not easily provide enriched feature vectors which yield more accurate predictions. In addition, our data fusion component enables easy manipulation (with a single configuration structure) of stream aggregates (such as moving averages or variances) and their historic values.

Experiments have been conducted on the Ljubljana polje aquifer dataset [35], which is also available via an endpoint (data gathering infrastructure API) exposed by the platform, described in this paper (check the dataset source for additional instructions on how to use the API). The dataset includes more than 600 groundwater stations for Slovenia, which measured groundwater levels between years 1950 and 2018. A subset relevant for the Ljubljana polje aquifer was used in the experiments. Weather data were provided by Slovenian environment agency (ARSO) and were also included in the dataset. Normalization has not been performed on the data, since tree-based methods do not benefit from it and because the features themselves are in the same order of magnitude, which should be sufficient for the convergence of the linear regression models.

Feature correlation matrix for one of the time-series is depicted in Figure 6. It includes groundwater level features and weather features. Initial features are from the original dataset. The others have been extracted from historic values. Weather features have been averaged by various intervals (from 1 to 100 days) and different averaging windows have been considered. Next, these features have also been shifted by different time intervals in order to compensate for the time needed for weather-related phenomena to have effect on the groundwater. Based on the correlation matrix we have chosen features from the top 100 features correlated with the target value. Best- and mutually least-correlated features (according to Figure 6) have been selected for each feature subset (aggregated data, aggregated weather data, shifted historic weather data). All the raw values have been considered in the models. Correlations of features with the target value vary from -0.4 to 0.78 (precipitation averaged over next 3-day weather forecast is the best correlated feature). The feature selection process can be achieved using a more thorough search through the grid of features. Greedy algorithms would not be able to accomplish this task in a reasonable time; however, smart heuristics powered by genetic programming and entropy-based similarity measures can canvas the most relevant sections of the feature space and extract (almost) optimal feature vectors from a large feature space, usually further improving accuracy of a particular model [39].

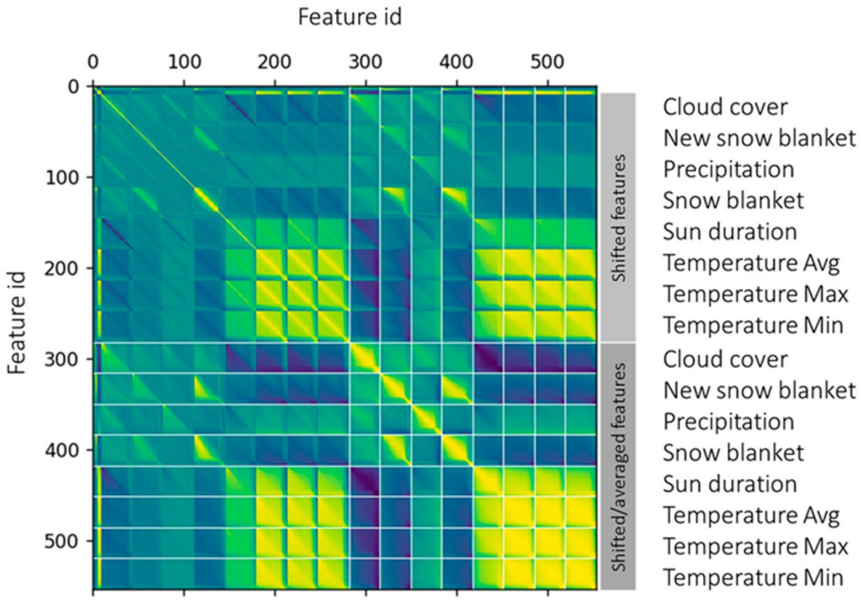


Figure 6. Correlation matrix includes 544 features. The features are the original ones represented in the dataset (such as weather data and current and historic groundwater levels) and derived ones (such as averaged and shifted by different time intervals). Our platform enables online generation of all these features. Positive correlation is depicted in yellow and negative in dark blue shades.

Accuracy of the models has been evaluated using R^2 score. R^2 is invariant to offset the target value from 0 (which is not true for other relative scores like mean average percentage error—MAPE) and to amplitude within the dataset (which influences the root mean squared error—RMSE). R^2 is therefore suitable for comparison of different approaches.

Figure 7 depicts learning curves (improvement of R^2 score with number of training examples) of different models for prediction of groundwater levels (with 3-day prediction horizon) in the Ljubljana region. Each model (defined by a learning algorithm and a feature set) is represented by a curve in Figure 7A–D. Curves with a larger number of contextual features are higher in the learning curve graphs, which indicates the benefits of the methodologies presented in this paper. Figure 7A depicts results based on linear regression, Figure 7B on decision trees, Figure 7C on gradient boosting regression and Figure 7D on random forests.

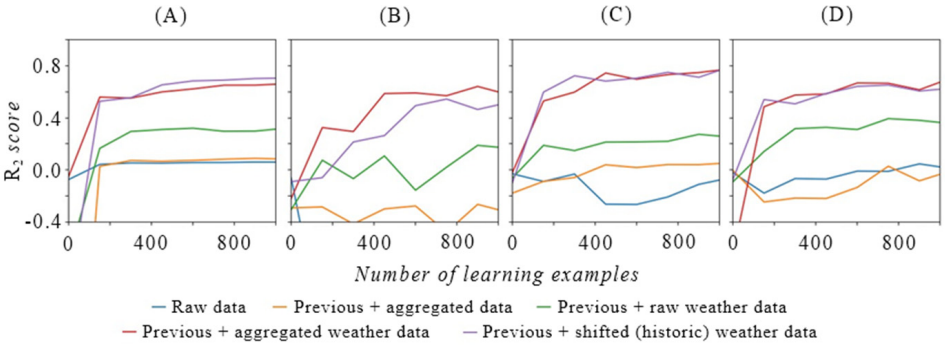


Figure 7. Learning curves of plain and enriched datasets with multiple learning methods: (**A**) linear regression, (**B**) decision trees, (**C**) gradient boosting regression, (**D**) random forest regression.

In every picture we can observe five different curves that represent five different feature sets. The blue curve represents a feature set with only direct features from the groundwater level time series, the orange curve represents the "blue" feature set enriched

with various stream aggregates, the green curve represents a feature set which also includes current weather data. The red curve is based on the results from a feature set which is further enriched with various aggregations of weather data, and finally the violet curve models include also time-shifted weather features to reflect the potential hysteresis effect of weather on groundwater levels.

Example results for groundwater level predictions (using linear regression and gradient boosting) are shown in Figure 8. We have modeled daily changes of the groundwater (depicted on the left side of Figure 8) and then calculated cumulatives (on the right side of the same figure). Cumulative values show how well the model captures the dynamics of groundwater levels. The values themselves diverge from the true values, but the important information is that the models reflect the trends in the real world well.

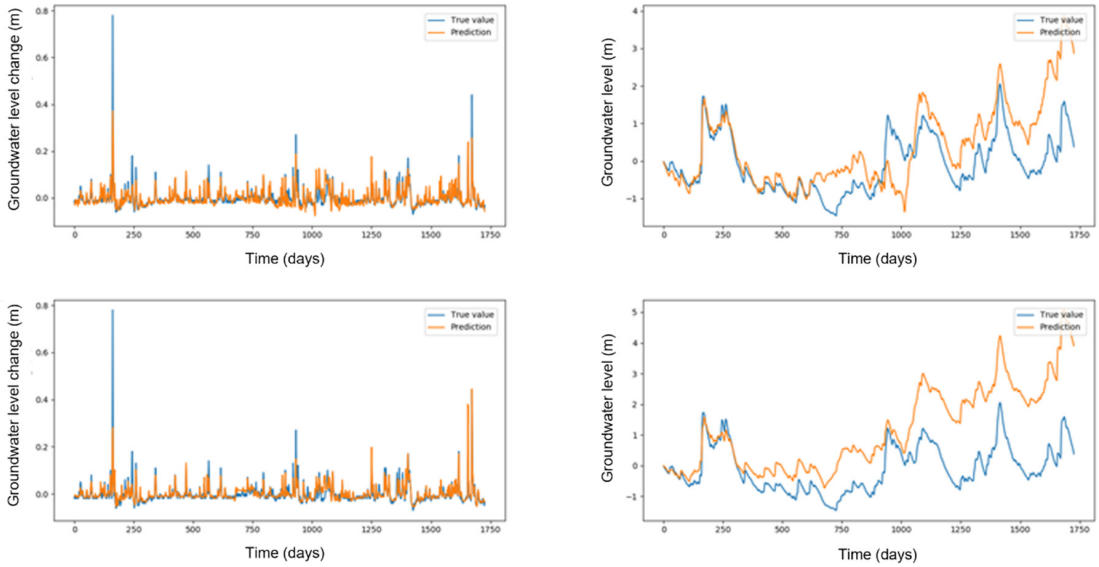


Figure 8. Groundwater level change predictions and summed predictions over time. First row depicts linear regression results, the second row depicts gradient boosting results. Summed predictions may drift over time, which does not give an objective measure of the model's accuracy. Capturing the correct trends is much more important.

Each consecutive dataset includes more potentially relevant features that enable the model to reflect the underlying process better. Results, represented with blue and orange lines, which only include features based on the groundwater level time series itself, behave even worse than a constant zero model. As soon as additional weather data are considered (green line), the accuracy improves drastically. The biggest improvement is however achieved by including different time-window and time-shifted aggregates of weather data, which seem to reflect dynamics of groundwater adequately. The linear regression in Figure 7A gives stable results (curves rise with the number of learning examples). Each addition to the feature set increases the accuracy of the model by bringing new knowledge. Decision trees in Figure 7B behave slightly worse and are much less stable. We also observe that decision trees learn slower. They need more examples to reach comparable accuracy to linear regression. The learning rate of gradient boosting regression in Figure 7C and random forests in Figure 7D matches the speed of linear regression. Random forests, which are usually the method of choice in environmental data-driven modeling, behave best with the raw weather dataset (green) and aggregated weather dataset (red), but slightly worse than linear regression in the best-case scenario. The best results are given by gradient boosting regression, which can achieve an R^2 score close to 0.8. As a baseline, a model with only the best feature would yield $R^2 \approx 0.6$, which is 0.16 less than the results of gradient boosting with the best feature subset.

Usage of data-fusion and data-driven modeling within the speed layer of the architecture enables real-time application of the predictive analytics, developed in batch mode.

4. Conclusions

We have presented a water domain view on the data mining approaches within a smart water management scenario. We have identified the needs of stakeholders and provided a description of typical data sources that support achieving the desired results. We have presented an architecture based on standard big data approaches and an early prototype which enables offline analysis tailored to the stakeholders' needs and real-time predictive analytics that can be applied to real-world scenarios. The platform provides parallelization of data processing, which enables horizontal scalability of the system. The measurements, however, demonstrate that even a single high-end server can support a reasonably big project with up to 400,000 connected IoT devices. The main contributions within the streaming part of the architecture (speed layer) are the data fusion component and the usage of computationally less demanding stream mining techniques for predictive analytics.

Although a lot of work has been done in the field of digitalization in the water management domain, there is still a gap to be bridged in the water management domain to reach its counterparts in energy management, traffic, and manufacturing. Many solutions have been developed in those fields, especially in energy management, which can be applied to the water domain. Understanding of dynamics in the water domain has traditionally been supported by process-based models, which require extensive knowledge of the domain, including geology and fluid dynamics. With more data provided, machine learning models can provide an efficient alternative to the existing state, simply because they are easier to implement, because they reflect the current state of the system immediately (including human behavior-related variables), because predictive techniques have been proven to mimic hidden process dynamics and because they can be cost effective. As both approaches, data-driven and process-based, have their own advantages, they should work hand in hand, which offers another research challenge for the future. On the other hand, the water community has to embrace and gain trust in data-driven approaches. We should therefore allow the community to perform their analysis with the tools they are already familiar with or with the tools that are well documented and widely used in the data mining community (i.e., scikit-learn with Jupyter notebooks). In order to maximize the accuracy of data-driven models, data fusion techniques (like we have described in this paper) will be of the utmost importance. Weather plays an important role in the modeling of water-related phenomena. An efficient generic way to input weather data into the models is needed. The weather data are similar (in structure as well as in volume) to the remote sensing data from Sentinel and LandSat systems. The water management community (as well agriculture or others from the environmental domain) should take advantage of efficient technologies for exploiting earth observation data that has been evolving fast in recent years.

Finally, all this knowledge will have to be integrated under a standardized framework which will enable an efficient exchange and processing of large amounts of high-velocity data streams enriched with appropriate contextual data. An effective provisioning system for deployment of models is needed in order for the framework to take full advantage of parallel processing and to successfully deploy the system within a large sensor network. We have depicted the foundation for such a system, but efficient management of the analytics components still remains a challenge.

Author Contributions: Conceptualization, K.K.; methodology, K.K.; software, K.K., M.S., and N.M.; validation, N.M. and P.P.; data curation, K.K.; writing—original draft preparation, K.K. and N.M.; writing—review and editing, P.P., M.S.; visualization, M.S., K.K.; project administration, K.K., N.M. and P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research has received funding from: (i) the EU Horizon 2020 Programme project Water4Cities under grant agreement number 734409, (ii) the EU Horizon 2020 Programme project NAIADES under grant agreement number 820985, and (iii) the PRIMA Foundation project MAGO with grant number 2022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data supporting reported results can be found in Slovenian Environment Agency public groundwater archive available at http://vode.arso.gov.si/hidarhiv/pod_arhiv_tab.php (accessed on 1 January 2022). The con-crete dataset used for experiments (including groundwater level data and weather data) was also published at https://researchgate.net/publication/336239471_Slovenia_-_groundwater_levels (accessed on 1 January 2022) [35].

Acknowledgments: The authors would like to thank partners and colleagues from H2020 Water4Cities project for their contributions in the phase of preparation and writing of this paper. We would like to thank Stamatia Rizou and Anja Polajnar, for project leadership and steering of the work, Kristina Klemen, for insights into the needs of the urban planning community for the solutions provided by data-mining approach, Filip Koprivec and Matej Ćerin, for developing subcomponents of the WMAP system, and finally Dimitris Kofinas and Chrysi Laspidou for providing significant insights into the water management domain.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
CRISP-DM	Cross–industry Standard Process for Data Mining
HTTP	Hyper–Text Transfer Protocol
ICT	Information and Communication Technology
IoT	Internet of Things
ISDI	IoT Streaming Data Integration
IUWM	Integrated Urban Water Management
MOA	Massive Online Analysis
SQL	Structured (English) Query Language
SWM	Smart Water Management
WMAP	Water Management Analytical Platform

References

1. Laspidou, C. ICT and stakeholder participation for improved urban water management in the cities of the future. *Water Util. J.* **2014**, *8*, 79–85.
2. Cominola, A.; Giuliani, M.; Piga, D.; Castelletti, A.; Rizzoli, A. Benefits and challenges of using smart meters for advancing residential water demand modeling and management: A review. *Environ. Model. Softw.* **2015**, *72*, 198–214. [CrossRef]
3. Ioannou, A.E.; Laspidou, C.S. The Water-Energy Nexus at City Level: The Case Study of Skiathos. *Proceedings* **2018**, *2*, 694. [CrossRef]
4. Yang, L.; Yang, S.H.; Magiera, E.; Froelich, W.; Jach, T.; Laspidou, C. Domestic water consumption monitoring and behavior intervention by employing the internet of things technologies. *Procedia Comput. Sci.* **2017**, *111*, 367–375. [CrossRef]
5. Rizou, S.; Kenda, K.; Kofinas, D.; Mellios, N.; Pergar, P.; Ritsos, P.D.; Vardakas, J.; Kalaboukas, K.; Laspidou, C.; Senožetnik, M.; et al. Water4Cities: An ICT Platform Enabling Holistic Surface Water and Groundwater Management for Sustainable Cities. *Proceedings* **2018**, *2*, 695. [CrossRef]
6. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache spark: A unified engine for big data processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
7. Marz, N.; Warren, J. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*, 1st ed.; Manning Publications Co.: Greenwich, CT, USA, 2015.
8. Ta-Shma, P.; Akbar, A.; Gerson-Golan, G.; Hadash, G.; Carrez, F.; Moessner, K. An Ingestion and Analytics Architecture for IoT Applied to Smart City Use Cases. *IEEE Internet Things J.* **2018**, *5*, 765–774. [CrossRef]
9. Aggarwal, C.C. *Data Streams: Models and Algorithms (Advances in Database Systems)*; Springer: Secaucus, NJ, USA, 2006.

10. Krempl, G.; Žliobaite, I.; Brzeziński, D.; Hüllermeier, E.; Last, M.; Lemaire, V.; Noack, T.; Shaker, A.; Sievi, S.; Spiliopoulou, M.; et al. Open challenges for data stream mining research. *ACM SIGKDD Explor. Newsl.* **2014**, *16*, 1–10. [CrossRef]
11. Huang, G.B.; Chen, L.; Siew, C.K. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans. Neural Netw.* **2006**, *17*, 879–892. [CrossRef]
12. Ikonomovska, E.; Gama, J.; Džeroski, S. Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing* **2015**, *150*, 458–470. [CrossRef]
13. Zhang, Q.; Yang, L.T.; Chen, Z.; Li, P. A survey on deep learning for big data. *Inf. Fusion* **2018**, *42*, 146–157. [CrossRef]
14. Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. MOA: Massive Online Analysis. *J. Mach. Learn. Res.* **2010**, *11*, 1601–1604.
15. Montiel, J.; Read, J.; Bifet, A.; Abdessalem, T. Scikit-Multiflow: A Multi-output Streaming Framework. *CoRR* **2018**, *19*, 1–5.
16. QMiner: Data Analytics Platform for Processing Streams of Structured and Unstructured Data. Available online: https://www.researchgate.net/publication/269100309_QMiner_Data_Analytics_Platform_for_Processing_Streams_of_Structured_and_Unstructured_Data (accessed on 1 January 2022).
17. Kenda, K.; Kažič, B.; Novak, E.; Mladenić, D. Streaming Data Fusion for the Internet of Things. *Sensors* **2019**, *19*, 1955. [CrossRef]
18. Tu, D.Q.; Kayes, A.; Rahayu, W.; Nguyen, K. *ISDI: A New Window-Based Framework for Integrating IoT Streaming Data from Multiple Sources*; Springer: Heidelberg, Germany, 2019; Volume 926, pp. 498–511.
19. Manes, C.L.; Laspidou, C. Biosensors for Aquaculture and Food Safety. In *Challenges and Innovations in Ocean In Situ Sensors: Measuring Inner Ocean Processes and Health in the Digital Age*, 1st ed.; Delory, E., Pearlman, J., Eds.; Elsevier: Amsterdam, The Netherlands, 2018.
20. Popović, T.; Latinović, N.; Pešić, A.; Zečević, Ž.; Krstajić, B.; Djukanović, S. Architecting an IoT-enabled platform for precision agriculture and ecological monitoring: A case study. *Comput. Electron. Agric.* **2017**, *140*, 255–265. [CrossRef]
21. Kossieris, P.; Kozanis, S.; Hashmi, A.; Katsiri, E.; Vamvakeridou-Lyroudia, L.; Farmani, R.; Makropoulos, C.; Savic, D. A Web-based Platform for Water Efficient Households. *Procedia Eng.* **2014**, *89*, 1128–1135. [CrossRef]
22. Catchment Hydrology Explorer for Water Stewards (CatchX Platform). Available online: https://ui.adsabs.harvard.edu/abs/2018EGUGA..20.9882A/abstract (accessed on 1 January 2022).
23. Gaffoor, Z.; Pietersen, K.; Jovanovic, N.; Bagula, A.; Kanyerere, T. Big data analytics and its role to support groundwater management in the Southern African development community. *Water* **2020**, *12*, 2796. [CrossRef]
24. Laspidou, C.; Kofinas, D.; Mellios, N.; Latinopoulos, D.; Papadimitriou, T. Investigation of factors affecting the trophic state of a shallow Mediterranean reconstructed lake. *Ecol. Eng.* **2017**, *103*, 154–163. [CrossRef]
25. Mellios, N.; Moe, S.J.; Laspidou, C. Machine Learning Approaches for Predicting Health Risk of Cyanobacterial Blooms in Northern European Lakes. *Water* **2020**, *12*, 1191. [CrossRef]
26. Kofinas, D.; Mellios, N.; Papageorgiou, E.; Laspidou, C. Urban water demand forecasting for the island of Skiathos. *Procedia Eng.* **2014**, *89*, 1023–1030. [CrossRef]
27. Parkinson, J.N.; Tucci, C.; Goldenfum, J.A. (Eds.) *Integrated Urban Water Management: Humid Tropics: UNESCO-IHP*; CRC Press: Boca Raton, FL, USA, 2010.
28. Oregi, X.; Roth, E.; Alsema, E.; van Ginkel, M.; Struik, D. Use of ICT tools for integration of energy in urban planning projects. *Energy Procedia* **2015**, *83*, 157–166. [CrossRef]
29. Bricker, S.; Banks, V.; Galik, G.; Tapete, D.; Jones, R. Accounting for groundwater in future city visions. *Land Use Policy* **2017**, *69*, 618–630. [CrossRef]
30. Senožetnik, M.; Herga, Z.; Šubic, T.; Bradeško, L.; Kenda, K.; Klemen, K.; Pergar, P.; Mladenić, D. IoT Middleware for Water Management. *Proceedings* **2018**, *2*, 696. [CrossRef]
31. Pereira, D.A.; de Morais, W.O.; de Freitas, E.P. NoSQL real–time database performance comparison. *Int. J. Parallel Emergent Distrib. Syst.* **2018**, *33*, 144–156. [CrossRef]
32. Shearer, C. The CRISP–DM model: The new blueprint for data mining. *J. Data Warehous.* **2000**, *5*, 13–22.
33. Kenda, K.; Mladenić, D. Autonomous Sensor Data Cleaning in Stream Mining Setting. *Bus. Syst. Res. J.* **2018**, *9*, 69–79. [CrossRef]
34. Kenda, K.; Koprivec, F.; Mladenić, D. Optimal Missing Value Estimation Algorithm for Groundwater Levels. *Proceedings* **2018**, *2*, 698. [CrossRef]
35. Groundwater Levels for Slovenia–Data Set. Available online: https://researchgate.net/publication/336239471_Slovenia_-_groundwater_levels (accessed on 1 January 2022).
36. Andjelov, M.; Frantar, P.; Mikulič, Z.; Pavlič, U.; Savić, V.; Souvent, P.; Uhan, J. Groundwater quantitative status assessment for River Basin Management Plan 2015-2021 in Slovenia. *Geologija* **2016**, *59*, 205–219. [CrossRef]
37. Wu, X.; Zhu, X.; Wu, G.; Ding, W. Data mining with big data. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 97–107.
38. Kenda, K.; Ćerin, M.; Bogataj, M.; Senožetnik, M.; Klemen, K.; Pergar, P.; Laspidou, C.; Mladenić, D. Groundwater Modeling with Machine Learning Techniques: Ljubljana polje Aquifer. *Proceedings* **2018**, *2*, 697. [CrossRef]
39. Koprivec, F.; Kenda, K.; Šircelj, B. FASTENER Feature Selection for Inference from Earth Observation Data. *Entropy* **2020**, *22*, 1198. [CrossRef] [PubMed]

5.2 Incremental Learning

This chapter presents the paper titled *Usage of statistical modeling techniques in surface and groundwater level prediction* by Klemen Kenda, Jože Peternelj, Nikos Mellios, Dimitris Kofinas, Matej Čerin, and Jože Rožanec. This paper has been published in the Journal of Water Supply: Research and Technology-AQUA (IF 2020: 1.644) [20]. Klemen Kenda contributed significantly to conceptualization and methodology development presented in this paper. He contributed to software development, design, and implementation of evaluation. He led the writing of the paper and also contributed to visualizations.

The lambda architecture, combined with the streaming sensor fusion module, can create feature vectors needed for machine learning methodologies that include a number of contextual and derived features from all available streaming data sources. Moreover, the system can generate the feature vectors needed for inference and also for learning (equipped with labels) on the fly. In such a system, inherently, incremental learning approaches can be utilized with ease.

The presented paper evaluates more than 20 algorithms (including five incremental algorithms) in two predictive tasks in the water sector. We built solutions to predict the water levels of surface and groundwater levels. The main finding of the paper is that (at least for the two problems analyzed) batch models are superior to incremental ones. R^2 scores of the batch models in the prediction of surface and ground water levels for one to five days ahead are $50 - 100\%$ higher than the scores of the incremental algorithms. Incremental models are still good enough in certain scenarios and could be used in certain large-scale operations. We also discovered that by converting a regressive modeling task into a multi-class classification task, we can effectively identify sudden changes in the time series.

248 © 2020 The Authors Journal of Water Supply: Research and Technology—AQUA | 69.3 | 2020

Usage of statistical modeling techniques in surface and groundwater level prediction

Klemen Kenda, Jože Peternelj, Nikos Mellios, Dimitris Kofinas, Matej Čerin and Jože Rožanec

ABSTRACT

The paper presents a thorough evaluation of the performance of different statistical modeling techniques in ground- and surface-level prediction scenarios as well as some aspects of the application of data-driven modeling in practice (feature generation, feature selection, heterogeneous data fusion, hyperparameter tuning, and model evaluation). Twenty-one different regression and classification techniques were tested. The results reveal that batch regression techniques are superior to incremental techniques in terms of accuracy and that among them gradient boosting, random forest and linear regression perform best. On the other hand, introduced incremental models are cheaper to build and update and could still yield good enough results for certain large-scale applications.

Key words | data-driven modeling, groundwater level modeling, incremental learning, stream mining, surface water level modeling

Klemen Kenda (corresponding author)
Jože Peternelj
Matej Čerin
Jože Rožanec
Jožef Stefan Institute,
Jamova cesta 39, 1000 Ljubljana,
Slovenia
E-mail: *klemen.kenda@ijs.si*

Klemen Kenda
Matej Čerin
Jože Rožanec
Jožef Stefan International Postgraduate School,
Jamova cesta 39, 1000 Ljubljana,
Slovenia

Nikos Mellios
Dimitris Kofinas
Department of Civil Engineering,
University of Thessaly,
38334 Volos,
Greece

INTRODUCTION

Water data are becoming increasingly accessible and low-cost. Investments in improvement of data acquisition and data transfers have enabled significant growth of knowledge-intensive economies (Washburn *et al.* 2010; Chourabi *et al.* 2012; Di Nardo *et al.* 2015a). However, there is still a great deal of room for improvement, especially compared to the energy or transportation sectors, as indicated by the expected future infrastructure costs by the sector (Laspidou 2014). On the contrary, water as a resource itself is becoming a more expensive commodity. Water utilities worldwide are incorporating – or have already incorporated – the opportunity costs of capital, operation, maintenance, and environmental impacts to the final price under the Polluter-Pays and the User-Pays principles, commonly accepted by the OECD countries (Rogers *et al.* 2002). Digitalization has penetrated most areas of human activity, including major manufacturing facilities, energy markets, health care, and even well-being, while various methodologies on improving resources management and optimizing consumption, usage or exploitation systems have been tested in various settings producing positive results (UNEP 2013). Water management digitalization process is showing great potential for the usage of modern technologies such as the Internet of Things (IoT) and Artificial Intelligence (AI). The latter can operate as a catalyst for investigating, understanding, forecasting, and optimizing water usage, leakage, fraud, and pollutant detection, flooding and damage prevention and protection (Di Nardo *et al.* 2015b).

AI methodologies, especially statistical modeling techniques from the family of machine learning (ML)

doi: 10.2166/aqua.2020.143

algorithms, have proven to successfully complement or even replace the traditional process-based models, usually, requiring much less data preparation and computing time; they are therefore more easily implemented in real-world scenarios (Adamowski *et al.* 2012; Kofinas *et al.* 2014; Tiwari & Adamowski 2014; Mellios *et al.* 2015). ML techniques have also proven to effectively catch patterns that involve complex interdependencies and non-linearities such as those found in the interdisciplinary boundaries of aquatic systems and ecological systems (Recknagel 2001). On the other hand, process-based models are more generalized and provide results that can be applied in broader areas, whereas ML models usually target a specific point in space. One should note that ML models are completely data-driven. This means that expert knowledge is required only to select and transform relevant data and their derivatives into meaningful inputs; and for the conclusive phase of evaluating, validating, and interpreting the outputs. Underlying processes are agnostically modeled by the ML methods. This can be perceived as a benefit or as a drawback, since on the one hand, it constitutes the modeling process practically easier, on the other hand, it deprives some of the interpreting function of building up the causal-effect relations (Sarle 1994; Krause *et al.* 2016). The above reveals a great potential (research and practical) and value for combining ML with process models in a similar way to 'injecting humans-in-the-loop' when modeling with ML, either by interactive training or interactive feature selection (Krause *et al.* 2014, 2016; Amershi *et al.* 2015).

During the last decades, scientific literature is overwhelmed by laboratory tests of ML methodologies, as the aforementioned benefits of such techniques have attracted researchers (Maier & Dandy 2000). Often, data preprocessing (including cleaning and data fusion) in such applications is done manually offline. However, transferring the ML applications to real-world scenarios would require automated data processing pipeline from the data source (sensor or web resource, e.g. for weather and weather forecast data) to the final user.

IoT and other sensor data itself are useful and can be applied to many scenarios, however – much better results can be obtained when using multiple interconnected data sources (Manyika *et al.* 2015). For example, predicting groundwater levels in the future will benefit significantly from weather and weather forecast data and perhaps in some special cases from the water withdrawal and population modeling data. This means that multiple heterogeneous data sources have to be combined in real time to achieve the best possible results.

This work, which extends (Kenda *et al.* 2018a), presents a comprehensive survey of standard statistical modeling techniques (ML models) on the use-cases of groundwater and surface water level prediction. To the best of our knowledge, in this paper, another family of ML techniques is introduced to water domain: incremental learning (sometimes referred to also as stream mining). This is specifically suitable for learning from continually generated sensor data as the models are updated with each subsequent measurement without extensive use of computer resources as in traditional batch methods, which require re-learning on the whole historical dataset. Savings are obtained in data storage, computing power and time.

Slovenia is a country with a dense hydrographical network, with a great difference in the amount of precipitation between areas in the east and the west, with areas of regular or occasional flooding or drought and a positive balance between incoming and outcoming waters. The population density and its related pressure on the aquatic environment also differ. Water landscape is affected by the anticipated climate change (Kanakoudis *et al.* 2016), which is causing longer lasting spring and summer droughts as well as less and more imbalanced precipitations. Despite an overall favorable water balance, shortages can be expected in 15% of country's surface area, mostly in the north-eastern part (https://www.arso.gov.si/en/soer/freshwater.html), as well as floods on another 15% of the territory. Since 1992, seven summer droughts have hit agriculture. In 2003, 2.4% of population required water to be supplied with a tanker.

Accurate groundwater and surface water level short-term predictions allow to better understand its dynamics and convey information about coming extreme events; identify factors that affect water consumption as well as optimize operating schedules of related infrastructure (Adamowski *et al.* 2012). This information also allows to better plan in a context of greater water scarcity (Griffin & Chang 1990) and manages the resource in new ways (e.g. by establishing dynamic pricing (Arbués *et al.* 2003)) in order to increase sustainability.

This analysis aims to introduce a handful tool for groundwater level assessment and forecasting. Such a tool could be used for short-term and mid-term water management. Regarding short-term forecasting, it is crucial to foresee on time a coming extreme event, such as a flood or a shortage crisis. These two extremes may cause a series of malfunctions that create problems in terms of well-being. A shortage crisis, when the aquifer level is too low, affects the urban water supply if the aquifer is used as a reservoir. Regarding mid-term forecasting, such a tool can facilitate the optimal planning of water resources management especially when there are conflicting needs and the potential of multiple sources. End-users of the results or of a product tool of this analysis could be a municipality, a water utility, urban designers, industrial, and agricultural sectors, who could benefit by the easiness of such black-box applications that do not require specific, advanced hydrological knowledge.

The main contributions of this paper are as follows:

1. Comparison of 21 different statistical modeling techniques applied to surface and groundwater levels forecasting at different time horizons.
2. Introduction of incremental learning techniques to modeling of surface and groundwater levels.
3. Usage of efficient and automatic feature selection techniques in surface and groundwater modeling.

METHODS

Statistical modeling techniques

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

Machine learning is a subfield of the wider AI field. The discipline has been blooming since mid-1970s and has provided widely used solutions such as ML translation, typing assistant, spam mail identification, and image recognition (Hastie *et al.* 2009).

In water management, ML has been used for predicting various key variables in water systems such as groundwater levels (Nie *et al.* 2016; Jeong & Park 2019), urban water demand in multiple scales from household to residential (Al-Qunaibet & Johnston 1985; Griffin & Chang 1990; Oyebode 2019), urban water consumption behavior (Ioannou *et al.* 2017), anomaly detection, such as fraud incidents (García Valverde *et al.* 2015; Candelieri 2017), leakage in water distribution networks (Di Nardo *et al.* 2015a, 2015b), and stratification in reservoirs (Soleimani *et al.* 2019). Often ML research in water management is focused on a particular method, the selection of which is not necessarily justified. There are few research papers, which really investigate a wider variety of algorithms and even among these, very few address the fact, that the usage of a particular modeling algorithm does not influence the final results as much as the appropriate use of contextual data (Hastie *et al.* 2009) and that ensuring proper feature generation and data fusion in real-time (in live, real-world systems) is a great challenge until today (Kenda *et al.* 2019).

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

The most relevant and widely used methods in environmental data-driven modeling nowadays are random forest (Hastie *et al.* 2009), gradient boosting (Friedman 2002), and deep learning (Goodfellow *et al.* 2016), which are

based on simpler building blocks like decision trees and perceptrons (Hastie *et al.* 2009). It should be noted that water management modeling often assumes regression problems, for which deep learning does not exhibit as much power as with other problems (e.g. image recognition, text translation, and similar). According to the nature of a dynamic water system, also linear models like linear regression and support vector machines (SVM classifier or SVC and SVM regressor or SVR) with linear kernel (Hastie *et al.* 2009) can achieve very good results while preserving low-computational cost. Quite often, very simple methods like k-nearest neighbors (Hastie *et al.* 2009) can yield good results.

Incremental learning techniques

Traditional statistical modeling techniques use batches of data points to learn. If the observed system is prone to concept drift (Gama *et al.* 2014), which means that the distribution of the target value is changing through time due to changes in the user behavior or in the environment, frequent re-learning of the models is needed, which is time-consuming (e.g. repeating of model learning step after each day or week). Incremental learning techniques (Bifet 2010) are able to update the existing models. The model, that has been taught on a set of learning examples, can be updated with the next one alone. The update itself is much cheaper than re-learning, and often the incremental learning techniques are aware of the concept drift (e.g. when user behavior is changed to a previously unseen mode due to some external reason and this influences the underlying model), which means that they are able to adapt to the change in the new systems behavior much faster.

Some of the tested methods were the streaming perceptron, Hoeffding trees (Domingos & Hulten 2000), and Hoeffding adaptive trees – HAT (Bifet & Gavaldà 2009). Other methods include recursive linear regression, model trees like FIMT-DD (Ikonomovska *et al.* 2015), incrementally learned neural networks (Zhang *et al.* 2018), and incrementally learned SVMs based on stochastic gradient descent (Bottou 2010). Algorithms like decision trees can be used in ensembles (e.g. bagging), where each tree in the ensemble is fed with a subset of input data (Oza 2005). For classification problems, which are rarer in the water management domain, there are many more methods available; however, there are still not many effective implementations of the state-of-the-art methods.

RESULTS AND DISCUSSION

Experiments were conducted on two different datasets: groundwater and surface water levels in Slovenia (see Table 1). Both cases represent typical regression problems where the modeling task is to predict water level for a certain time period (or prediction horizon) in the future.

Initial experiments have shown that predicting absolute water levels is problematic because the system itself is cumulative. Therefore, reducing the prediction problem to the prediction of daily level differences and not the water levels themselves has proven a good approach. Absolute water levels are finally calculated by the addition/subtraction of predictions from a historical absolute true value. Figure 1 presents the experimental workflow. Green boxes represent data retrieval, blue data manipulation, orange modeling tasks, and yellow model evaluation and results (please refer to the online version of this paper to see this figure in color: http://dx.doi.org/10.2166/aqua.2020.143).

Data

Groundwater and surface water data (see Table 1) have been acquired from an online repository (http://vode.arso.gov.si/hidarhiv/pov_arhiv_tab.php) at the Slovenian environment agency (ARSO). Weather data have been retrieved from DarkSky (https://darksky.net/) web service and ARSO historical weather data repository (http://meteo.arso.gov.si/met/sl/archive/). Sensor data have been thoroughly inspected and only the sensors with data in the period from 2010 to (including) 2017 were selected, for which accurate weather could be retrieved as well. Weather data related to underground water modeling have been retrieved

Table 1 | Experimental datasets include 24 time series

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

Two for groundwater levels in the Ljubljana region and 22 for surface water levels in Slovenia.

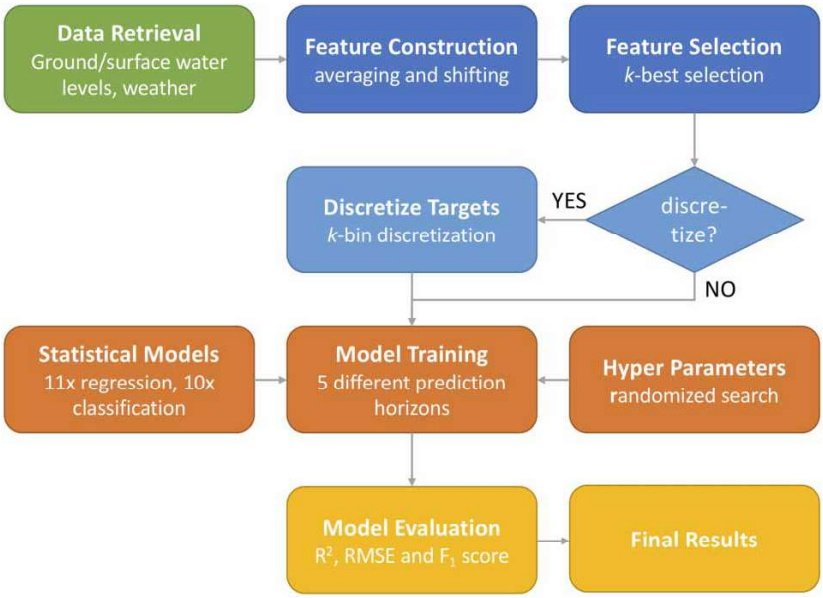


Figure 1 | The workflow of the data-driven approach includes data acquisition, feature generation and selection, modeling and evaluation tasks.

for numerous spatial points in the aquifer, based on the locations of the target sensors, while the weather data for surface water modeling have been retrieved from a single location in the vicinity of the water spring for each selected watercourse.

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

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

Feature construction and selection

Statistical models require not only the raw data but also derived features, which reflect a certain physical process that are influencing water level changes. When building additional features, the raw hourly weather forecast data were used, consisting of precipitation probability, precipitation intensity, precipitation type, temperature, cloud cover, dew point, humidity, pressure, and daytime. From these, daily averages, minima and maxima were calculated, producing 24 distinct features which were then analyzed with a correlation matrix (see Figure 2).

A correlation matrix can be used in two ways. Firstly, the correlations of particular attributes with the target variable can be read (water level daily change), and the most correlated attributes can be selected to be used in the models. Secondly, it can be used for filtering of highly correlated attributes. Highly correlated attributes will not bring additional knowledge to the model and might worsen model accuracy.

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

The remaining 18 initial features were used to construct additional derivatives by introducing time delays (shifts) and averages over multiple past days. The idea behind this comes from the intuition that groundwater and surface

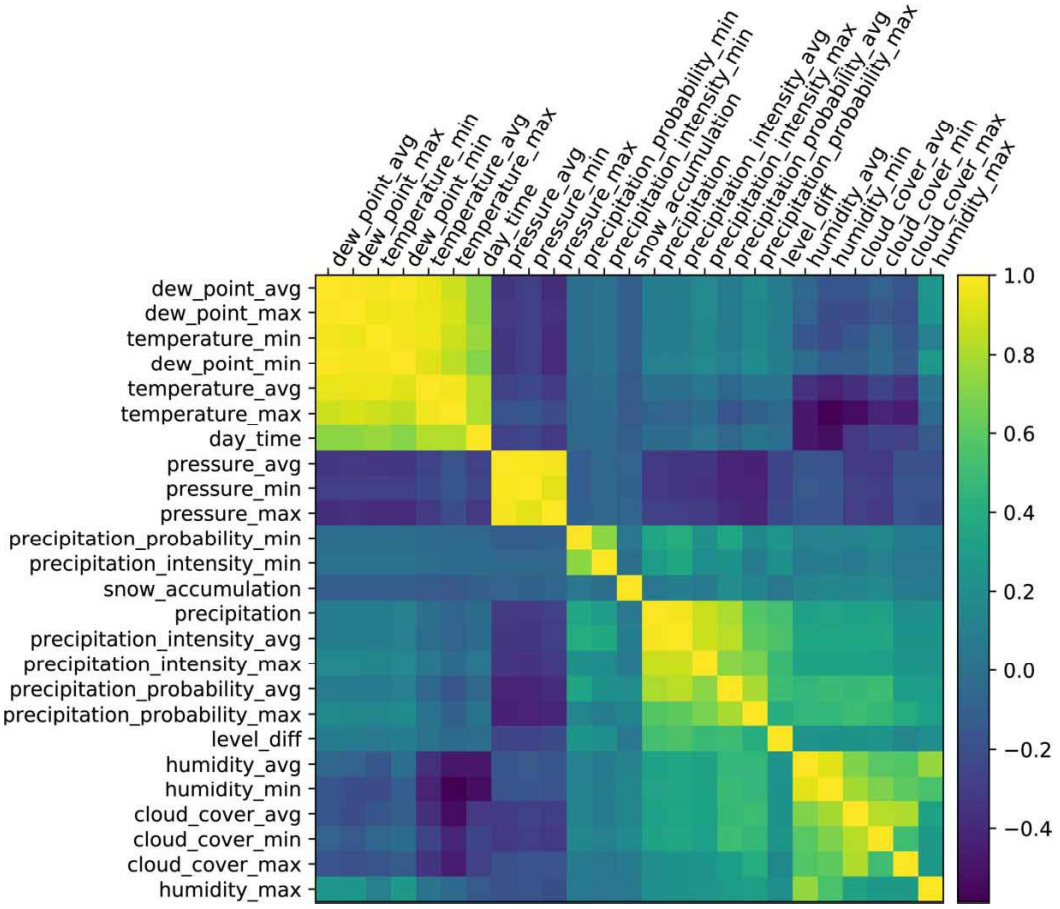


Figure 2 | Correlation matrix of the target value (level_diff) and 24 base features calculated from hourly weather forecasts.

water level responses to the weather changes have some hysteresis.

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

After the addition of all the feature derivatives, a total of 3,890 features were available. With approximately 3,000 samples of water level measurements in the datasets and the number of features in the same order of magnitude, further steps to reduce the size of the feature vectors were necessary to avoid overfitting. Feature selection (Liu & Motoda 2007) can significantly reduce the number of features without sacrificing the expressivity of the model. In water-related scenarios, one often encounters problems, where the number of measurements of a time series that is modeled is not very high (e.g. one measurement per day in a period of a couple of years). Many methods exist that provide efficient feature selection.

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

following equation:

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

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

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

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

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

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

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

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

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

Secondly, the correlation is converted to F-score as defined in the following equation:

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

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

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

Evaluation

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

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

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

Note: R^2 is calculated for level differences, which is a rapidly changing time series. This means that the R^2 for

the actual water level is much higher. For example, the highest coefficient of determination for surface water levels was 0.572 for level differences, while for actual surface water levels, it was above 0.93.

Experiments

The evaluation of the experimental results is divided into two sections. The first section is dedicated to the modeling of groundwater levels and the second to the modeling of surface water levels. Each section presents the following results: illustrative prediction results for water level differences and for water levels with multiple prediction horizons, comparison of the accuracy of different methods, and qualitative results on the importance of model features that can be further analyzed by domain experts.

Groundwater level modeling

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

Illustrative results for the prediction of level differences are shown in Figure 3, and cumulative results for water levels are shown in Figure 4. The overall experimental results are listed in Table 2 and depicted in Figure 5.

Figure 3 depicts the actual results of the prediction experiments. Level differences are calculated for 3 days in advance. The statistical model can predict major changes in groundwater levels with fairly good accuracy (in terms of start, duration, and amplitude). The modeling results are converted into water level predictions as shown in Figure 4.

Figure 4 illustrates the characteristics of different types of statistical models for groundwater level prediction. The best regression models, the best streaming regression models, and the most illustrative classification-based

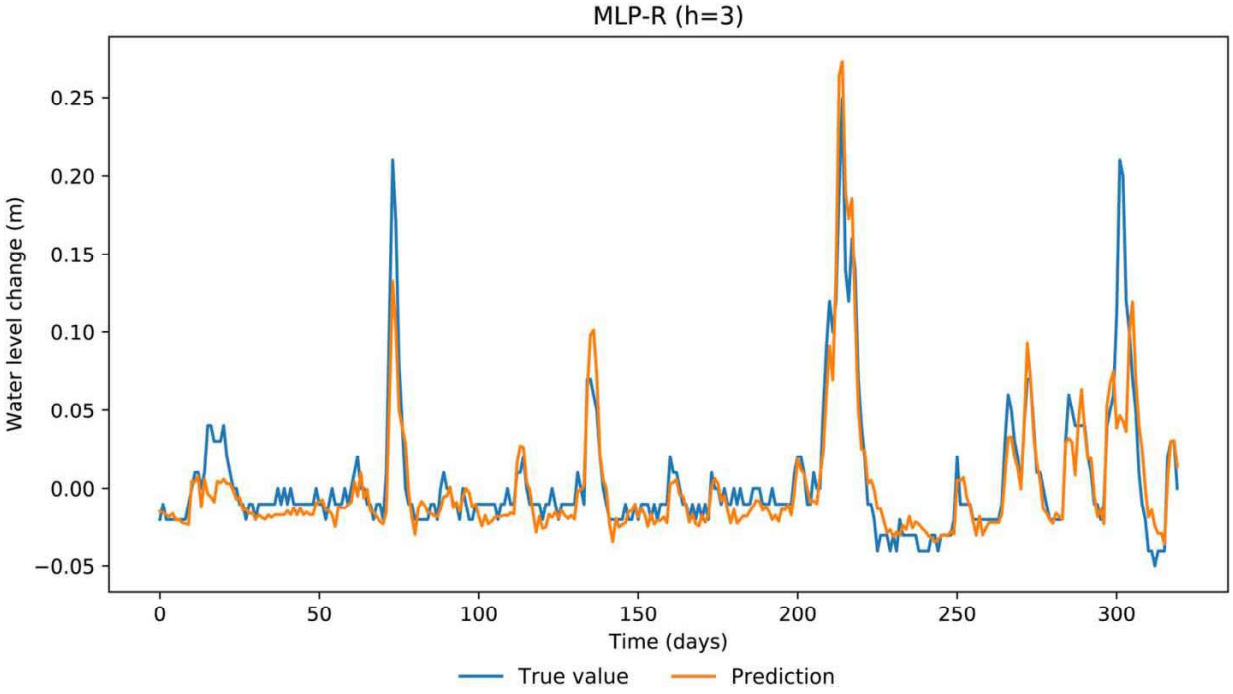


Figure 3 | Prediction of groundwater level difference (change) for MLP (multilayer perceptron neural network) regressor for 3-day prediction horizon.

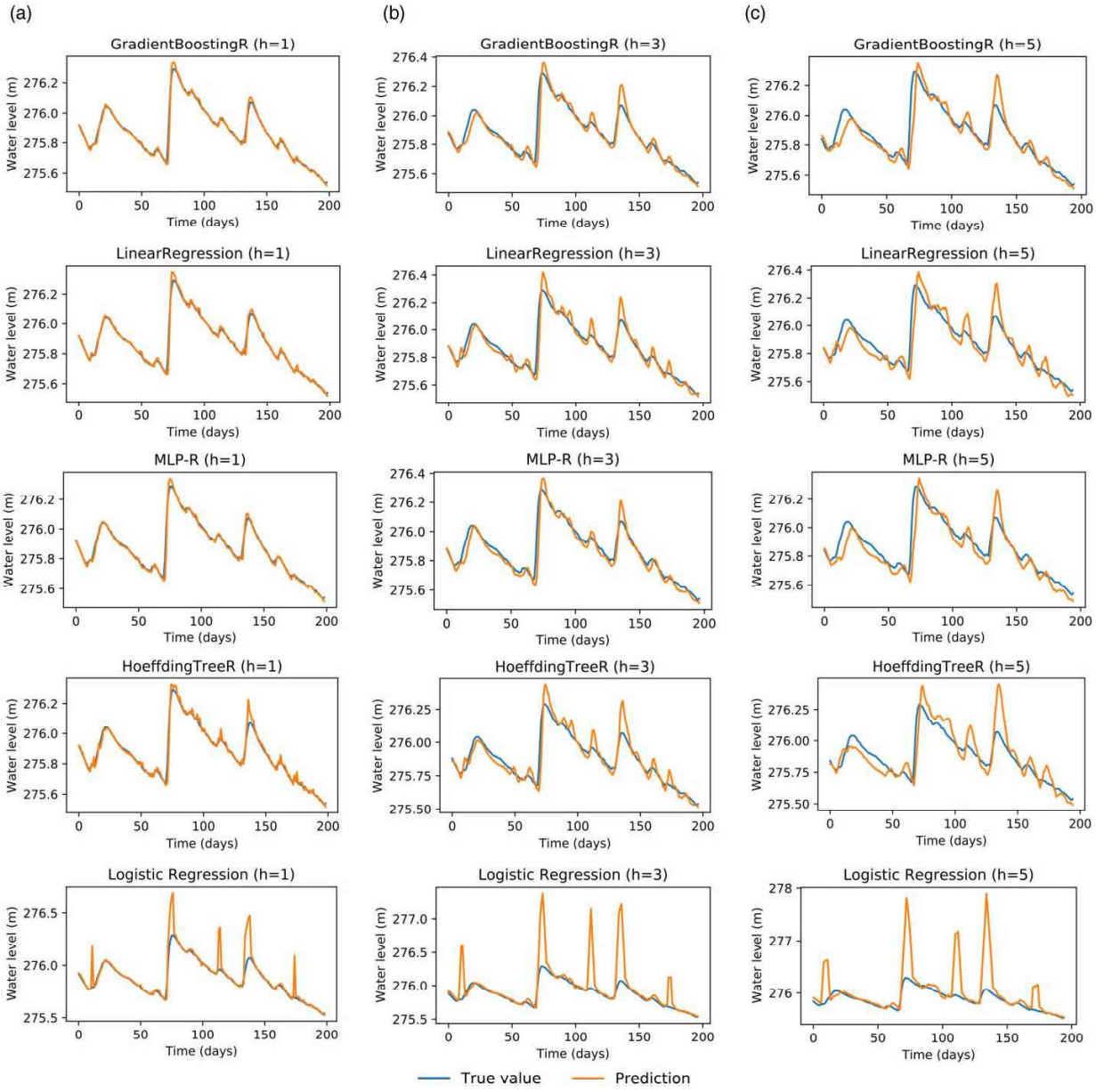


Figure 4 | Illustrative results of different prediction models for groundwater levels. Different algorithms are depicted in rows, different prediction horizons: (a) 1 day ahead, (b) 3 days ahead, and (c) 5 days ahead are shown in columns.

models are included in the figure. The majority of models overshoots larger level changes. This means that in historical data in some cases includes bigger level changes under similar conditions. This could, for example, indicate higher water withdrawal or a change in the dynamics of the groundwater system. Prediction accuracy decreases with the prediction horizon and that the models sometimes miss the beginning of a change (with longer prediction horizons). The only model that correctly records the beginning of the water level change properly even for a 5-day prediction horizon is the classification model. This means that although the accuracy of the model is poor, the classification

Table 2 | Averaged modeling results for groundwater levels for five different prediction horizons (from 1 to 5 days ahead)

	1 day ahead		2 days ahead		3 days ahead		4 days ahead		5 days ahead			
Method	R^2	σ	R^2	σ	R^2	σ	R^2	σ	R^2	σ	t_t (ms)	t_p (ms)
LinearRegression	0.834	0.037	0.708	0.063	0.661	0.066	0.638	0.063	**0.641**	0.07	1.9	0.2
DecisionTreeR	0.677	0.146	0.61	0.111	0.545	0.093	0.541	0.061	0.543	0.059	3.3	0.1
RandomForestR	**0.842**	0.031	**0.726**	0.108	**0.669**	0.08	0.644	0.067	0.600	0.054	1,047	8.5
GradientBoostingR	**0.849**	0.037	**0.775**	0.052	**0.732**	0.071	**0.690**	0.081	**0.655**	0.09	322.6	0.5
PLSRegression	0.726	0.052	0.65	0.073	0.639	0.076	0.639	0.076	0.639	0.077	2.9	<0.1
ExtraTreeR	0.677	0.146	0.61	0.111	0.545	0.093	0.541	0.061	0.543	0.059	3.3	<0.1
SVR	− 0.137	0.41	− 0.16	0.356	− 0.113	0.325	− 0.069	0.317	− 0.086	0.324	2.5	0.4
MLP-R	0.825	0.045	0.691	0.084	0.659	0.09	**0.646**	0.092	**0.641**	0.082	210.2	1.3
KNeighborsR	0.747	0.053	0.661	0.089	0.631	0.102	0.618	0.112	0.610	0.114	5.9	8.0
HoeffdingTreeR	0.495	0.164	0.513	0.127	0.518	0.144	0.493	0.139	0.482	0.13	3,347.1	28.1
HAT-R	0.506	0.176	0.513	0.127	0.518	0.144	0.493	0.139	0.482	0.13	3,722.2	28.5
LogisticRegression	0.485	0.153	0.408	0.131	0.395	0.146	0.376	0.179	0.370	0.16	83.3	0.2
DecisionTreeC	0.506	0.141	0.395	0.116	0.342	0.183	0.365	0.203	0.377	0.183	4.9	<0.1
ExtraTreeC	0.306	0.09	0.336	0.113	0.24	0.162	0.332	0.058	0.256	0.258	2.4	0.1
RandomForestC	0.554	0.108	0.489	0.178	0.481	0.19	0.498	0.176	0.470	0.186	279.8	9.8
SVC	0.522	0.088	0.413	0.131	0.375	0.158	0.391	0.193	0.400	0.191	135.4	16.9
KNeighborsC	0.530	0.071	0.378	0.143	0.387	0.201	0.357	0.157	0.353	0.16	7.8	15.8
Perceptron	0.435	0.206	0.102	0.388	− 0.007	0.788	0.325	0.291	0.266	0.268	10.5	0.2
GaussianNB	0.472	0.112	0.378	0.123	0.374	0.138	0.362	0.144	0.358	0.148	1.2	0.3
HoeffdingTreeC	0.472	0.112	0.378	0.123	0.374	0.138	0.362	0.144	0.358	0.148	1,054.2	119.5
HAT-C	0.453	0.108	0.371	0.14	0.364	0.153	0.346	0.147	0.353	0.148	1,912.2	120.4

Best results by prediction horizon are bolded. Best classification-based results are underlined.

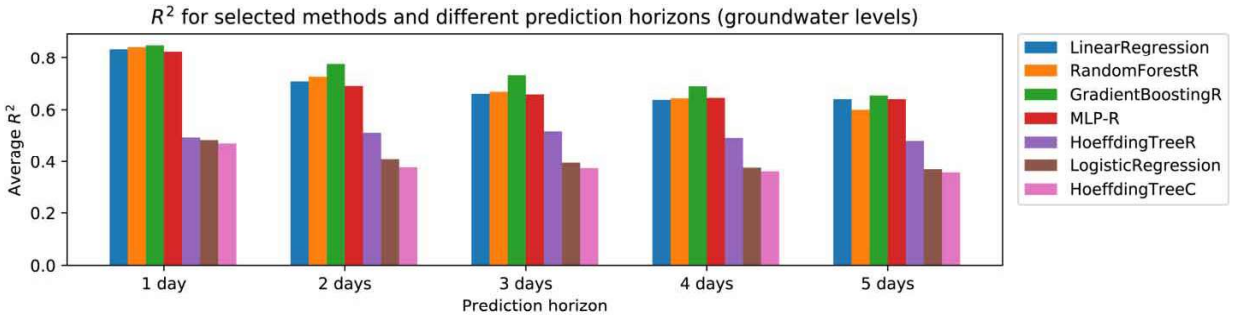


Figure 5 | R^2 of selected statistical models for different prediction horizons (groundwater levels).

could be a good choice for estimating the start of level changes. In addition, the reason for poor classification results can be seen from the figure. During the discretization, the rarely occurring high values of the level changes were grouped into a single bin (which produces a fixed amplitude), although these values are distributed over a bigger interval. For extreme values, finer discretization would lead to better results.

The modeling methods perform as expected. Traditional workhorses perform best (with R^2 scores between 0.6 and

0.85), and incremental learning methods and classifiers yield substantially worse results. Gradient boosting is significantly better than competing methods. As shown in Kenda *et al.* (2018a), linear regression is also a good choice in this setup. However, it is important to note that extensive feature engineering has helped it catch the latent dynamics of the aquifer, while gradient boosting could perform reasonably well even without so many features. The discretization of daily level differences in classes and the use of multi-class classification did not perform well. During discretization some information is lost, which is reflected in the results.

Hoeffding Tree regressor performed best among incremental learning methods. It maintains its performance even with longer prediction horizons and is more competitive in scenarios with a prediction horizon of more than 2 days. The use of incremental learners is most effective in scenarios where the distribution of target values changes over time, which is not the case for groundwater and surface water levels in Slovenia between 2010 and 2017. The methods could be much more effective in modeling the behavior of water consumers.

The main advantage of the usage of incremental learning methods is that they do not have to re-learn from all the data in the training phase. They can only be updated with the latest value and are thus computationally much more efficient. This could be taken advantage of in scenarios where many sensors with fast updates are used (e.g. modeling consumers' behavior in a large city). Batch models should be retrained regularly. The training (t_t) and prediction (t_p) times listed in Table 2 show that linear regression could be the most effective method in practice, since its training and prediction times are among the smallest, while the competitive methods like random forest, gradient boosting, and MLP require significantly more time to (re-)train.

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

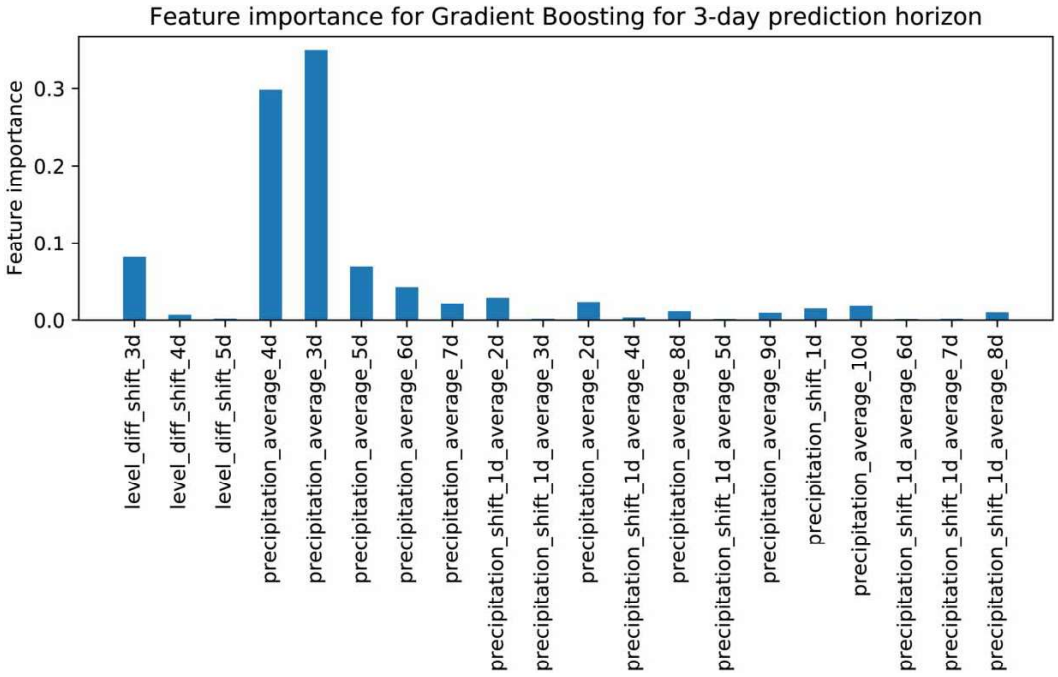


Figure 6 | Feature importance scores (relative) as given by the gradient boosting regression model for 3-day prediction horizon for groundwater level modeling.

methods and intelligent feature selection. The study (Chen *et al.* 2020) also shows that data-driven models give superior results to the traditional process-based models by a high margin (R^2 scores of data-driven models are higher by approximately 0.2).

Most ML algorithms provide a kind of score on the importance of a feature in the statistical models. In linear regression, these are coefficients, and in ensemble tree-based methods, these are importance weights. An example importance weights for a 3-day prediction horizon for gradient boosting is shown in Figure 6. The groundwater level change is influenced by the current trend (last level difference) and different values related to precipitation and its history. Among other weather phenomena, none was selected using the automatic method. The current average precipitation values are the most important ones. Some shifted features also play an important role.

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

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

Surface water level modeling

The same methodology was used for modeling the surface water levels as for groundwater. Illustrative results of surface water level differences and levels are depicted in Figures 7 and 8. The results are listed in Table 3. The selected sensor accuracy in terms of R^2 is depicted in Figure 9.

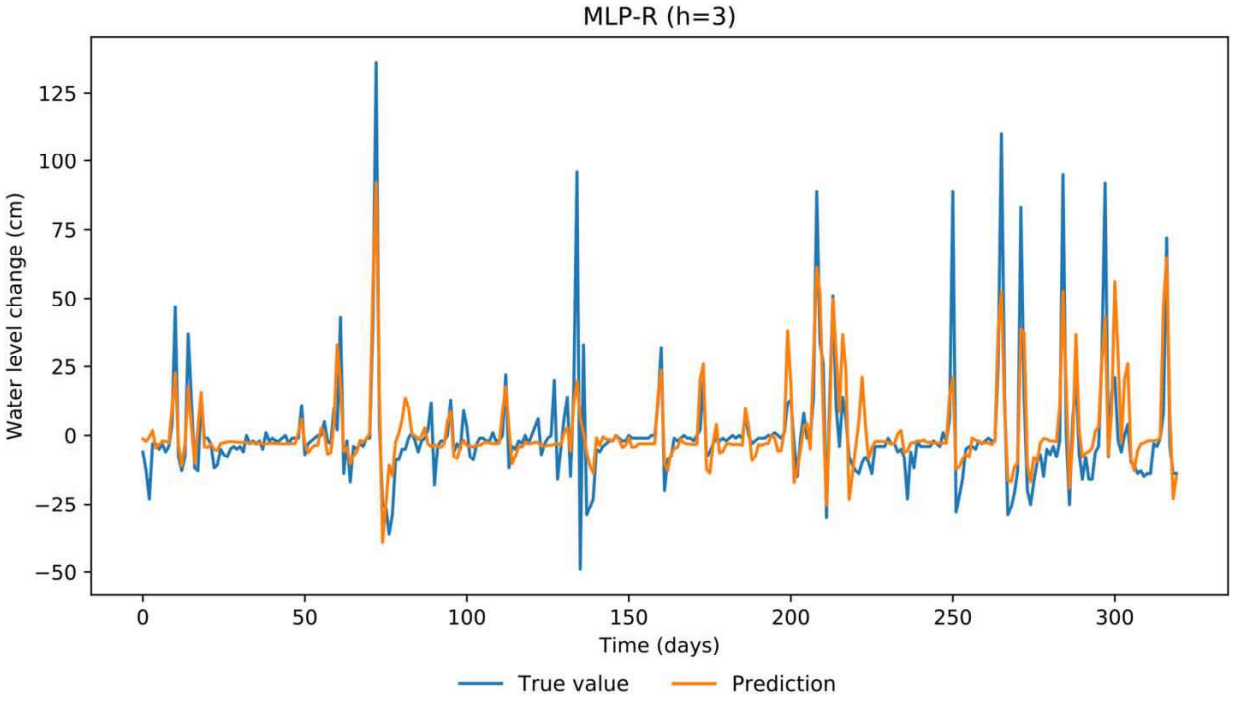


Figure 7 | Prediction of surface water level difference (change) for MLP (multilayer perceptron neural network) regressor for 3-day prediction horizon.

Figure 10 also shows the standard deviation of R^2 scores among 22 sensors for three selected methods.

Comparing Figure 7 to Figure 3 reveals larger discrepancies between the prediction (orange) and the true values (blue) (please refer to the online version of this paper to see these figures in color: http://dx.doi.org/10.2166/aqua.2020.143). For example, it can be observed that the model was unable to follow the dynamics in the real world between days 120 and 150. The fluctuations, even the large ones, are sometimes not reflected in the modeling results. This means that certain mechanisms in nature could not be modeled with the input data. The model is also conservative in estimating the high peaks in the level change. This usually means that similar situations (similar feature

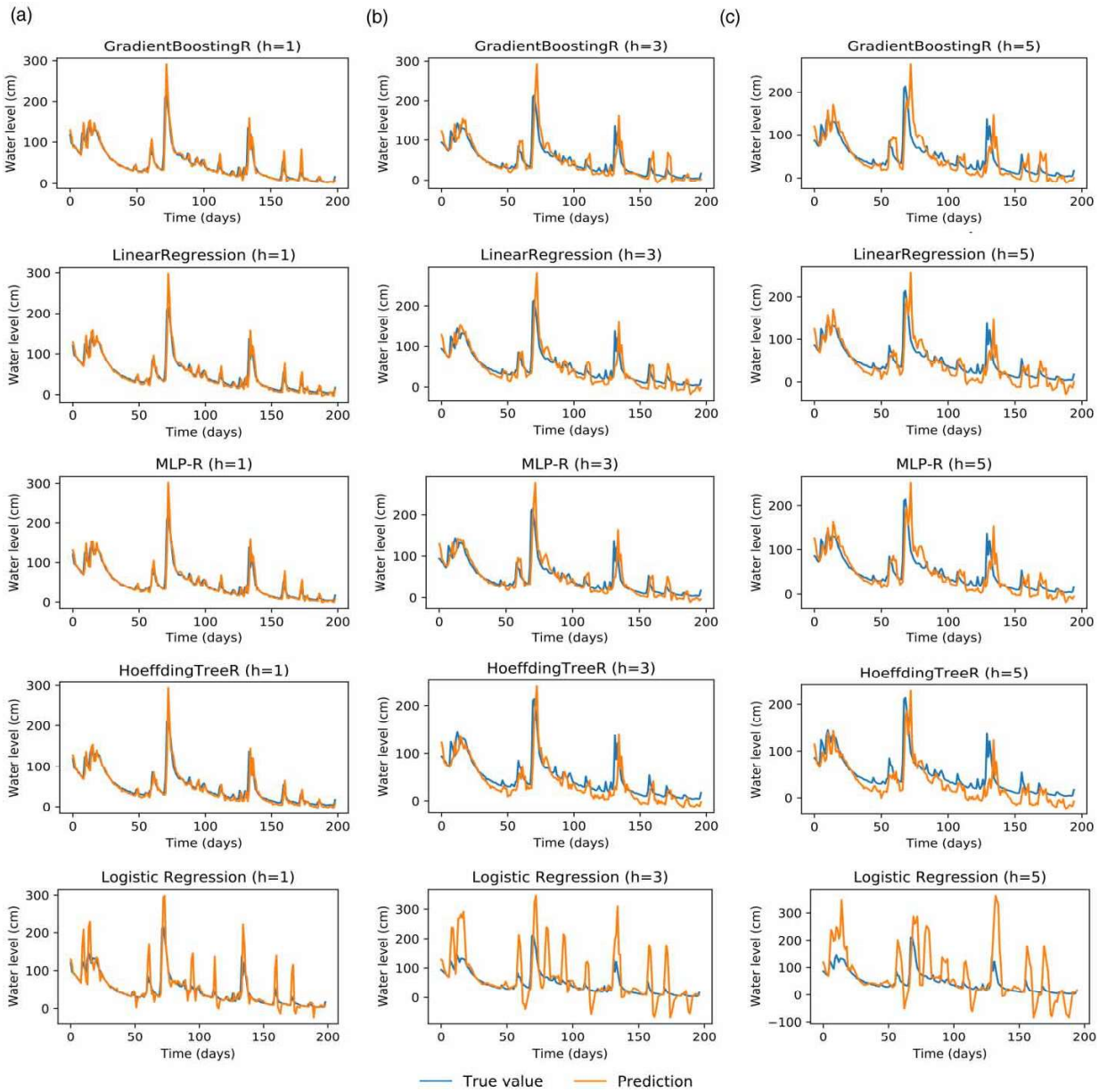


Figure 8 | Illustrative results of different prediction models for surface water levels. Different algorithms are depicted in rows, different prediction horizons: (a) 1 day ahead, (b) 3 days ahead, and (c) 5 days ahead are shown in columns.

Table 3 | Averaged modeling results for surface water levels for five different prediction horizons (from 1 to 5 days ahead)

	1 day ahead		2 days ahead		3 days ahead		4 days ahead		5 days ahead			
Method	R^2	σ	R^2	σ	R^2	σ	R^2	σ	R^2	σ	t_t (ms)	t_p (ms)
LinearRegression	0.553	0.069	0.499	0.071	0.487	0.081	0.484	0.081	0.489	0.078	3.7	0.3
DecisionTreeR	0.350	0.172	0.325	0.161	0.287	0.202	0.294	0.195	0.267	0.302	3.8	0.1
RandomForestR	**0.571**	0.106	**0.514**	0.109	**0.492**	0.109	**0.501**	0.111	**0.502**	0.1	1,194	6.4
GradientBoostingR	**0.572**	0.106	0.500	0.088	0.480	0.108	0.480	0.113	0.482	0.109	399.0	0.4
PLSRegression	0.502	0.065	0.457	0.068	0.450	0.072	0.450	0.071	0.449	0.07	3.0	0.4
ExtraTreeR	0.350	0.172	0.325	0.161	0.287	0.202	0.294	0.195	0.267	0.302	3.7	0.1
SVR	0.154	0.059	0.155	0.059	0.156	0.058	0.156	0.058	0.156	0.058	158.6	18.2
MLP-R	0.538	0.078	**0.512**	0.077	**0.511**	0.083	**0.511**	0.082	**0.509**	0.082	3,014	1.5
KNeighborsR	0.507	0.092	0.458	0.095	0.439	0.111	0.435	0.109	0.436	0.103	8.6	12.7
HoeffdingTreeR	0.312	0.212	0.246	0.203	0.279	0.178	0.284	0.181	0.282	0.18	3,748	39.8
HAT-R	0.312	0.213	0.218	0.28	0.276	0.183	0.281	0.186	0.279	0.184	4,140	40.1
LogisticRegression	0.205	0.162	0.212	0.158	0.207	0.167	0.200	0.159	0.206	0.169	110.6	0.2
DecisionTreeC	− 0.066	0.301	− 0.108	0.353	− 0.096	0.35	− 0.09	0.383	− 0.104	0.35	6.3	0.1
ExtraTreeC	− 0.189	0.447	− 0.181	0.403	− 0.163	0.371	− 0.252	0.479	− 0.192	0.455	2.1	0.1
RandomForestC	0.081	0.241	0.065	0.254	0.069	0.221	0.08	0.224	0.058	0.248	264.5	8.5
SVC	0.083	0.174	0.130	0.178	0.126	0.173	0.124	0.178	0.135	0.172	167.9	20.1
KNeighborsC	0.06	0.18	0.047	0.206	0.029	0.206	0.039	0.207	0.037	0.209	8.8	17.6
Perceptron	− 0.121	0.739	− 0.224	0.732	− 0.279	0.964	− 0.14	0.736	− 0.088	0.622	14.0	0.2
GaussianNB	0.127	0.225	0.143	0.217	0.143	0.219	0.142	0.22	0.139	0.218	1.3	0.4
HoeffdingTreeC	0.097	0.221	0.111	0.216	0.111	0.218	0.109	0.218	0.111	0.216	1,592	154.7
HAT-C	0.099	0.218	0.115	0.209	0.108	0.207	0.108	0.206	0.108	0.208	2,694	161.8

Best results by prediction horizon are bolded. Best classification-based results are underlined.

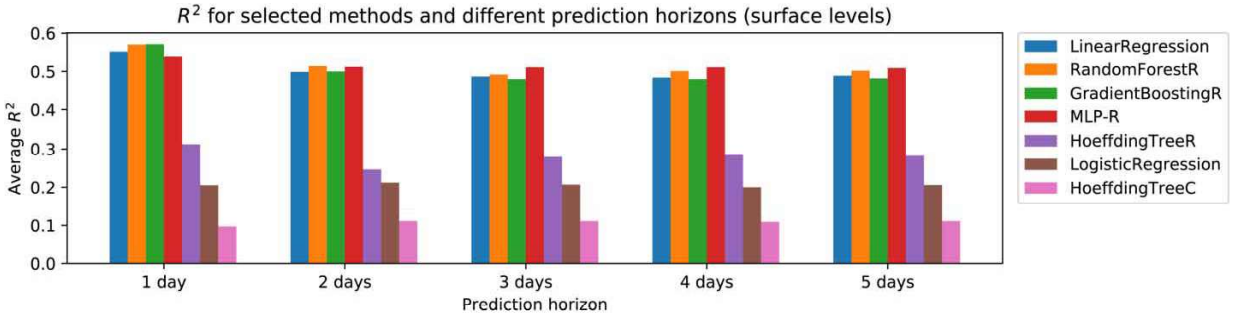


Figure 9 | R^2 of selected statistical models for different prediction horizons (surface water levels).

vectors) also occur in the learning data in cases where the change is smaller and the model learns to predict the value in between.

Compared to Figure 4, Figure 8 shows predictions that are less accurate. Furthermore, it can be concluded that the surface water levels are much less stable than groundwater levels. The changes in water levels are more rapid (the system is more responsive). The water levels are rising but also decreasing much faster. It is much more difficult to accurately predict such behavior.

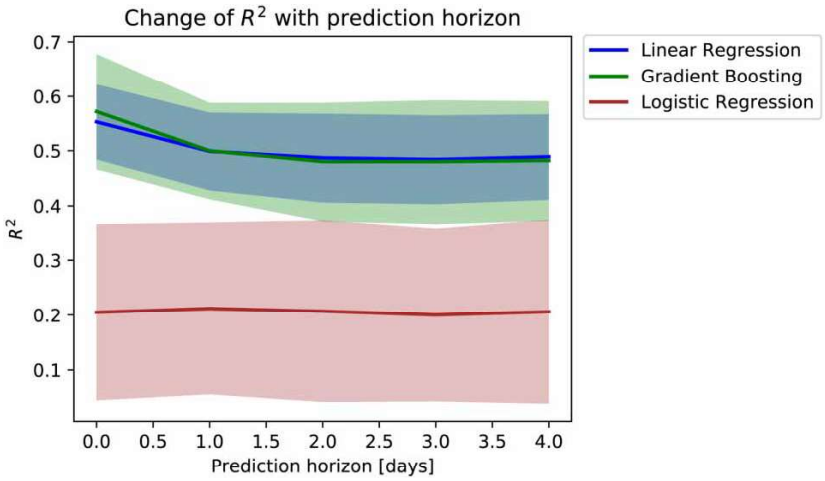


Figure 10 | Change of R^2 and its standard deviation with prediction horizon for linear regression, gradient boosting, and Hoeffding tree regression.

The performance of statistical modeling techniques in this case is similar to the modeling of groundwater levels. R^2 is generally lower than in the case of groundwater, which means that the input data did not include all the features that reflect the system dynamics and that the dynamics of surface water is less stable compared to groundwater. There is also a greater relative discrepancy between the batch regression methods and the streaming and classification methods, which are unsuitable in this scenario.

A further visualization of R^2 is shown in Figure 10 where beside the value of R^2 also the standard deviation of R^2 in 22 experiments (for 22 different stations) is shown. A decrease in accuracy in linear regression and gradient boosting for longer prediction horizons can be observed. It is also evident that linear regression is more stable than gradient boosting because the distribution of results is closer to the mean. The classification-based method (logistic regression) behaves significantly worse and gives unstable results.

Feature importances of gradient boosting regression with a 3-day prediction horizon for a selected surface water level sensor are shown in Figure 11. The precipitation intensity averaged over 2 days is the predominant characteristic that is easy to interpret. It is more important, whether precipitation can infiltrate the ground than how much precipitation there is. Excess water is transported on the surface and raises the surface water levels. Other features are far less important. The precipitation probability and its derivatives represent the other family of important features that influence the model.

CONCLUSIONS

This paper provides a comprehensive overview of the performance of statistical modeling techniques in applications of groundwater and surface water level forecast. Standard batch and incremental ML techniques for regression and classification are included. The latter are used on binned target values of water levels.

Comparison of regression techniques with classification methods on discretized bins reveals that the classification techniques are significantly inferior. An interpretation is that even though classification offers a wide range of ML techniques, the nature of the data is such that no binning is possible without worsening the results. The final performance depends heavily on how well the targets are binned, which is a complex task that must consider the density and distribution of values and is a matter of subjective interpretation. On the other hand, regression techniques can naturally handle the prediction of a target continuum of values, which leads to better results. Nevertheless, the classification techniques are much more successful in determining the starting time of a groundwater level change in longer prediction horizons. In combination with regression

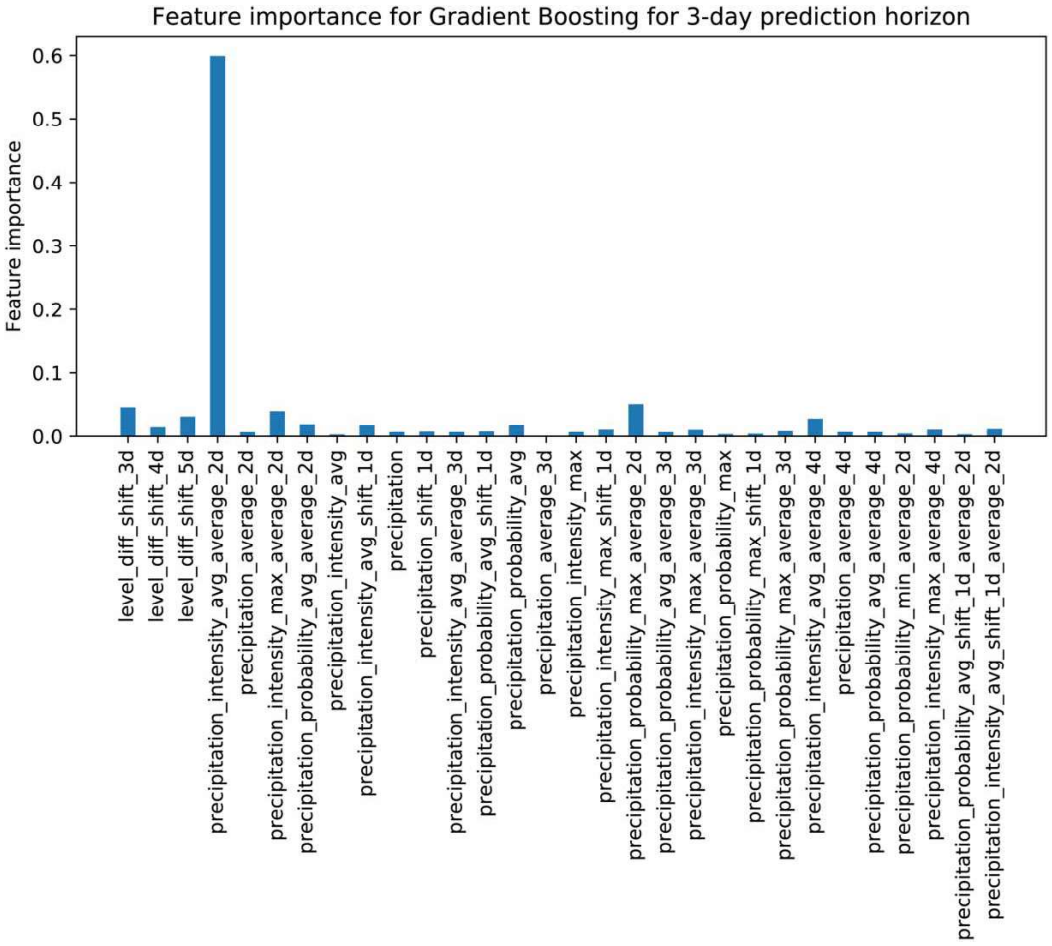


Figure 11 | Feature importance scores (relative) as given by the gradient boosting regression model for 3-day prediction horizon for surface water level modeling.

techniques, this could represent a potentially superior modeling approach.

Analysis of regression methods shows that despite streaming techniques adapt to incoming data and detect concept drift, they are consistently outperformed by their batch counterparts by a significant margin. Among streaming methods, the best results were obtained with the Hoeffding tree regression, which could provide competitive results (in terms of computational performance) in some scenarios. Gradient boosting, multilayer perceptron neural network, random forest regression, and linear regression achieved the best results in both use-cases. The good performance of linear regression can be attributed to the extensive feature engineering and to the nature of the underlying physical models.

Automatic feature engineering and feature selection algorithms, that enrich the water level values with contextual information and later prune it in order to avoid overfitting of ML models, are important contributions of the presented approach.

Finally, this work could be extended in various directions. The main challenge would be to provide a more comprehensive approach to the feature selection. The current approach has explored similarity-based and information gain-based methods; however, wrapper methods are expected to give even better results. Since the datasets are relatively small, the latter approach could find the nearly optimal set of features per sensor/prediction horizon in a reasonable time.

End-users can benefit from the effectiveness, accessibility, simplicity, and speed of the presented modeling solution. In order to put the AI methods into practice, a suitable Big Data architecture should be developed that

can handle automatic data acquisition, transformation, and fusion, as well as the generation of predictions in near real-time. Without these, the modeling results remain in the laboratory.

ACKNOWLEDGEMENTS

This research was funded by the European Union's Horizon 2020 programme project Water4Cities (Research and Innovation Staff Exchange) grant number 734409 and the European Union's Horizon 2020 programme project NAIADES (Innovation Action) grant number 820985.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this paper is available online at https://dx.doi.org/10.2166/aqua.2020.143.

REFERENCES

Adamowski, J., Fung Chan, H., Prasher, S. O., Ozga-Zielinski, B. & Sliusarieva, A. 2012 Comparison of multiple linear and nonlinear regression, autoregressive integrated moving average, artificial neural network, and wavelet artificial neural network methods for urban water demand forecasting in Montreal, Canada. *Water Resources Research* **48** (1), W01528.

Al-Qunaibet, M. H. & Johnston, R. S. 1985 Municipal demand for water in Kuwait: methodological issues and empirical results. *Water Resources Research* **21** (4), 433–438.

Amershi, S., Chickering, M., Drucker, S. M., Lee, B., Simard, P. & Suh, J. 2015 Modeltracker: redesigning performance analysis tools for machine learning. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, pp. 337–346.

Arbués, F., Garcıa-Valiñas, M. Á. & Martınez-Espiñeira, R. 2003 Estimation of residential water demand: a state-of-the-art review. *The Journal of Socio-Economics* **32** (1), 81–102.

Bifet, A. 2010 Adaptive stream mining: pattern learning and mining from evolving data streams. In: *Proceedings of the 2010 Conference on Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*. IOS Press, Amsterdam, The Netherlands, pp. 1–212.

Bifet, A. & Gavaldà, R. 2009 Adaptive learning from evolving data streams. In: *International Symposium on Intelligent Data Analysis*. Springer, Berlin, Heidelberg, pp. 249–260.

Bottou, L. 2010 Large-scale machine learning with stochastic gradient descent. In: *Proceedings of COMPSTAT'2010* (Y. Lechevallier & G. Saporta, eds). Physica-Verlag HD, Heidelberg, Germany, pp. 177–186.

Candelieri, A. 2017 Clustering and support vector regression for water demand forecasting and anomaly detection. *Water* **9** (3), 224.

Chen, C., He, W., Zhou, H., Xue, Y. & Zhu, M. 2020 A comparative study among machine learning and numerical models for simulating groundwater dynamics in the Heihe River Basin, northwestern China. *Nature Scientific Reports* **10** (1), 3904.

Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., Pardo, T. A. & Scholl, H. J. 2012 Understanding smart cities: an integrative framework. In: *Proceedings of 45th Hawaii International Conference on System Sciences (HICCS 2012)*. IEEE Computer Society Press, Maui, HI, USA, pp. 2289–2297.

Di Nardo, A., Alcocer-Yamanaka, V. H., Altucci, C., Battaglia, R., Bernini, R., Bodini, S., Bortone, I., Bourguett-Ortiz, V. J., Cammissa, A., Capasso, S. & Cascetta, F. 2015a New perspectives for smart water network monitoring, partitioning and protection with innovative on-line measuring sensors. In: *Procceding of IAHR World Congress*, The Hague, The Netherlands.

Di Nardo, A., Di Natale, M., Gisonni, C. & Iervolino, M. 2015b A genetic algorithm for demand pattern and leakage estimation in a water distribution network. *Journal of Water Supply: Research and Technology – AQUA* **64** (1), 35–46.

Domingos, P. & Hulten, G. 2000 Mining high-speed data streams. In: *Proceedings of KDD 2000*. ACM, Boston, USA, pp. 71–80.

Friedman, J. H. 2002 Stochastic gradient boosting. *Computational Statistics & Data Analysis* **38** (4), 367–378.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. 2014 A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* **46** (4), 44.

García Valverde, D., Quevedo Casín, J. J., Puig Cayuela, V. & Saludes Closa, J. 2015 Water demand estimation and outlier detection from smart meter data using classification and Big Data methods. In: *2nd New Developments in IT & Water Conference*, Rotterdam, Holland, pp. 1–8.

Goodfellow, I., Bengio, Y. & Courville, A. 2016 *Deep Learning*. MIT Press, Cambridge, MA, USA.

Griffin, R. C. & Chang, C. 1990 Pretest analyses of water demand in thirty communities. *Water Resources Research* **26** (10), 2251–2255.

Hastie, T., Tibshirani, R. & Friedman, J. 2009 *The Elements of Statistical Learning*. Springer Series in Statistics. Springer, New York, NY, USA.

Ikonomovska, E., Gama, J. & Džeroski, S. 2015 Online tree-based ensembles and option trees for regression on evolving data streams. *Neurocomputing* **150**, 458–470.

Ioannou, A. E., Kofinas, D., Spyropoulou, A. & Laspidou, C. 2017 Data mining for household water consumption analysis using self-organizing maps. *European Water* **58**, 443–448.

Jeong, J. & Park, E. 2019 Comparative applications of data-driven models representing water table fluctuations. *Journal of Hydrology* **572**, 261–273.

Kanakoudis, V., Tsitsifli, S., Papadopoulou, A., Curk, B. C. & Karleusa, B. 2016 Estimating the water resources vulnerability index in the Adriatic Sea region. *Procedia Engineering* **162** (Suppl. C), 476–485.

Kenda, K., Čerin, M., Bogataj, M., Senožetnik, M., Klemen, K., Pergar, P., Laspidou, C. & Mladenić, D. 2018a Groundwater modeling with machine learning techniques: Ljubljana polje aquifer. *Proceedings* **2** (11), 697.

Kenda, K., Koprivec, F. & Mladenić, D. 2018b Optimal missing value estimation algorithm for groundwater levels. *Proceedings* **2** (11), 698.

Kenda, K., Kažič, B., Novak, E. & Mladenić, D. 2019 Streaming data fusion for the Internet of Things. *Sensors* **19** (8), 1955.

Kofinas, D., Mellios, N., Papageorgiou, E. & Laspidou, C. 2014 Urban water demand forecasting for the island of Skiathos. *Procedia Engineering* **89**, 1023–1030.

Krause, J., Perer, A. & Bertini, E. 2014 INFUSE: interactive feature selection for predictive modeling of high dimensional data. *IEEE Transactions on Visualization and Computer Graphics* **20** (12), 1614–1623.

Krause, J., Perer, A. & Ng, K. 2016 Interacting with predictions: visual inspection of black-box machine learning models. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, pp. 5686–5697.

Laspidou, C. 2014 ICT and stakeholder participation for improved urban water management in the cities of the future. *Water Utility Journal* **8**, 79–85.

Liu, H. & Motoda, H. 2007 *Computational Methods of Feature Selection*. CRC Press, Boca Raton, FL, USA.

Maier, H. R. & Dandy, G. C. 2000 Neural networks for the prediction and forecasting of water resources variables: a review of modeling issues and applications. *Environmental Modeling & Software* **15** (1), 101–124.

Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J. & Aharon, D. 2015 *Unlocking the Potential of the Internet of Things*. McKinsey Global Institute, New York, NY, USA.

Mellios, N., Kofinas, D., Papageorgiou, E. & Laspidou, C. 2015 A multivariate analysis of the daily water demand of Skiathos Island, Greece, implementing the artificial neuro-fuzzy inference system (ANFIS). In: *E-Proceedings of the 36th IAHR World Congress*, The Hague, The Netherlands, pp. 1–8.

Nie, S., Bian, J., Wan, H., Sun, X. & Zhang, B. 2016 Simulation and uncertainty analysis for groundwater levels using radial basis function neural network and support vector machine models. *Journal of Water Supply: Research and Technology – AQUA* **66** (1), 15–24.

Oyebode, O. 2019 Evolutionary modeling of municipal water demand with multiple feature selection techniques. *Journal of Water Supply: Research and Technology – AQUA* **68** (4), 264–281.

Oza, N. C. 2005 Online bagging and boosting. 2005. In: *IEEE International Conference on Systems, Man, and Cybernetics*, Waikoloa, Hawaii, USA, pp. 2340–2345.

Recknagel, F. 2001 Applications of machine learning to ecological modeling. *Ecological Modeling* **146** (1–3), 303–310.

Rogers, P., De Silva, R. & Bhatia, R. 2002 Water is an economic good: how to use prices to promote equity, efficiency, and sustainability. *Water Policy* **4** (1), 1–17.

Sarle, W. S. 1994 Neural networks and statistical models. In: *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. SAS Institute, Cary, NC, USA, pp. 1538–1550.

Soleimani, S., Bozorg-Haddad, O., Saadatpour, M. & Loáiciga, H. A. 2019 Simulating thermal stratification and modeling outlet water temperature in reservoirs with a data-mining method. *Journal of Water Supply: Research and Technology – AQUA* **68** (1), 7–19.

Swilling, M., Robinson, B., Marvin, S. & Hodson, M. 2013 *City-Level Decoupling: Urban Resource Flows and the Governance of Infrastructure Transitions*. A Report of the Working Group on Cities of the International Resource, UNEP (United Nations Environment Programme), Nairobi, Kenia.

Tiwari, M. K. & Adamowski, J. F. 2014 Medium-term urban water demand forecasting with limited data using an ensemble wavelet–bootstrap machine-learning approach. *Journal of Water Resources Planning and Management* **141** (2), 04014053.

Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N. M. & Nelson, L. E. 2010 *Helping CIOs Understand 'Smart City' Initiatives: Defining the Smart City, Its Drivers, and the Role of the CIO*. Forrester Research, Inc, Cambridge, MA, USA.

Zhang, Q., Yang, L. T., Chen, Z. & Li, P. 2018 A survey on deep learning for big data. *Information Fusion* **42**, 146–157.

First received 13 November 2019; accepted in revised form 25 March 2020. Available online 20 April 2020

Chapter 6

Conclusions and Further Work

> The only thing that makes life possible
> is permanent, intolerable uncertainty;
> not knowing what comes next.
>
> *Ursula K. Le Guin*

This thesis investigates the entire vertical needed to handle multiple heterogeneous time series analysis and the implementation of predictive models for IoT in real-world scenarios. The contributions are significant in terms of the architectures and frameworks involved, as well as the development of data cleaning and pre-processing capabilities, and ultimately modeling. The focal point of the thesis is the framework for autonomous online fusion of heterogeneous sensor streams data, which serves as the foundation for many other features of the proposed platform, including efficient modeling with contextual data and real-time prediction generation.

We have proposed several new algorithms and innovations. A new algorithm using Kalman filtering has been suggested for autonomous online time series data cleaning. A new approach for online data fusion of heterogeneous data streams in predictive modeling has been introduced. A feature selection algorithm, called FASTENER, has been proposed, which utilizes genetic algorithms and multi-objective optimization methods to achieve faster convergence and improved feature sets compared to previous methods. To the best of our knowledge, the usability of incremental learning methods in water management scenarios has been analyzed for the first time. Lastly, an extension to the Big Data lambda architecture has been proposed, enhancing the predictive capabilities of the analytical frameworks in the speed (online) pillar.

The solutions described in this thesis have been developed and validated as part of our activities in several research projects funded by the FP7 and H2020 programmes and have also been implemented in industry. The entire range of applications has been utilized and deployed in the H2020 NAIADES project, where it was enhanced with various specific water management solutions. The thesis encompasses numerous evaluations conducted in the fields of environment, transportation, energy, and Earth observation.

In the rest of this chapter, we begin by providing a summary of the scientific contributions made in this thesis. We then proceed to discuss the goals and hypotheses presented in Chapter 1, taking into account our contributions and results. Finally, we conclude by outlining some potential directions for future research.

6.1 Contributions to Science

The scientific contributions of the thesis lie in the domains of data pre-processing, big data architectures. We also offer an assessment of the developed methodologies in various real-world scenarios. Below, we will further divide each contribution and provide a brief discussion and evaluation for each.

SC1 - Novel methodologies for data cleaning, data fusion and feature selection:

- **SC1.1 - Incremental data cleaning**: We have developed an incremental data cleaning methodology based on Kalman filter. Our approach demonstrates that this methodology can be initialized without prior knowledge regarding the data stream to be cleaned, making it well-suited for application in IoT data streams, where periodicity of the observed phenomena is typically considerably shorter than the sampling interval. To evaluate our methodology, we conducted tests on a variety of datasets, including one synthetic labeled dataset and six unlabeled datasets, consisting of two synthetic datasets and four real-world datasets. Within the unlabelled datasets, we have implemented an outlier detection assessment based on the outcomes of autoregressive modeling. The results clearly indicate that the cleansed datasets lead to improved modeling results.

- **SC1.2 - Incremental data enrichment and data fusion:** We have introduced a formal definition of heterogeneous data streams fusion. In this formal definition, we establish the essential components, encompassing data streams and the requisite operators, which collectively yield the generation of enriched feature vectors that enable more accurate predictive modeling. We developed a generic streaming data fusion framework for heterogeneous data streams. To the best of our knowledge, we provided the first generic framework capable of generating feature vectors from heterogeneous data streams, thereby enabling the application of machine learning techniques in real-time streaming scenarios. We demonstrated the usability of the platform beyond the controlled laboratory environment, integrating it in several real-world pilot scenarios.

- **SC1.3 - Feature selection:** We proposed a novel genetic wrapper algorithm for feature selection that we coined FASTENER (feature selection enabled by entropy). The algorithm reduces the number of features in a machine learning problem while preserving or even improving modeling accuracy. Using pre-trained models for information gain calculation, the algorithm converges faster towards the optimal feature set for a particular modeling task than similar algorithms. The algorithm improved state-of-the-art accuracy in the remote sensing application for land-cover classification and has also shown competitive performance on several feature selection benchmark datasets (with very small samples and very big number of features).

SC2 - Extension of lambda architecture:

- **SC2.1 - Extended lambda architecture:** The proposed analytical platform is based on the lambda and hut architectures. We suggested refinements of the big data architectures in order to support the needs in the water domain. In the lambda architecture, we suggested to include modeling capabilities in the speed pillar to overcome the traditional capabilities, limited to (complex) event processing.

- **SC2.2 - Conceptual architecture for water management analytical platform based on extended lambda architecture:** A solution for heterogeneous

sources data fusion in a stream was integrated in the architecture. The setting enables contextual information to be included in the data-driven models. This contribution enables integration of machine learning applications to the real-world scenarios and overcomes the laboratory setting for testing of forecasting or anomaly detection models on a stream of data.

SC3 - Evaluation in real-world scenarios:

- **SC3.1 - Evaluation of the proposed architecture and methods in several real-world scenarios:** We successfully integrated a solution for heterogeneous sources data fusion in a stream in the architecture. Such a setting not only facilitates the inclusion of contextual information in data-driven models but also culminates in a notable enhancement of the overall model accuracy.

  The initial ideas for this work have been developed already in 2014 within FP7 NRG4CAST project and then later implemented also in FP7 Sunseed project, where data fusion has been utilized in the energy management sector, handling forecasting of energy demand/production in thermal plants, smart buildings, smart cities and smart grids. The implementation of smart grid solutions was successfully executed within the project in collaboration with Iskratel d.d. Furthermore, these solutions were effectively deployed in their commercial applications. Further development followed during the H2020 In2Dreams project, where the infrastructure has been used in the mobility use case, mainly estimating energy consumption of electric trains for short- and mid-term prediction horizons. The usability of data fusion has been studied for land-cover applications based on satellite imagery in H2020 PerceptiveSentinel project, where FASTENER feature selection has been developed. Finally, the lambda framework has been formalized in H2020 Water4Cities and H2020 NAIADES projects. The former provided a rich implementation of the framework (including data fusion, anomaly detection, predictions and complex domain applications) in the water management sector.

  The papers presented in this thesis also include evaluation of presented methodologies with the focus on the benefits of heterogeneous on-line sensor data fusion in several scenarios: environmental scenarios [1], [3], [20], energy management [2], transport [2], and Earth observation [5].

- **SC3.2 - Evaluation of incremental learning methods in the water domain:** Stream mining techniques improve the computational performance of the pipeline and provide models that are better able to adjust to changes (concept drift) in real-time data than traditional batch models. The architecture natively supports incremental methods, covered in SC1, and therefore fits well with incremental learning techniques. We introduced incremental learning techniques to modeling of surface and groundwater levels. In that scenario, we prepared a comparison of 21 different statistical modeling techniques at different time horizons.

6.2 Further Work

The thesis proposes solutions for applications of analytics in IoT, addressing a broad variety of challenges through the entire data processing vertical, specifically in data cleaning, data fusion, feature selection, predictive modeling, and the corresponding architectural design. In the further work section, we limit ourselves to the areas where this thesis provides the most significant contributions. Further work is outlined in three categories: applications, architecture and data fusion. In the latter part, we also acknowledge and discuss

the significant influence of deep learning models on the accomplishments presented in this thesis.

Applications. One potential avenue for further research would involve assessing the performance of the proposed system in various other contexts, such as smart cities, smart buildings, smart agriculture, health care, finance, manufacturing, and other relevant domains. The unique characteristics of different domains may provide valuable insights to enhance the functionality of the presented components to better meet the specific requirements of use cases.

Architecture. From an architectural perspective, there are a couple of potential directions to consider. An intriguing enhancement to the current system could be the implementation of a feature store, which would extend the current data management layer. As feature vectors are generated in an incremental manner, the feature store would greatly enhance the system's usability and also offer improved traceability and troubleshooting capabilities for data engineers.

The need to automate and operationalize machine learning models has led to a new field within computer engineering and data science called machine learning operations (MLOps) [28]. Several new architectures have been proposed in the field in recent years that present end-to-end ML frameworks, which address and solve similar problems as enhanced lambda architecture. The solutions rely on horizontally scalable elements capable of managing large volumes of data from the Internet of Things (IoT) and other sources. Future research should take into account the potential impact of recent achievements in MLOps to the rationality and efficacy of the proposed lambda architecture.

Data fusion. The limitations of the custom data fusion components have been observed through practical experience. These include issues such as the delayed calculation of features that rely on a significant amount of historical data for computation, as well as the complex process of restarting the system after a crash or the injection of invalid data. The latter situation often occurs during the development of a new model and, therefore, represents a bottleneck in the whole modeling cycle. Over the past few years, there has been a significant increase in the popularity of time series databases (TSDB), leading to the development and availability of several mature implementations. Utilizing the capabilities of TSDB's, a data fusion approach would conceptually represent a regression from the presented incremental data fusion algorithms. Nevertheless, the anticipated outcome would offer improved stability and faster prototyping of machine learning solutions.

Evaluation of data fusion systems is difficult and is usually achieved through its effect on modeling capabilities. We propose a future investigation of the evaluation of the level of expressiveness of the language used to define feature vectors.

The connection between the data fusion component and its applicability in time series deep learning models has not been explored in our research. By utilizing Long Short-Term Memory (LSTM), the models have acquired the capability to retain information in memory over extended periods of time. Models like these have shown great success in processing sequential data, such as in analyzing time series. The ability to recall information may make certain aspects of the data fusion component obsolete, such as including delayed measurements in the feature vector. A similar effect could be achieved with self-attention mechanisms in transformers and other relevant deep learning approaches (N-BEATS [29], N-HiTS [30], PatchTST [31]). Additional studies are needed to determine the extent to which data fusion contributes to the performance of deep neural network models.

Driven by the success of ChatGPT and other large language models, several propos-

als have emerged recently for foundational models for time series. One such example is TimeGPT-1 [32], which is very successful with zero-shot inference of volatile time series. In the coming years, significant advancements are expected with these types of models, which could potentially provide a feasible alternative to current time-series modeling approaches. Because multivariate time series are not considered in these models, our approach may consistently outperform foundational time-series models for several use cases from presented domains. We recommend that future research examines the usability of foundational models for time series in use cases that involve relevant contextual data.

References

[1] K. Kenda, N. Mellios, P. Pergar, and M. Senožetnik, "Computer architectures for incremental learning in water management," *Sustainability*, vol. 14, no. 5, p. 2882, 2022.

[2] K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the Internet of Things," *Sensors*, vol. 19, no. 8, p. 1955, 2019.

[3] K. Kenda and D. Mladenić, "Autonomous sensor data cleaning in stream mining setting," *Business Systems Research Journal*, vol. 9, no. 2, pp. 69–79, 2018.

[4] T. John and P. Misra, *Data lake for enterprises*. Packt Publishing Ltd, 2017.

[5] F. Koprivec, K. Kenda, and B. Šircelj, "FASTENER feature selection for inference from Earth observation data," *Entropy*, vol. 22, no. 11, p. 1198, 2020.

[6] K. Kenda, M. Škrjanc, and A. Borštnik, "Modelling of the complex data space: Architecture and use cases from nrg4cast project," in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, IEEE, 2015, pp. 1–4.

[7] K. Kenda, M. Škrjanc, and A. Borštnik, "Modelling in energy related scenarios," in *2015 18th International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2015, pp. 21–24.

[8] G. Petkovšek, M. Erznožnik, and K. Kenda, "Anomaly detection on live water pressure data stream," in *2021 24th International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2021, pp. 53–56.

[9] J. Costa, A. Costa, K. Kenda, and J. Pita Costa, "Entropy for time series forecasting," in *2021 24th International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2021, pp. 53–56.

[10] S. Li, L. D. Xu, and S. Zhao, "The Internet of Things: A survey," *Information systems frontiers*, vol. 17, pp. 243–259, 2015.

[11] T. Kolajo, O. Daramola, and A. Adebiyi, "Big data stream analysis: A systematic literature review," *Journal of Big Data*, vol. 6, no. 1, p. 47, 2019.

[12] M. Bahri, A. Bifet, J. Gama, H. M. Gomes, and S. Maniu, "Data stream analysis: Foundations, major tasks and tools," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 3, e1405, 2021.

[13] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "Unlocking the potential of the Internet of Things," *McKinsey Global Institute*, vol. 1, 2015.

[14] D. Q. Tu, A. Kayes, W. Rahayu, and K. Nguyen, "ISDI: A new window-based framework for integrating IoT streaming data from multiple sources," in *Advanced Information Networking and Applications: Proceedings of the 33rd International Conference on Advanced Information Networking and Applications (AINA-2019) 33*, Springer, 2020, pp. 498–511.

[15] L. Zhang, N. Xiao, W. Yang, and J. Li, "Advanced heterogeneous feature fusion machine learning models and algorithms for improving indoor localization," *Sensors*, vol. 19, no. 1, p. 125, 2019.

[16] J. Wu, Y. Feng, and P. Sun, "Sensor fusion for recognition of activities of daily living," *Sensors*, vol. 18, no. 11, p. 4029, 2018.

[17] F. Zhou, P. Hu, S. Yang, and C. Wen, "A multimodal feature fusion-based deep learning method for online fault diagnosis of rotating machinery," *Sensors*, vol. 18, no. 10, p. 3521, 2018.

[18] B. Wu, T. Huang, Y. Jin, J. Pan, and K. Song, "Fusion of high-dynamic and low-drift sensors using Kalman filters," *Sensors*, vol. 19, no. 1, p. 186, 2019.

[19] A. Akbar, G. Kousiouris, H. Pervaiz, J. Sancho, P. Ta-Shma, F. Carrez, and K. Moessner, "Real-time probabilistic data fusion for large-scale IoT applications," *Ieee Access*, vol. 6, pp. 10 015–10 027, 2018.

[20] K. Kenda, J. Peternelj, N. Mellios, D. Kofinas, M. Čerin, and J. Rožanec, "Usage of statistical modeling techniques in surface and groundwater level prediction," *Journal of Water Supply: Research and Technology—AQUA*, vol. 69, no. 3, pp. 248–265, 2020.

[21] D. Hall and J. Llinas, *Multisensor data fusion*. CRC press, 2001.

[22] J. Hua, W. D. Tembe, and E. R. Dougherty, "Performance of feature-selection methods in the classification of high-dimension data," *Pattern Recognition*, vol. 42, no. 3, pp. 409–424, 2009.

[23] W. Jia, M. Sun, J. Lian, and S. Hou, "Feature dimensionality reduction: A review," *Complex & Intelligent Systems*, vol. 8, no. 3, pp. 2663–2693, 2022.

[24] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by Pareto optimization," *Advances in neural information processing systems*, vol. 28, 2015.

[25] H. Alhakbani and M. M. Al-Rifaie, "Feature selection using stochastic diffusion search," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 385–392.

[26] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[27] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[28] D. Kreuzberger, N. Kühl, and S. Hirschl, "Machine learning operations (MLOps): Overview, definition, and architecture," *IEEE Access*, 2023.

[29] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.

[30] C. Challu, K. G. Olivares, B. N. Oreshkin, F. G. Ramirez, M. M. Canseco, and A. Dubrawski, "N-HiTS: Neural hierarchical interpolation for time series forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 6989–6997.

[31] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," *arXiv preprint arXiv:2211.14730*, 2022.

[32] A. Garza and M. Mergenthaler-Canseco, "TimeGPT-1," *arXiv preprint arXiv:2310.03589*, 2023.

Bibliography

Publications Related to the Thesis

Journal Articles

K. Kenda and D. Mladenić, "Autonomous sensor data cleaning in stream mining setting," *Business Systems Research Journal*, vol. 9, no. 2, pp. 69–79, 2018.

K. Kenda, B. Kažič, E. Novak, and D. Mladenić, "Streaming data fusion for the Internet of Things," *Sensors*, vol. 19, no. 8, p. 1955, 2019.

F. Koprivec, K. Kenda, and B. Šircelj, "FASTENER feature selection for inference from Earth observation data," *Entropy*, vol. 22, no. 11, p. 1198, 2020.

K. Kenda, J. Peternelj, N. Mellios, D. Kofinas, M. Čerin, and J. Rožanec, "Usage of statistical modeling techniques in surface and groundwater level prediction," *Journal of Water Supply: Research and Technology—AQUA*, vol. 69, no. 3, pp. 248–265, 2020.

K. Kenda, N. Mellios, P. Pergar, and M. Senožetnik, "Computer architectures for incremental learning in water management," *Sustainability*, vol. 14, no. 5, p. 2882, 2022.

Conference Papers

G. Petkovšek, M. Erznožnik, and K. Kenda, "Anomaly detection on live water pressure data stream," in *2021 24th International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2021, pp. 53–56.

J. Peternelj, B. Šircelj, and K. Kenda, "Usage of incremental learning in land-cover classification," in *2020 23rd International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2020, pp. 57–60.

F. Koprivec, J. Peternelj, and K. Kenda, "Feature selection in land-cover classification using EO-learn," in *2019 22nd International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2019, pp. 37–40.

F. Koprivec, M. Čerin, and K. Kenda, "Crop classification using PerceptiveSentinel," in *2018 21st International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2018, pp. 37–40.

K. Kenda, M. Čerin, M. Bogataj, M. Senožetnik, K. Klemen, P. Pergar, C. Laspidou, and D. Mladenić, "Groundwater modeling with machine learning techniques: Ljubljana polje aquifer," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, 2018, p. 697.

M. Senožetnik, Z. Herga, T. Šubic, L. Bradeško, K. Kenda, K. Klemen, P. Pergar, and D. Mladenić, "IoT middleware for water management," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, 2018, p. 696.

K. Kenda, F. Koprivec, and D. Mladenić, "Optimal missing value estimation algorithm for groundwater levels," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, 2018, p. 698.

S. Rizou, K. Kenda, D. Kofinas, N. Mellios, P. Pergar, P. D. Ritsos, J. Vardakas, K. Kalaboukas, C. Laspidou, M. Senožetnik, *et al.*, "Water4Cities: An ICT platform enabling holistic surface water and groundwater management for sustainable cities," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, 2018, p. 695.

Other Publications

Journal Articles

P. Zajec, J. M. Rožanec, E. Trajkova, I. Novalija, K. Kenda, B. Fortuna, and D. Mladenić, "Help me learn! architecture and strategies to combine recommendations and active learning in manufacturing," *Information*, vol. 12, no. 11, p. 473, 2021.

J. Rožanec, E. Trajkova, K. Kenda, B. Fortuna, and D. Mladenić, "Explaining bad forecasts in global time series models," *Applied Sciences*, vol. 11, no. 19, p. 9243, 2021.

J. Rožanec, E. Trajkova, I. Novalija, P. Zajec, K. Kenda, B. Fortuna, and D. Mladenić, "Enriching artificial intelligence explanations with knowledge fragments," *Future Internet*, vol. 14, no. 5, p. 134, 2022.

J. M. Rožanec, I. Novalija, P. Zajec, K. Kenda, H. Tavakoli Ghinani, S. Suh, E. Veliou, D. Papamartzivanos, T. Giannetsos, S. A. Menesidou, *et al.*, "Human-centric artificial intelligence architecture for industry 5.0 applications," *International Journal of Production Research*, pp. 1–26, 2022.

Book Chapters

F. Koprivec, G. Kržmanc, M. Škrjanc, K. Kenda, and E. Novak, "Screening tool for anti-money laundering supervision," in *Big Data and Artificial Intelligence in Digital Finance: Increasing Personalization and Trust in Digital Finance using Big Data and AI*, Springer International Publishing Cham, 2021, pp. 233–251.

J. M. Rožanec, I. Novalija, P. Zajec, K. Kenda, and D. Mladenic, "Knowledge modelling and active learning in manufacturing," *Trusted Artificial Intelligence in Manufacturing: A Review of the Emerging Wave of Ethical and Human Centric AI Technologies for Smart Production*, pp. 52–72, 2021.

Conference Papers

J. Costa, A. Costa, K. Kenda, and J. Pita Costa, "Entropy for time series forecasting," in *2021 24th International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2021, pp. 53–56.

M. Čerin and K. Kenda, "Amazon forest fire detection with an active learning approach," in *2020 23rd International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2020, pp. 69–72.

S. Kralj, Ž. Urbančič, E. Novak, and K. Kenda, "Document embedding models on environmental legal documents," in *2019 22nd International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2019, pp. 29–32.

M. Čerin, F. Koprivec, and K. Kenda, "Early land cover classification with Sentinel 2 satellite images and temperature data," in *2019 22nd International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2019, pp. 45–48.

M. M. Besher, S. Brezec, E. Novak, and K. Kenda, "Semantic enrichment and analysis of legal domain documents," in *2019 22nd International Multiconference Information Society (IS)*, Institut Jožef Stefan, 2019, pp. 17–20.

Biography

Klemen Kenda is a researcher at Jožef Stefan Institute and Qlector d.o.o. He obtained his diploma in physics at University of Ljubljana and is pursuing his PhD in Information and Communication Technologies at Jožef Stefan International Postgraduate School.

His broad early working experience includes development of control systems for large physics experiments at Sinchrotrone Trieste and Forschungzentrum Karlsruhe (collaboration with F2 department at JSI, later Cosylab), data analysis for online advertising at Httpool d.o.o., implementing geospacial solutions at DFG Consulting, analysing environmental software at Environmental Agency of Slovenia, independent web development, teaching of physics, mathematics and programming at Primary School Cerkno and Gimnazija Jurija Vege Idrija, managing public relations at Scout Association of Slovenia and several local, national and international management positions within Slovenian orienteering federation.

As a researcher, he has been involved with machine learning and stream mining of heterogeneous data sources. Applications of his work have been made in the fields of environmental intelligence, production planning and energy management. From 2011, he has contributed to several EU FP7, H2020 and Horizon Europe projects (Planetdata, Envision, NRG4CAST, Sunseed, PerceptiveSentinel, EnviroLENS, Factlog, Water4Cities, NAIADES, STAR, AquaSPICE, APRIORI, Plooto and HumAIne) and acted as a leader of several technical tasks and work packages.

As a researcher, he has been involved in the Qlector company, firstly as QlectorLEAP developer and later as EU project leader. Beside being used in several EU projects, his work described in the thesis has been implemented in several use cases also in industry.