NRG4CAST

Deliverable D3.1

Modelling of the Complex Data Space

Editor:	Klemen Kenda, JSI
Author(s):	Klemen Kenda, Maja Škrjanc, Branko Kavšek, Andrej Borštnik, Kristjan Mirčeta, JSI; Tatsiana Hubina, Diego Sanmatino, CSI; Yannis Chamodrakas, SLG; Irene Koronaki, Rosa Christodoulaki, NTUA; Giulia Losi, IREN; George Markogiannakis, CRES; Simon Mokorel, ENV;
Reviewers:	Yannis Chamodrakas, SLG; Irene Koronaki, NTUA;
Deliverable Nature:	Prototype (P)
Dissemination Level: (Confidentiality)[1]	Public (PU)
Contractual Delivery Date:	November 2014
Actual Delivery Date:	November 2014
Suggested Readers:	Developers creating software components to be integrated into final tool for different users. System analytics – expert NRG4Cast system user.
Version:	0.14
Keywords:	modelling, prediction, data streams, sensor data, sensor networks, model trees, Hoeffding trees, SVM

[1] Please indicate the dissemination level using one of the following codes:
• **PU =** Public • **PP =** Restricted to other programme participants (including the Commission Services) • **RE =** Restricted to a group specified by the consortium (including the Commission Services) • **CO =** Confidential, only for members of the consortium (including the Commission Services) • **Restreint UE =** Classified with the classification level "Restreint UE" according to Commission Decision 2001/844 and amendments • **Confidentiel UE =** Classified with the mention of the classification level "Confidentiel UE" according to Commission Decision 2001/844 and amendments • **Secret UE =** Classified with the mention of the classification level "Secret UE" according to Commission Decision 2001/844 and amendments

Disclaimer

Executive Summary

Deliverable D3.1 offers technical solution for heterogeneous multivariate data streaming modelling, built on top of the open-source QMiner platform. The prototype is able to receive data from different sources (sensors, weather, weather and other forecasts, static properties etc.) and with many different properties (frequency, update interval etc.). It is also able to merge and resample this data and build models on top of it.

Modelling use-cases for 5 pilots were defined and tested. Results for Turin public buildings, IREN thermal plant, NTUA university campus buildings, and EPEX energy spot markets have been produced. Average relative mean absolute error of the model predictions varies between five and ten percent, and qualitative analysis of predictions shows significant correlation between predictions and true values.

NRG4Cast models are a set of 24 models per modelling task. Each set has to make predictions for the next day – hour by hour. The following methods were used: moving average, linear regression, neural networks, and support vector machine regression. Additionally Hoeffding trees have been introduced and their implementation is based on many recent findings.

A by-product of this deliverable is also a set of visualization tools for data mining.

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
CET	Central European Time
DB	Database
GUI	Graphical User Interface
HT	Hoeffding tree (method)
JS	JavaScript
KF	Kalman Filter
LR	linear regression
NN	neural network (method)
RR	ridge regression (method)
MA	moving average (method)
OGSA-DAI	Open Grid Services Architecture – Data Access and Integration
SVMR	support vector machine regression (method)
QMAP	QMiner Analytical Platform

1 Introduction

"Prediction is very difficult, especially if it's about the future."

Niels Bohr, Nobel laureate in Physics

Deliverable D3.1 – Modelling of the Complex Data Space is one of the most important deliverables of the 2nd year of the NRG4Cast project. In this deliverable we have addressed some very **technical issues** including development of the streaming heterogeneous data modelling infrastructure and obtaining many new data sources for developing better models. A more **substantial part** of the deliverable includes developing modelling scenarios for different pilots, analysing different stream mining methods, analysing early pilot data, feature engineering, model creation, and model testing.

From the technical point of view, this deliverable is dealing with the streaming multivariate data infrastructure for modelling. The problem, as trivial as it might seem at first glance, brings a wide variety of smaller problems into the picture.

There are obvious issues, which include the problem of availability of stream mining methods and reusing batch methods in the streaming scenario. Once the methods are in place, there are many issues regarding the data stream. Streaming methods expect data to arrive in a timely fashion. In reality this is quite often not the case. When dealing with multivariate data from heterogeneous data sources many things do not match: frequency of the data is different, data is updated using different protocols (some data is coming on-line, other sources send data in many small batches; some collect the data for 15 minutes and then send it all in one batch, others update every hour, others every day, some sources are dependent on human interaction and update irregularly, etc.).

A lot of the issues mentioned above have been addressed and solved in this deliverable. The result is a working star network system, which is able to process heterogeneous streaming data and brings it to the point where it can be used for real time predictions in an easy way.

The next part of work in the deliverable is more substantial (modelling oriented). All the pilots have prepared modelling scenarios. Some of them changed substantially during the task (Miren, FIR).

Quite often old data sources have been found insufficient (e.g. weather, as open weather API's out there mostly do not offer historical weather data), some static sources have been rediscovered on the internet, and parsers have been re-implemented (EPEX). Some pilots even required completely new data sources (Miren – traffic sensors) …

We have prepared initial data analysis of the selected pilots. Common modelling demands have been extracted and infrastructure adjusted accordingly. Available sensor data has been gathered, additional features have been constructed, imported into infrastructure, and registered. We have prepared feature vector propositions for each of the pilots.

A short survey of prediction algorithms has been done. Encouraged by the good performance of model trees, we have supported the work on Hoeffding tree models that started in some fellow EU projects. Big parts of the implementation have been included within the NRG4Cast project as a contribution to the open-source community. Final results are a bit less encouraging, but nevertheless – quite some effort resulted in a functional, fast, and thorough implementation of the algorithm. Also some improvements have been made to the state-of-the-art, reported by Ikonomovska in [28].

Finally different methods have been tested on CSI, NTUA, IREN and EPEX pilots. We tested moving average, linear regression, Hoeffding trees, neural networks, and SVM regression. Some fine-tuning methods have also been implemented. Final models have been deployed and are sending predictions to the monitoring database and event detection service for further use (visualization, analysis).

Within the preparations of the models some side-results have been implemented. We started developing a QMiner Streaming Data Mining visualization tools.

Last, but not least, the QMiner Analytical platform has evolved substantially in the last year. QMiner became an open-source project and there had been 3 major revisions during the last 12 months. Many improvements have been made on this account and a lot of code rewriting was involved.

The task/deliverable to follow this one is T5.2/D5.2 (Data-driven prediction methods environment). This task will continue the work done in D3.1 and extend the findings with a more in-depth analysis of the models (although some superficial analysis of the findings has been already done in Section 5).

1.1 Phases of Work

The work for this deliverable was executed in 5 phases, as is depicted in Figure 1. Data related tasks are depicted in blue, modelling related tasks in green, and infrastructure related tasks in orange. The 5 phases roughly follow the basic steps in a forecasting task [25]: problem definition, gathering information, exploratory analysis, choosing and fitting models, and using and evaluating forecasting models.

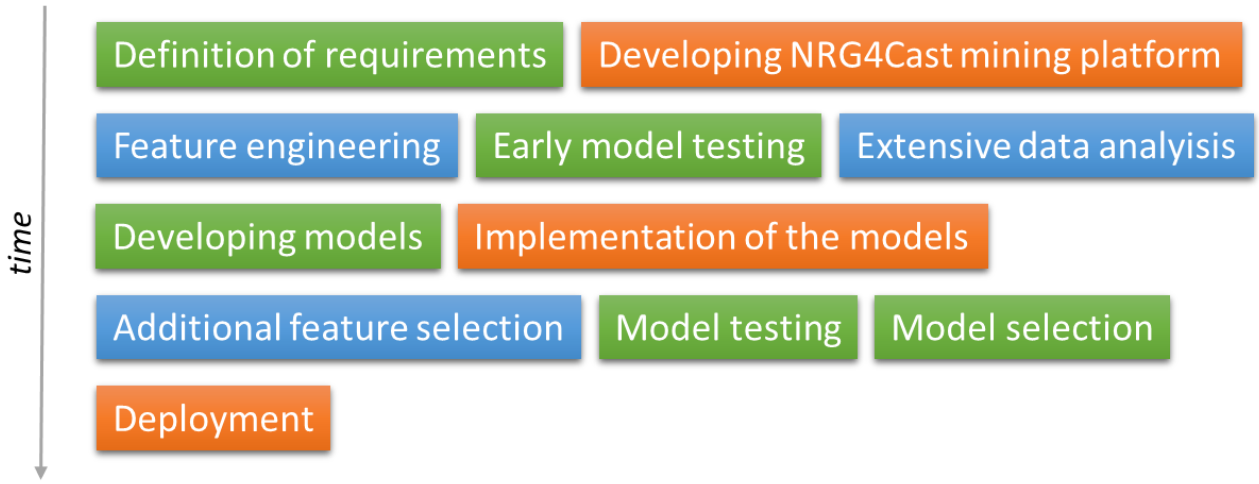


Figure 1: Modelling tasks in D3.1.

In the first phase we have extended the work of deliverable D7.2 (Pilot Scenarios) with more insight into the data and modelling needs of pilot scenarios. We have identified the needed additional features to the already provided data (among those we have determined the common/specific features for the pilots). Simultaneously we have also developed the NRG4Cast stream mining platform based on the QMiner to serve the needs of the planned modelling requirements.

In the next phase we have engineered the features and collected the needed data, which has been inserted into the NRG4Cast through already established infrastructure. Parallel to engineering features, we have conducted an early offline model testing of the algorithms, that are supported (or that shall be integrated) in the QMiner. We have also done some extensive data analysis of the selected pilot cases.

The findings were used in the next phase, where we developed and implemented the models.

These models have been refined and tested, until the best candidates have been selected in the phase 4, which was followed by the deployment of models to the *production* servers.

1.2 Composition of the Deliverable

Section 2 is presenting the efforts on the problem definition. Problem definitions in the chapter also include information on additional data needed in the pilot scenarios (additional features that can be common to all the scenarios or specific). Section 2 is extending the work done in deliverable NRG4CAST D7.2. The section includes all the pilot scenarios, although only 4 have been implemented in this phase of the project. Scenarios with most complete data and best defined outcomes have been chosen: IREN (thermal), TURIN (CSI building), NTUA (campus buildings), and EPEX spot market.

Section 3 is presenting work on feature engineering and the main modelling task: preparation of the feature vectors. Some exploratory data analysis is reported in the Appendix.

Section 4 is dedicated to Data Mining methods. We are presenting an overview of the measures for evaluating the models and methodology for this task. We are also briefly describing the algorithms we have tested. In this section we pay more attention to the development of the Hoeffding regression trees, which we have also implemented, tested, and contributed to the open-source QMiner platform.

Results of the model selection are presented in Section 5. Some early testing results are available in the Appendix.

We continue with two sections dedicated to the technical aspects of the modelling prototype. We discuss the optimal flow of the data in Section 6 and present the prototype (with its API and GUI) in Section 7. Some technical aspects (like contributions to the open-source software and sensor data description) are presented in the Appendix as well.

2 Problem Definition

Any modelling starts with the problem definition and the definition of desired results of the prediction methods. This whole section is dedicated to this task. Pilot case requirements for modelling have been materialized and concrete tasks have been set.

The problem definition is also accompanied with data (and additional properties) requierments, that could help the performance of the models. The first subsection is therefore dedicated to the common additional properties for modelling (which are used by most of the pilots).

2.1 Common Additional Properties for All Use Cases (CSI)

Common properties are brought into the system in the form of a time series. It was proposed the time granularity be 15 minutes. The streaming infrastructure however demands new value of a property to be updated only when a change occurs (the last value is carried on with the merger interpolators – see technical modelling details in Section 6). Properties can be prepared in advance and sent to the system in one single batch using the standard NRG4Cast Streaming API.

- Day of the week
- Free day or weekend or holiday
- Day/night (sunrise and sunset depends on geo-location)
- Season
- Month of Year
- Current Weather (temp, cloudy/sunny etc.)
- Weather prediction
- Regular lunch time
- Holiday season

Within the Data Layer properties are handled like normal sensor data. Aggregates are also calculated on top of the features. They can also be useful in the modelling scenario (for example: a portion of working time in a week could be a nice feature to have). However, the properties are handled differently within the model.

2.2 Public Buildings Turin

In the CSI Scenario a publicly owned building offering office space to private companies has been equipped with all kinds of energy sensors. The building offers rooms, offices, meeting rooms, and shared space, where all typologies represent different kinds of energy demand profiles and thus the building offers a broad lateral cut for typical office buildings.

The sensors track real time energy consumption of the different typologies of the building and measure electricity, as well as thermal demands. Collected data is then enriched with weather data. Energy management enables the tracking of power quality and reliability, while also offering measures to react quickly to critical situations. Moreover it aids in analysing historical data and detecting energy waste or unused capacities. All this data is also able to allocate costs for buildings, departments, and processes. The main objective is to monitor the entire building and predict energy demands at specific times. Furthermore individual suggestions on the use of energy can be made and potential for improvement can be shown, thus raising energy awareness.

2.2.1 Use case 1: Streaming data integration and management

This use case aims to build a reliable and comprehensive solution for data integration and to achieve complete information on energy consumption of a single building. It is the base for setting up the politics on changing the employee / inhabitants' behaviour.

The target dimensions that needs to be optimised for this use case are the actual building energy performance, situation on energy saving, money saving, and possibility for anomaly detection. Energy types considered by this use case are electric energy and district heating. The main user would be the energy provider, the building owner, employees, and energy operators. By using the comprehensive solution for data integration and management, the user will be able to make a decision on how to use the energy, "where to buy energy", and try to optimize employees' habits. To provide these decision options, the Turin pilot needs to take several information into account, such as detailed information on energy consumption, number of employees in the building, building/office description, historical data on energy consumption, and behaviour. The effect of this use case would be a chance to influence energy consumption (priorities for use of electrical energy and district heating), as the user can monitor detailed energy consumption regarding day times.

Italian pilot in Turin is situated within the area with moderate climate and no extreme climate situations can be evaluated. The pilot takes in consideration building typologies and Energy performance coefficient which refers to these typologies. All the pilot achievements should be considered for single climate zone. In case of replication of pilot results for different European areas, climate zone has to be taken in consideration.

The addition information this use case needs is the detailed information on energy consumption of a single building. This information is obtained through monitoring of the electrical and thermal energy consumption of a single building and typical offices. These information will support energy managers in making decisions.

2.2.2 Use case 2: Real-time analysis, reasoning, and network behaviour prediction

The second use case handles the real-time analysis, reasoning, and network behaviour prediction. This use case describes the improved and accurate prognosis on clusters of buildings energy consumption. The target dimensions that need to be optimised are the knowledge of the overall energy consumption of a cluster of public buildings in Turin, status on energy saving, and money saving. This use case will help Turin buildings involved in the project to be in line with the European policy on CO2 emissions. This use case deals with electric energy and district heating. The main user involved in this use case are building energy managers, ESCOs, energy providers, and employees.

By using this tool the user will be able to make a better decision on how to effectuate Improvement of energy efficiency and energy saving policy, which are very important . To provide these decision options, the Turin pilot needs to take information such as building typology and actual and historical energy consumption into account. This use case will make for an easier decision on priorities for use of electrical energy, district heating, alternative energy, improved forecast for a cluster of buildings, the amount of energy needed for the next year, and how much is the City expected to pay for a cluster of buildings.

Italian pilot in Turin is situated within the area with moderate climate and no extreme climate situations can be evaluated. The pilot takes in consideration building typologies and Energy performance coefficient, which refers to these typologies. All the pilot achievements should be considered for single climate zone. In case of replication of pilot results for different European areas, climate zone has to be taken in consideration. Another restriction is a limited building typology. Many historic buildings are taken in consideration in the case of Turin pilot. These building typologies would be difficult to reproduce in other countries. In order to apply the project achievements within the European or world building typologies, it's necessary to refer to international studies on building typologies, such as TABULA etc.

The additional information this use case needs is detailed information on energy consumption of a cluster of buildings. This pattern can be used for decision making at the city level. Another type of information, that can be delivered is the information based on the precise monitoring of a single building or a cluster of buildings. These new patterns are a forecast for a cluster of buildings.

2.2.3 Available data

Number of sensors: 4 (Building total energy consumption, except energy used for cooling; Energy consumption used for cooling of the Data Centre; Energy used for building cooling; Building Total energy consumption (without the Data Centre) – to be used by NRG4Cast).

Final number of sensors (they will be provided): 10 sensors (four sensors already installed in the building. Another six sensors will be installed in the typical offices of the CSI building).

Number of sensors (in the prototype): 4

Number of external sensors (in the prototype): 68 sensors measuring electrical and thermal energy (district heating) consumption are installed in the 34 public buildings in Turin. This number of public owned buildings are chosen based on the data availability of thermal energy consumption to be provided by IREN. These buildings will be involved in the MIREN-FIR-CSI-IREN Scenario, for the Turin part. The data flow will be provided for the project by IREN and integrated by CSI with the 3D GIS Turin models and 3D Energy Cadastre ENERCAD3D.

2.2.4 Proposed Additional Features

Specific features for CSI

- public offices working day schedule
- CSI working day schedule
- Lunch time
- Italian holidays

2.2.5 Desired results

For the 2nd year prototype daily consumption profiles will be predicted. This means that at 14:00 each day the system will predict energy consumption (electrical power) for the next day (24 values, hour-by-hour). In year 3 prediction horizon should be prolonged and modelling should be able to provide predictions of aggregated values (daily, weekly and monthly predicted consumption).

2.3 IREN pilot site

Use-case 1 (UC1): District heating production forecasting

Overall aim of UC1 and UC2: Use the NRG4CAST model to improve the energy efficiency of the production of DH in the city of Reggio Emilia.

UC1 Objective: The NRG4CAST model will foresee the total amount of thermal energy (MWh) requested by the DH network of the city of Reggio Emilia two days in advance, hour per hour, with respect to the outdoor temperature.

2.3.1 Available data

- Historical data of DH production
- Current data of DH production

2.3.2 Proposed Additional Features

- Historical data on Outdoor temperature (historical and current)

- Historical data on Wind speed (historical and current) – TBD
- Historical data of DH thermal production (MWh) (historical and current)

2.3.3 Desired results

The NRG4CAST model output will be a table (see the table below) that, 48 hours in advance, estimates the total amount of Thermal Energy requested by the city district heating network of the city of Reggio Emilia hour by hour, according to the forecasted outdoor temperature.

The model output, hour by hour, should be provided by 12.00 a.m. of each day during the thermal season (from 15th of October to 15th of April).

Example: Today, 10th of March 2014, the model produces an output concerning the 12th of March 2014, reporting the estimated value of Thermal Energy requested by the network and the forecasted outdoor temperature.

Date	Hour	Outdoor temperature (°)	Thermal Energy request (MWh)
12/03/2014	1	- 5	50
	2		
	3		
	4		
	5		
	6		
	7		
	8		
	9		
	10		
	11		
	12		

Date	Hour	Outdoor temperature (°)	Thermal Energy request (MWh)
	13		
	14		
	15		
	16		
	17		
	18		
	19		
	20		
	21		
	22		
	23		
	24		

Figure 2: Table for forecasting results for IREN UC1.

Influencing factors on the forecasted thermal energy requested by the network:

1. **Outdoor temperature:** The thermal energy requests vary with respect to the **outdoor temperature of the target day and of the day before**.
2. Additional influencing factors are: **wind speed, wind bearing, humidity rate**.
3. **Season:** 10% of district heating is consumed in summer time, compared to 90% produced and consumed in winter time. Winter time lasts from the 15th of October to the 15th of April. The NRG4CAST model will be used specifically for winter time predictions.
4. **Week day:** The Thermal energy demand varies significantly on working days compared to weekends and public holidays (e.g. Christmas time), when schools and public buildings as well as some private customers switch off their heating system.

2.4 District Heating in the Campus Nubi

Objective: The Campus Nubi will be used as a test site. The overall aim is improving the building energy performance.

The Campus Nubi is made up of 6 substations for heating and 1 substation for heating and domestic hot water.

The type of buildings involved are the following: warehouses, laboratories, offices and changing rooms.

The 6 thermal power plants provide with heating following buildings (see locations in Figure 3):

- **SST 5312: workshops heating and gas production**, district heating offices and chemical laboratories offices.

- **SST5319: offices and laboratories of electricians**, energetic class: D, volume: 2783,82 m3
- **SST5320: Warehouse**, energetic class: C, volume: 17457,88 m3
- **SST5310: office building and changing rooms**, energetic class: E, volume: 4213,36 m3
- **SST5305: building H – offices (Glass and steel palace)**, energetic class: F, volume: 3145,03 m3
- **SST5318: Management Building,** energetic class: D, volume: 7289,06 m3

Figure 3: Geographic location of buildings.

2.4.1 Available data

There are no historical data available for the Campus Nubi.

- the outdoor temperature,
- the indoor temperature,
- the forward water temperature,
- the backward water temperature,
- the energy consumption of each substation

Current energy consumption of the buildings within the Nubi Campus is available 4-5 times per year.

2.4.2 Desired results

Output of the forecasting model:

Visualization of a table that hour by hour shows the forecasted value of the water temperature of the secondary level for each substations according to the outdoor and indoor temperature of each building. The objective is to keep the indoor temperature at 20° C by regulating the temperature of the hot water.

As is situation:

Nowadays, the setting of both operation times and water temperature is set by IREN according to the outdoor temperature.

In the Nubi Campus each office is equipped with fan convectors on which the chosen room temperature is set.

The water is supplied at a fixed temperature, ranging between 55°C / 60°C, and the fan convector regulates the room temperature between 18°C and 22°C.

The only instrument for energy saving is the regulator, which, depending on the outside temperature and the settings of IREN as the service provider, increases the water temperature (e.g. at certain times I will have a certain outgoing water temperature).

To-be situation:

Act on the temperature regulator by modulating and setting the temperature of the warm water flow to the radiators, depending on the information provided by the outdoor probe (placed outside the building) and the room indoor probe (e.g. Water flow Temperature might be set at 60 ° from 7 am to 8.30 am in the morning in order to reach 20 ° of room temperature, then water temperature can be lowered for the rest of the time, in order to maintain the temperature at 20°).

Hour	Outdoor temperature	Indoor temperature	Secondary level water temperature
1	5°	20°	47°
2			
3			
4			
…			

Figure 4: Table of forecasting results for IREN UC2.

Expected benefits:

Saving of 5-15% of energy consumption.

Impact provided by the use of a new thermal ECU:

- By installing new regulators and new counters, that are remotely read and controlled, the district heating service will be optimized with more efficient district heating supply planning on the network (over time) and regulated according to the registered temperatures (indoor and outdoor) *(e.g. I can act to lower or dilute the peak of the central production, as well as distribute it in a wider range of time. I can choose the fact that it will reach the selected temperature in two hours, rather than in one hour).*

Impact provided by the usage of the Energy forecast system developed within the project:

- Possibility to predict, on the basis of the trends of the past years, as well as on the correlation between the environmental conditions and weather forecasts, the energy to be purchased for producing district heating

Possibility to determine in advance when to switch the various heat production plants on and how much energy to supply to the district heating network.

2.5 University Campus NTUA

The National Technical University of Athens includes nine academic Schools. The main campus is located in the Zografou area of Athens, spreading over an area of about 770,000 m^2; 260,000 m^2 of them are the buildings. Apart from the offices, lecture rooms, and laboratories, the campus hosts also the Central library, sports centre, conference centre, restaurant and cafes. The installed capacity is

- 30 MW for heating (natural gas boilers and heat pumps),
- 14.5 MW for cooling (heat pumps).

The annual energy demand of the NTUA campus is:

- 16000 MWh (6.1MW peak) for electricity (cooling, lighting and equipment)
- 8100 MWh for natural gas (space heating).

The objective of the NTUA pilot plant is threefold;

- **to monitor** the electricity consumption of each Building separately and of the Campus as a whole
- **to monitor** the thermal comfort levels inside a typical office in the Campus and
- to be able **to predict its electricity demand**.

Up to now, two buildings are being monitored in terms of electricity consumption: the Laboratory of Applied Hydraulics and the Rural & Surveying Engineering - Lampadario building. Moreover, at the time of writing, the required electricity meters for the monitoring of the whole Campus and the thermal comfort sensors are being installed. More specifically, 47 electricity sensors and 12 thermal comfort sensors (dry bulb temperature, relative humidity and illuminance) will have been installed by the first fortnight of January 2015. For demonstration purposes, a screen will also be installed at the entrance of the Rector's building. This screen will show the real time energy consumption of the NTUA campus and each School separately.

The objective of the NRG4Cast pilot in NTUA is to provide to all possible stakeholders the necessary information on the energy consumption of the Campus, the thermal comfort level, and the prediction of electricity demand, with the goal to assist in the energy management and decision making process. The information that will be produced will be used to select the best cost-effective measures for building renovation, to upgrade or to implement maintenance services to the heating, ventilation and air-conditioning systems, to select the optimum renewable energy solution for the Campus and to provide the employees/building users with information about the energy consumption in their building.

2.5.1 Available data

The available data so far is taken from two electricity consumption sensors installed at two different buildings on the Campus: the Laboratory of Applied Hydraulics and the Rural & Surveying Engineering - Lampadario building.

During the next month the available data will have been multiplied, since 47 electricity meters, 4 temperature sensors, 4 lux meters, and 4 relative humidity sensors will be installed in the Campus buildings. The deadline for the sensors installation is set to January 2015.

The aim is to monitor the electricity consumption of all Campus buildings and the thermal comfort of occupants in an indicative office.

2.5.2 Proposed Additional Features

The additional external data that the NTUA case should use are the following:

- Air conditioned area of each building, air conditioned area of all NTUA Campus
- Day of the week
- Weekend or holiday or strike
- Day/night
- Weather (temperature, irradiation, wind speed, humidity)
- Classes weekly schedule
- Exams annual schedule (September, February, June)
- Labs' occupancy
- Type of courses (undergraduate or graduate)
- Type of electromechanical system for heating and cooling of buildings

- Orientation of buildings
- Shading of Openings

Please note that the NTUA Campus does not have a regular lunch time.

2.5.3 Desired results

The main goal is to monitor the electricity consumption of the entire Campus area and also of each building and School individually. In the 2nd year prototype we will address predictions that are related to the individual building energy profiles (measured by the currently available sensors – measuring power, current and cumulative energy consumption).

Monitoring results

The time frame for the monitoring and reporting will be daily, weekly, monthly and yearly basis

- Electricity load (kW/time) of each Building
- Electricity load (kW/time) of each School
- Electricity load (kW/time) of the whole NTUA Campus
- Electricity consumption (kWh/time, kWh/m^2time) of each Building
- Electricity consumption (kWh/time, kWh/m^2time) of each School
- Electricity consumption (kWh/time, kWh/m^2time) of the whole NTUA Campus
- Thermal comfort level of the office: the dry bulb temperature in ^oC, relative humidity in % and illuminance lux.

Prediction results

The time frame for the prediction will be the first day for the 2nd year prototype. For the 3rd year we will experiment with weekly monthly and yearly horizons. It is expected that autoregressive methods will be more useful with the longer horizons.

- Electricity consumption (kWh, kWh/m^2) of each Building
- Electricity consumption (kWh, kWh/m^2) of each School
- Electricity consumption (kWh, kWh/m^2) of the whole Campus

2.6 Public Lighting in Miren

Envigence is working on use case in Municipality Miren, where we try to find the optimum installation of sensors and lights actuators to achieve the maximum impact on electricity savings. We are working on a different approach to find out how NRG4Cast tools can help reduce the energy consumption.

From various possible saving models we selected three: Moon impact, traffic, and dynamic electricity market, with which we can achieve the desired electricity savings.

We will compare 6 different types of installation:

1. Old lights – this was the previous installation
2. New lights (100%) – with new LED lights
3. New lights + profiles – LED lights with simple day/night dimming profiles
4. New lights + profiles + weather (moon) – moon impact
5. New lights + profiles + weather + traffic – traffic impact

6. New lights + profiles + weather + traffic + dynamic electricity market – monetary saving (impact if the lights could order the needed predicted electricity consumption daily)

What we want to achieve with such a model:

- day of the year and geolocation (sunrise and sunset) influence on our data streams
- Moon phases (moon lighting contribution could, as we found out in our test, result in 1-2% energy saving per month, as lights could be additionally dimmed when road illumination from the moon is high)
- City area (additional savings could be achieved with dimming the lights according the area in the city (residential area, business area, walkways, local streets ...) - savings could be around 25-35% per month on these lights
- Traffic flow (additional savings could be achieved with the lights according to the traffic flow (in night hours between 23pm and 4am the lights could be dimed to 60-70% of the original level - savings could be around 20-25% per month - if we want to use this we need to measure the traffic flow - now we use the 20-25% values because we do not have the data from the field)
- Day/night tariffs on electricity (additional economic savings could be achieved if we could buy electricity on the fly. In the night time the price of electricity is very low, but there is no system, with which we could buy the electricity. We expect that if such a system existed, we could get additional savings of around 5% on the price of electricity. With the reliable prediction model we could additionally save around 2-3%).

2.6.1 Available data

Untill now we have the following data from the each light:

1. Light operation (on/off)
2. Consumption
3. Dimming profiles
4. Dimming data
5. Type of light fixtures
6. Type of the streets
7. Moon phases – regarding the geo location
8. Weather forecast and weather conditions
9. Outdoor luminosity
10. Traffic data
11. Monthly electricity consumption per power station - from invoices
12. Area data – residential, industry, regional road …

2.6.2 Proposed Additional Features

Count	Sensor	Description
1-2	Sunrise, sunset	*Sunrise and sunset define the time at which the lights must be turned on.*

3-4	Moonrise, moonset	*Forecasts should include the same features as weather stations, although it is to be expected, that some would not be available.*
5	Moon phase	*Moon phase, combined with cloud cover can give an estimation for the needed additional illumination.*
6-12	Weather station: Miren	*1 additional weather station data, which includes 7 features (wind speed, wind direction, temperature, pressure, cloud cover, humidity, and visibility).*
13-19	Weather forecast: Miren	*Forecasts should include the same features as weather stations, although it is to be expected that some would not be available.*
14-15	Day/night tariff on electricity	
16-19	Traffic information	*Traffic flow information is the basic quantity that will help us estimate energy demand for the pilot case. Traffic information contains density, speed, and traffic flow information. The most relevant for us is traffic flow in the unit of cars/h.*

Table 1: List of additional features to model energy prices.

2.6.3 Desired results

With the Miren pilot we want to demonstrate the importance of legislation that allows dynamic classification of roads and on-line energy trading. NRG4Cast can contribute in the savings (energy and monetary) in the following steps:

- **Dynamic street classifications:** nowadays street classification is fixed. Even in the night, when the streets are empty, they retain the same class they had during the evening rush-hour, when the traffic is dense. With modelling traffic flow data, we can predict the class of the street in advance or even classify the street in real-time.

  A desired result of the modelling service would be a traffic flow profile for 1 day in advance (in 15-minute intervals). The prediction would be transformed into street classes and from there into the lighting profile.

- **Moon:** Full moon in a clear night can give even so much as 1.0lux of luminance. With the NRG4Cast system, which includes weather prediction, we can update the lighting profile with this information. Low clouds reflection from the large lighting polluters could also be taken into account.

  A desired result of the modelling service would be contribution of moon luminance during the night (in 15-minute intervals).

- **Energy trading:** nowadays energy consumers pay fixed prices for electricity. A dynamic market offers possibilities to save money, when you are able to estimate your consumption precisely. Preliminary information for the Miren use case suggests that precise estimation of energy consumption would yield in a 3.82% lower price of energy. When we can estimate energy profiles in advance, we can calculate the energy needed and we are able to take advantage of the lower prices.

  A desired result would contain overall consumption estimation per distribution point and (if needed) also for a single street light.

2.7 Electric Vehicles in Aachen

The smart charging algorithm is trying to develop a sufficient concept to charge several electric vehicles simultaneously without overloading the network. In order to develop this algorithm, it is necessary to gain a good insight into drivers' characteristics. Where, when, and how much the vehicle is charging. Therefore, a sophisticated approach is to collect data from several electric vehicles used in the Aachen area. However, the data acquisition turns out to have generate problems related to the data transmission between the car and the cloud system. Consequently, a second approach was proposed. In this approach the data of the charging stations within Aachen is used to predict the energy demand of electric vehicles. This has the advantage that the vehicles do not need to be fitted with a certain cloud box to communicate their data. However, the drawback is that vehicles that are charged at home cannot be monitored. In conclusion, receiving data from the charging stations in Aachen is a solid alternative of receiving the car data.

2.7.1 Available data

Receiving vehicle data (first approach for the smart charging algorithm):

The available NRG4Cast data regarding the smart charging algorithm are listed in detail in Chapter 6 of Deliverable 1.6. In general, the data consists of the total distance, current speed, state of charge, battery current, battery voltage, ambient temperature, longitude, latitude, altitude, and a timestamp. All the data are acquired from several electric vehicles and stored every minute. An extract of this data is displayed in Figure 5 and Figure 6. Figure 5 illustrates the transit of an employee from an office in Aacheen, to his home in Konzen, which is approximately 28km away. In addition to the route, the elevation of the track is shown. In fact, the altitude is 144m above sea level in Aachen and it continuously increases untill almost 580m in Konzen. Figure 6 again illustrates the elevation during the same way home (blue line, left axis). However, it also shows the state of charge of the battery (red line, right axis). An interesting point can be found at roughly 18:43. On the one hand it is a very steep part of the route and on the other hand it displays the matching decrease of battery load. This example shows the importance of a known track profile (here altitude) for a decent range estimation.

Figure 5: Tracked route from Aachen to Konzen displayed with an elevation colour schema

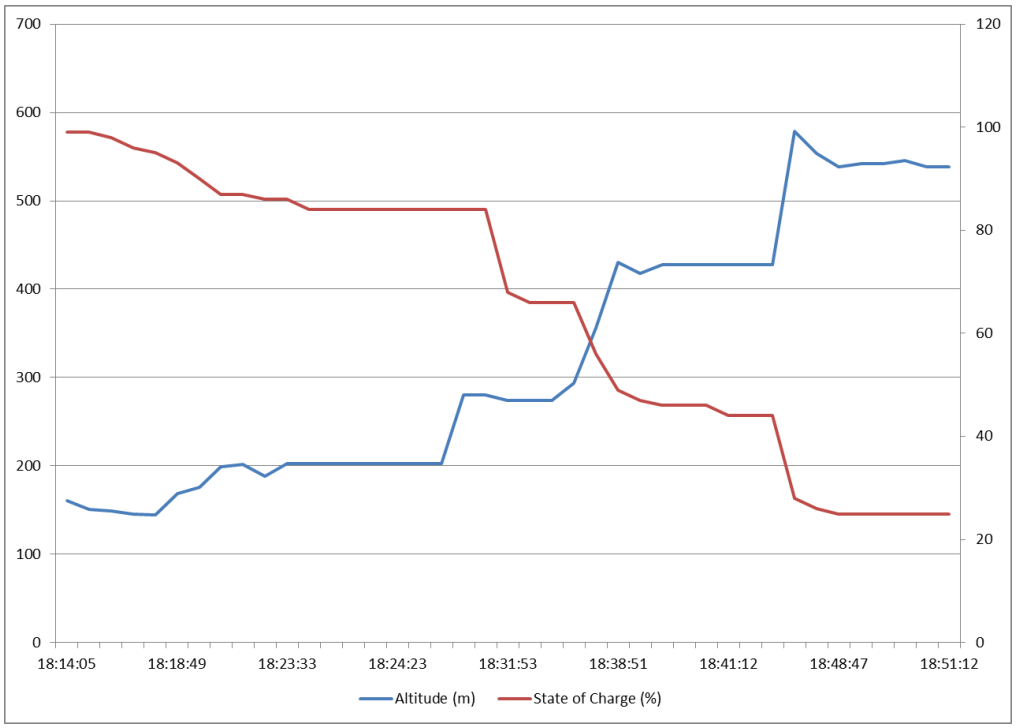


Figure 6: Altitude and State of Charge during a Trip from Aachen to Konzen

Receiving charging station data (second approach for the smart charging algorithm):

The second approach is to receive data from the charging stations allocated in the Aachen city centre. The needed dataset is similar to the received vehicle data and should contain the information, when, where, and how much energy is needed. This approach has the advantage that electrical cars need not be equipped with a sensor system. Thus vehicles without sensors can be considered for the smart charging algorithm. Additionally the data connection does not need to be wireless and data can be transmitted by already existing data infrastructure. The drawback of this approach would be, that electrical vehicles that are charged at home cannot be included into the forecast. The process of charging at home can be discussed by other partners, which deal with the energy demand of public or office buildings.

To obtain the data information, discussions with the local energy and grid provider Stawag/Stawag Netze are currently ongoing.

2.7.2 Proposed Additional Features

As the driver's behaviors and the range is influenced by a lot of external factors, the following external information sources should be considered (see also Table 2): The weather has a big impact on the battery of the electric vehicle. For example, during cold days, the capacity is limited in comparison to a hot day. In addition, the driver usually wants to heat his vehicle, which also costs battery power. Therefore it is especially relevant to obtain the temperature information. In addition, the weather forecast is important to estimate the battery capacity (and therefore the possible range) for the next days.

Apart from the weather, the range is obviously influenced by the traffic on the desired route. Especially for electric vehicles, this information is crucial, since a longer route might circumvent a traffic jam, but lead to problems regarding battery load. Furthermore, it is important to know when and how long the electric vehicle is using the electric light, since it also drains the main battery. Finally, the holiday seasons are interesting regarding the charging station distribution. Especially during the travel times, the demand regarding charging stations along the highway might be higher than on regular days.

Count	Sensor	Description
1	Weather stations: at least in North Rhine-Westphalia, better whole Germany	*The weather station should provide (especially) details about temperature and snow/rain situation.*
2	Weather forecasts for the stations above.	*Forecasts should include the same features as the weather station.*
3	Traffic	*These features should be calculated for Germany. Length of the useful daylight might have an effect on total energy consumption.*
4	Time features	*Time of daylight*
5	Holidays	*Information regarding public holidays and school holidays*

Table 2: Additional Features

Specific features for FIR

- Holiday Season: During the School Holidays and especially on the framing weekends, there is a lot of traffic on the road and the demand for electric charging stations is shifted from the cities to the highways. This differs from the every-day rush hour since the journeys the car takes are usually longer and it's not sufficient to charge a car only at the starting point or destination. This effect could also be visible on weekends and public holidays.

Example: *on Easter holidays in Germany, a lot of families drive towards Austria or Switzerland to go skiing. If a certain amount of those travellers use electric cars, the charging station demand (and therefore demand for sufficient electric power supply) along the southern high ways increases.*

- Events at a certain area/city: An event such as a football game, a concert, or a large convention will increase the demand of charging stations and power supply at a certain area of the city (assuming visitors are using electric vehicles). This demand is not regular, but usually predictable due to the schedule of events.

  Example: *During a football game in Cologne approximately 50,000 spectators are visiting the football stadium. A large amount are using private vehicles to go there. Consequently, the amount of charging station would be increased during those events.*

- Obstruction of the public transport: If the public transport in cities is obstructed e.g. by a strike, the energy demand is distributed differently, since people try to use alternatives to reach their destination. This especially occurs during rush hours.

  Example: *During a strike, a lot of commuters fall back on their own vehicles to reach their workplace. Therefore the distribution of energy demand differs from a usual rush hour.*

- Age of Battery: More on the technical site. The age of a battery affects its capacity. An older battery needs to be charged more often and therefore affects the energy demand.

  Example: *An old electric vehicle needs to be charged more often. During aging, the energy demand shifts to less needed power, but it is required at a higher frequency.*

- Frequency of battery usage: A battery has two factors that have an influence on the capacity during the aging process. One would be the age, the other one would be the usage.

- Network coverage: Taking the "influence on the data stream" from a technical side into account. Since the electric vehicles are moving objects and upload their measurements right into a cloud system, the data stream depends on the available network coverage.

  Example: *The network coverage in cities is well developed. However in rural areas there are some "blind spots", where it's not possible to send data. This could also appear when driving to foreign countries, where the network operators are not compatible.*

2.7.3 Desired results

The main information that are needed in this use case is:

- What amount of energy is needed in general
- Where the energy is needed at what location

Those two predictions build on following information that needs to be acquired first: The energy prognosis of each car and the behavioural pattern for using electric vehicles and charging stations.

Since electric vehicles can be used all day and night, the information needs to be acquired continuously. This is valid for both approaches (the vehicle data and the charging station). The prediction should aim for at least one day in advance.

2.8 Energy Prices in European Energy Exchange

European Energy Exchange AG[2] is the leading energy exchange in Central Europe [10]. It holds 50% of shares in the European Power Exchange spot market, called EPEX SPOT[3]. On EPEX SPOT there was a total of 346TWh of energy traded in year 2012 (Germany's total yearly production is estimated roughly to 600TWh[4]). As the

[2] http://www.eex.com/

[3] http://www.epexspot.com/

[4] http://en.wikipedia.org/wiki/Electricity_sector_in_Germany

laws of any market, the laws of EPEX SPOT market are based on variability of supply and demand of commodities traded.

Generation and consumption of electrical energy has to be in equilibrium to maintain the grid stability. There are big penalties (for consumers who order energy as there are for the grid owners) in case of redundant energy in the power grid. Variability of produced energy has its cause in intermittent energy sources, such as tidal, solar and wind energy. In the Central European context the latter sources are dominant. The expert in the EPEX SPOT trading suggested further analysis of impact of wind power production on energy prices, which is discussed in subsections 2.8.2 and 2.8.3.

2.8.1 Available data

The data available in the 1st year NRG4Cast Prototype (see Chapter 7 in [9]) has been expanded with a new on-line parser of the data. The newly available data is listed in Table 3. It contains two time-series, one containing traded quantity and the other trading price for a certain timestamp. Both time series are illustrated in Figure 7. Time series contain hourly data on quantity and electricity price. A number of aggregates is also computed for both time series (average, min, max, standard deviation, count and sum), for different time windows (relevant time windows for this use case would be daily and weekly). Prices are in units of EUR/MWh, quantity is also measured in energy units (MWh).

Sensor	Period
Electricity-Quantity	1. 1. 2005 – 30. 11. 2014
Electricity-Price	1. 1. 2005 – 30. 11. 2014

Table 3: Available data sources for EPEX SPOT.

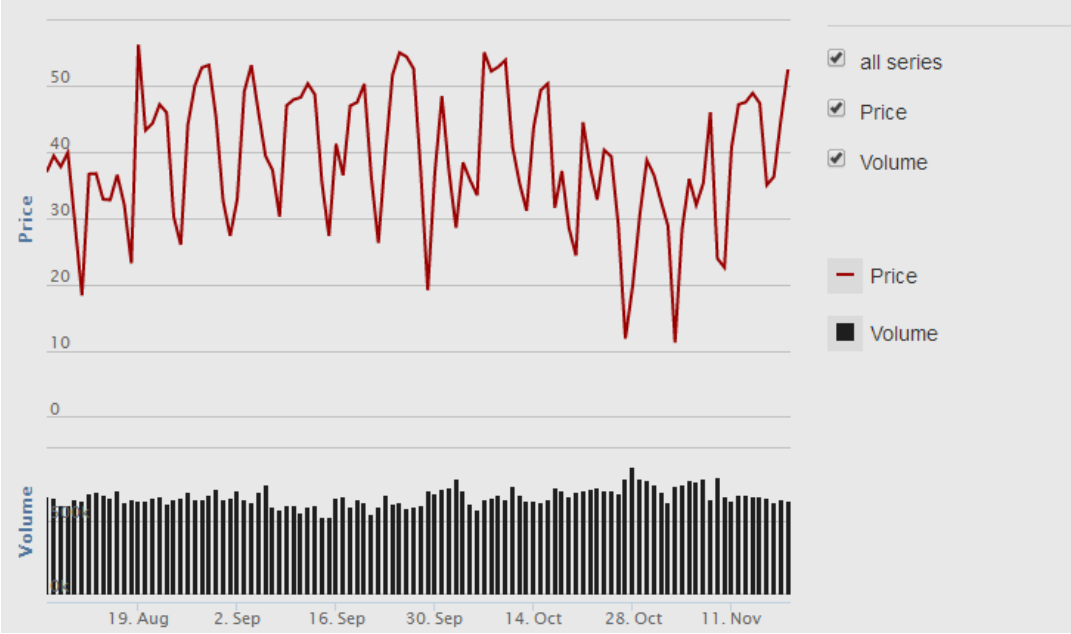


Figure 7: Energy volume and Electricity prices from EPEX SPOT.

2.8.2 Spot Market Trading Details[5]

For the purpose of the NRG4Cast use-case the only important thing is the closing of the energy spot market for the next day. New data is published every day at 12:00 for the next day. The requirement for the models is to have an estimate for the prices of the following day shortly before the official values are known.

2.8.3 Analysis of Wind Power in Germany

Wind farms play, in experts' opinion, a crucial role in the defining of the electricity prices. Wind energy is essentially cheap (comparable to fossil fuel generated energy) and renewable. Furthermore, there is no operational cost for producing electricity from wind energy, like there is with fossil fuels. Wind is a given type of energy, that either exist in a certain moment or not. When wind energy has a high market penetration, peaks have been observed, where only such wind farms have produced more than all required energy needs (for example Denmark for more than 90 hours in October 2013) [11].

Installed wind power capacity in Germany is rising substantially in last years (see Figure 8) and has reached the net share of almost 10% (see Table 3), whereas in certain states it is almost reaching 50%. In figure below installed capacity (in MW) is shown in red, average power generated in blue (in MW).

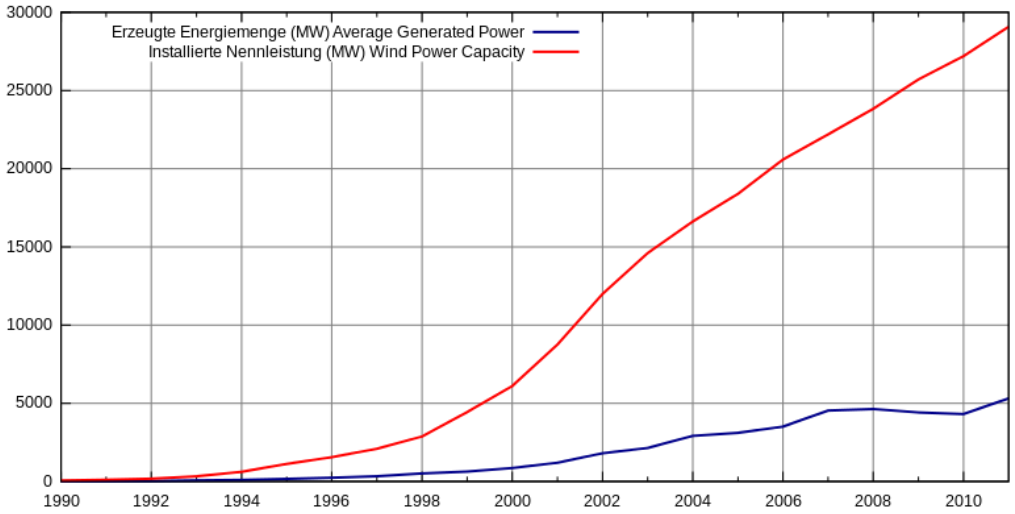


Figure 8: Wind power in Germany (1990 – 2011) [7].

[5] http://www.eex.com/en/trading/ordinances-and-rules-and-regulations

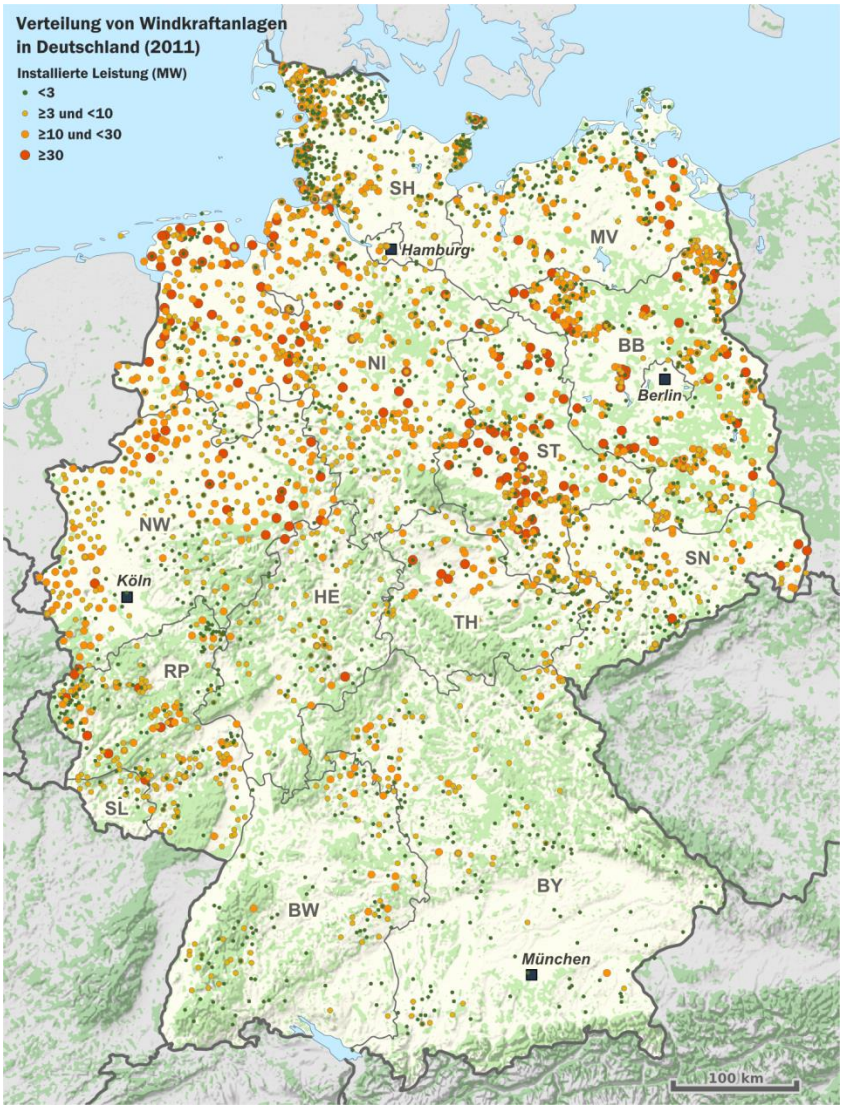


Figure 9: Map of German wind farms [7].

Most important regions/states with wind farms are listed in Table 4.

State	No. Turbines	Installed Capacity [MW]	Share in the net electrical energy consumption [%]
Saxony-Anhalt	2,352	3,642.31	48.11
Brandenburg	3,053	4,600.51	47.65
Schleswig-Holstein	2,705	3,271.19	46.46
Mecklenburg-Vorpommern	1,385	1,627.30	46.09
Lower Saxony	5,501	7,039.42	24.95
Thuringia	601	801.33	12.0
Rhineland-Palatinate	1,177	1,662.63	9.4
Saxony	838	975.82	8.0

Bremen	73	140.86	4.7
North Rhine-Westphalia	2,881	3,070.86	3.9
Hesse	665	687.11	2.8
Saarland	89	127.00	2.5
Bavaria	486	683.60	1.3
Baden-Württemberg	378	486.38	0.9
Hamburg	60	53.40	0.7
Berlin	1	2.00	0.0
offshore **North Sea**	31	155.00	
offshore **Baltic Sea**	21	48.30	
Germany Total	**22,297**	**29,075.02**	**9.9**

Table 4: Overview of wind farm capacity in different states in Germany [7].

2.8.4 Proposed Additional Features

Based on the map in Figure 9 and data in Table 4 we decided to include 7 more weather stations in the most important regions for wind energy production. Data about wind speed and wind direction should bare the most impact to modelling the energy prices, but also other features from weather stations should be included. Weather forecast also has a big impact on forming prices in the energy stock market. Therefore historical data should be obtained for weather forecast for the important areas for wind power production.

Count	Sensor	Description
1-49	Weather stations: Saxony-Anhalt, Brandenburg, Schleswig-Holstein, Mecklenburg-Vorpommern, Lower Saxony, Rhineland-Palatinate, North Rhine-Westphalia	*7 additional weather stations, which all include 7 features (wind speed, wind direction, temperature, pressure, cloud cover, humidity, and visibility).*
50-98	Weather forecasts for the stations above.	*Forecasts should include the same features as weather stations, although it is to be expected that some would not be available.*
99-103	Time features: Day in the Week, Work-free day, Hour of Day, Sunrise, Sunset	*These features should be calculated for Germany. Length of the useful daylight might have an effect on total energy consumption.*

Table 5: List of additional features to model energy prices.

The feature vectors should include also aggregates of the original features (daily, weekly, monthly) and should expand proposed additional features (sensors 1-49) with corresponding aggregates. It would be interesting also to experiment with more consecutive values of aggregates (like today, previous day, two days ago, and similar). Also, yearly dynamics could be taken into account, where features from exactly one (or more) year ago could be used.

2.8.5 Desired results

The features to model are the two quantities, representing the main two data streams in the use case. Those are:

- volume of traded energy (Electricity-Quantity)
- energy price (Electricity-Price)

According to the dynamic of the EPEX SPOT market trading finishes at 12:00 for the next day and finishes at 12:00. Trading of energy is performed in a resolution of 1 hour.

Main goal of the modelling would be to ensure **prediction** for the two stated quantities in a relatively short term (from 12 to 36 hours).

3 Feature Vector Generation

Modelling efficiency is rather more dependent on the input data than on the methods used. Good models require good data, meaning clean and reliable input data, as well as relevant and meaningful supporting properties. We have divided the input data into sensor data, weather data, weather forecast data, and additional properties data.

3.1 Additional Properties Generation

Additional features are listed in the table of features, which can be found in the Appendix of this document. The properties have been generated offline and imported into the NRG4Cast platform. The main granularity for all the features is 1 hour.

List of implemented properties:

- working hours/CSI
- working hours/NTUA
- day of the week (in numeric and boolean form for each day separately)
- a month (in numeric and boolean form)
- day of the month
- day of the year
- heating season/IREN
- heating season/CSI
- weekend
- holiday/IT
- holiday/SI
- holiday/GR
- holiday/DE
- day before and day after holiday (for all the pilot sites)

Properties have been calculated for relevant periods within the NRG4Cast (1. 1. 2009 until 31. 12. 2015). In a table in the appendix additional properties can be found that are not yet implemented in the NRG4Cast platform.

3.2 Additional Data Sources

3.2.1 EPEX On-line Service

The EPEX module is a service that scrapes data from the EPEX spot market webpage[6], transforms it into a desired JSON shape and sends it to the local QMiner Data Instance at http://localhost:9889 via a string query (defined in the Streaming API[3]).

Specifically, the service retrieves data from the HOURS table on

http://www.epexspot.com/en/market-data/auction/auction-table/2014-09-13/FR/<YYYY-MM-DD>/FR.

[6] http://www.epexspot.com/en/market-data/auction/auction-table/

There are 3 tables with energy spot market data on this site:

- FR, DE/AT, CH, for the spot markets of France, Germany and Switzerland respectively.

Entries $(i, j), i > 1 \ \& \ j > 2$ are the measurements, which the service scrapes. The 2nd column in the table is the unit of measurement, and the dates in the 1st row of the table and the times in the 1st column of the table together give us the date-times of respective measurements. The only two units of measurement are €/MWh (euros per megawatt hour), used for measuring the cost of a megawatt hour and MWh (megawatt hours), used for measuring total energy consumption.

An example packet of three measurements is included:


```
[{
  "node": {
      "id": "2",
      "name": "spot-fr", "subjectid": "spot-fr", "lat": 46.19504, "lng": 2.10937,
      "measurements": [{
          "sensorid": "4", "value": 2475, "timestamp": "2005-04-22T00:00:00.000",
          "type": {
              "id": "1", "name": "spot-fr-energy-price", "phenomenon": "total-energy",
              "UoM": "MWh"
          }
      },
      {
          "sensorid": "1", "value": 33.171, "timestamp": "2005-04-22T00:00:00.000",
          "type": {
              "id": "2", "name": "spot-fr-energy-price", "phenomenon": "energy-pricing",
              "UoM": "EUR/MWh"
          }
      },
      {
          "sensorid": "1", "value": 32.054, "timestamp": "2005-04-22T01:00:00.000",
          "type": {
              "id": "2", "name": "spot-fr-energy-price", "phenomenon": "energy-pricing",
              "UoM": "EUR/MWh"
          }
      },
      {
          "sensorid": "4", "value": 2711, "timestamp": "2005-04-22T01:00:00.000",
          "type": {
              "id": "1", "name": "spot-fr-energy-price", "phenomenon": "total-energy",
              "UoM": "MWh"
          }
      }]
  }
```


Figure 10: Streaming API JSON example for the EPEX module.

Using the EPEX module

There are mainly three important **storage** files for the service:

- errlog.txt: *This file contains errors that have occurred during runtime of the service.*
- log.txt: *Preventively stores scraped data from the EPEX site in case the service crashes. Basically if we have to re-run the service, we don't have to scrape all the data from the EPEX site again, but instead read from 'log.txt' and send to the local QMiner instance again.*
- timelast.txt: *Stores the date of the last time measurements were scraped. This allows us to know which measurements were last retrieved from the EPEX site if the service crashes.*

First start of the service:

When we first start the service executable, the file 'timelast.txt' will be generated, containing the date (YYYY-MM-DD) of the first measurements on EPEX. The above described files 'log.txt' and 'errlog.txt' will also be created. Then the service will start retrieving data from EPEX. Every time the service parses the data and sends in to the local QMiner instance, it will update 'timelast.txt' according to the date of the last measurement received and save the parsed JSON data into 'log.txt'. With this, whenever the service crashes, we can safely presume that all of the data scraped so far is in 'log.txt'.

Each subsequent start of the service:

After a crash of the service due to any reason. We can just re-run the executable. The service will check if 'timelast.txt' exists, and extract the date of the last scraped data. After this, it will send the whole content of 'log.txt' to the local QMiner instance, then begin to scrape new data from the EPEX page. After there is no more available data from EPEX, the service will go to sleep and wake up every hour to check whether there is new data to be scraped from EPEX.

Restarting the service:

If we don't wish to continue scraping where we left off after the last crash of the service, but would rather like to start from the beginning again for some reason, one should delete all of the storage files: 'log.txt', 'errlog.txt' and 'timelast.txt'.

Possible content of 'errlog.txt':

- QM Server Crash: the EPEX service has crashed due to the local QMiner instance crashing.
- Missing Measurement Warning: Some measurement are and will be missing on EPEX.

3.2.2 Forecast.IO

Most of the open weather services (or even national weather services) do not provide historical weather predictions. This is a major drawback when preparing models that depend on it. The only general enough service that keeps weather prediction is Forecast.IO[7].

Parsers for the Forecast.IO depend on the infrastructure for gathering weather data developed within the D2.3 – SensorFeed [3]. Weather forecasts have been taken for NRG4Cast relevant timespan for the relevant locations (6 in Germany and one at the site of each pilot). New forecasts are being scanned regularly and are being updated.

3.2.3 Weather (Weather Underground)

Weather services that have been included in the first year of the project unfortunately do not provide historical data. Therefore another service has been added: Weather Underground historical data. The service provides a simple CSV interface, which is freely accessible. The major drawback is that the service only contains min, max and average values of the relevant weather phenomena. Special parsers have been created that gather weather data and store it in the local CSV files. The data is then transferred to the QMiner instance using special support applications, which take advantage of the Streaming API.

3.2.4 Traffic Data

Traffic data is the basic data source for the Miren use case. The need for this data has not been foreseen in the first year of the project and has been added within the work in the WP3. Data is gathered from the services provided by opendata.si[8], which is parsing the services on the promet.si[9], which is a national traffic information service. Data is provided in huge JSON files including all the official traffic sensors in Slovenia. Only relevant sensors near Miren are extracted and used.

[7] http://forecast.io/
[8] http://www.opendata.si
[9] http://www.promet.si

3.3 Final Feature Vector Descriptions

In the subsections below (full) feature vectors for all the tested models are presented. This means, that all the features, that were identified as possibly relevant, are included. In the model selection feature pruning has also been performed. Each feature vector is represented by a table. Tables consist of data source name (feature, weather, weather prediction, or property), unit of measurements for a given source, values for each time represented by $X(t_1, t_2, \ldots)$, where X represents value of a data stream at times t_1, t_2, etc. Similar is also the notation at the aggregate selection, where aggregates are denoted by A. Relevant aggregates are the moving average (MA), exponential moving average (EMA), minimum (MIN), maximum (MAX), sum (SUM) and variance (VAR). Some aggregates also need the time window defined. Time windows are labelled with h (hour), 6h (6 hours), d (day), w (week), m (1 month = 30 days) and y (year). The last column in each table represents a number of feature vectors' values. Sum of all features is also calculated at the bottom.

A general remark is that feature vectors are quite big, but reduction of features has been performed according to the model evaluation.

3.3.1 CSI

Description: Each day at 15:00 energy demand per hour for the next day should be calculated.
Time: 0 time refers to the time of prediction generation and t refers to the time of the prediction.
Models: 24

				Aggregates							
	Name	UoM	Value (t)	Aggr(t)	MA	EMA	MIN	MAX	SUM	VAR	N
Sensor:	**total consumption**	kWh	X(0,h,d)	A(0)	6h,d,w,m		d,w	d,w		6h,d,w,m	15
	cooling	kWh	X(0, d, 2d)								3
	consumption cooling	kWh	X(0, d, 2d)								3
	data centre cooling	kWh	X(0, d, 2d)								3
Weather:	temperature	°C		A(0)	h, d, w		d, w	d, w		h, d	9
	windspeed	m/s		A(0)	h,d					h,d	4
	winddir	°		A(0)	h,d						2
	visibility	km		A(0)	d					d	2
	humidity	%		A(0)	h,d,w,m			d		h,d	7
	pressure	mbar		A(0)	d					d	2
	cloudcover[2]	%		A(0)	d,w					h,d	4
Weather	temperature	°C	X(t)								1
forecast:	windspeed	m/s	X(t)								1
	humidity	%	X(t)								1
	sky/cloudcover	%	X(t)								1
	winddirection	°	X(t)								1
Properties:	weekday		X(t)								1
	hour		X(t)								1
	month		X(t)								1
	dayOfWeek		X(t)								1
	weekend		X(t)								1
	working day		X(t)	A(t)					w		2
	working hour		X(t)	A(t)					d,w		3

	heatingSeason		X(t)								1
	holiday		X(t)	A(t)						w	2
	dayBeforeHoliday		X(t)								1
	dayAfterHoliday		X(t)								1
										Number of features:	**74**

Table 6: CSI feature vector schema.

3.3.2 NTUA

Description: Each day at 12:00 predictions for energy demand for the next day should be calculated hour-by-hour.
Time: 0 time refers to the time of prediction generation and t refers to the time of the prediction. Time resolution for sensor data is 1 hour (1h aggregates are therefore not included).
Models: 24

				Aggregates							
	Name	UoM	Value (t)	Aggr(t)	MA	EMA	MIN	MAX	SUM	VAR	N
Sensor:	current_l1[1]	A	X(0)								1
	current_l2[1]	A	X(0)								1
	current_l3[1]	A	X(0)								1
	energy_a[2]	kWh	X(0,h,d)								1
	demand_a[3]	MW	X(0)	A(0)	6h,d,w,m		d,w	d,w		6h,d,w,m	13
	demand_r[3]	kvar	X(0)								1
Weather:	temperature	°C		A(0)	h, d, w		d, w	d, w		h,d	9
	windspeed	m/s		A(0)	h,d					h,d	4
	winddir	°		A(0)	h,d						2
	visibility	km		A(0)	d					d	2
	humidity	%		A(0)	h,d,w,m			d		h,d	7
	pressure	mbar		A(0)	d					d	2
	cloudcover	%		A(0)	d,w					h,d	4
Weather	temperature	°C	X(t)								4
forecast:	windspeed	m/s	X(t)								3
	humidity	%	X(t)								3
	sky/cloudcover	%	X(t)								3
	winddirection	°	X(t)								2
Features:	weekday		X(t)								1
	dayOfWeek		X(t)								1
	month		X(t)								1
	working day		X(t)	A(t)					w		3
	working hour		X(t)	A(t)					d,w		4
	heatingSeason		X(t)		d,w,m						4
	strike		X(t)		d						2
	classes schedule		X(t)		d						2
	holiday		X(t)	A(t)					w		3
	dayBeforeHoliday		X(t)								2

	dayAfterHoliday		X(t)		2
				Number of features:	**88**

Table 7: NTUA feature vector schema.

Description of sensors:

1 - electric currents for 3 different points
2 - cumulative value of consumed energy
3 - active and reactive power

3.3.3 IREN (thermal)

Description: According to the section 2.3.3 models for each hour need to be prepared. Models should predict energy demand for each hour from 01 to 24 for one day in advance.
Time: 0 time refers to the time of prediction generation and t refers to the time of the prediction.
Models: 24

				Aggregates							
	Name	**UoM**	**Value (t)**	**Aggr(t)**	**MA**	**EMA**	**MIN**	**MAX**	**SUM**	**VAR**	**N**
Sensor:	**thermal production**[1]	MWh	X(0,h,d)	A(0,d,2d)	6h,d,w,m		d,w	d,w		6h,d,w,m	39
Weather:	temperature	°C		A(0)	h, d, w		d, w	d, w		h, d	9
	windspeed	m/s		A(0)	h,d					h,d	4
	winddir	°		A(0)	h,d						2
	visibility	km		A(0)	d					d	2
	humidity	%		A(0)	h,d,w,m			d		h,d	7
	pressure	mbar		A(0)	d					d	2
	cloudcover[2]	%		A(0)	d,w					h,d	4
Weather	temperature	°C	X(t)								4
forecast:	windspeed	m/s	X(t)								3
	humidity	%	X(t)								3
	sky/cloudcover	%	X(t)								3
	winddirection	°	X(t)								2
Features:	weekday		X(t)								1
	hour		X(t)								1
	month		X(t)								1
	dayOfWeek		X(t)								1
	weekend		X(t)								1
	working day		X(t)	A(t)					w		2
	working hour		X(t)	A(t)					d,w		3
	heatingSeason		X(t)								1
	holiday		X(t)	A(t)					w		2
	dayBeforeHoliday		X(t)								1
	dayAfterHoliday		X(t)								1
									Number of features:		**99**

Table 8: IREN (thermal plant) feature vector schema.

Description of sensors:

1 - Thermal production of the plant in MWh.
2 - Percentage of the sky covered by clouds.

3.3.4 Miren

Description: According to the section 2.3.3 models for each hour need to be prepared. Models should predict energy demand for each hour from 01 to 24 for one day in advance.
Time: 0 time refers to the time of prediction generation and t refers to the time of the prediction.
Models: 24

				Aggregates							
	Name	UoM	Value (t)	Aggr(t)	MA	EMA	MIN	MAX	SUM	VAR	N
Sensor:	**number**		X(0, d, -2d)								5
	speed	km/h	X(0, d)								2
	gap	s	X(0, d)								2
Weather:	temperature	°C		A(0)	h, d, w		d, w	d, w		h, d	9
	windspeed	m/s		A(0)	h,d					h,d	4
	winddir	°		A(0)	h,d						2
	visibility	km		A(0)	d					d	2
	humidity	%		A(0)	h,d,w,m			d		h,d	7
	pressure	mbar		A(0)	d					d	2
	cloudcover[2]	%		A(0)	d,w					h,d	4
Weather	temperature	°C	X(t)								4
forecast:	windspeed	m/s	X(t)								3
	humidity	%	X(t)								3
	sky/cloudcover	%	X(t)								3
	winddirection	°	X(t)								2
Features:	weekday		X(t)								1
	hour		X(t)								1
	month		X(t)								1
	dayOfWeek		X(t)								1
	weekend		X(t)								1
	working day		X(t)	A(t)					w		2
	working hour		X(t)	A(t)					d,w		3
	heatingSeason		X(t)								1
	holiday		X(t)	A(t)					w		2
	dayBeforeHoliday		X(t)								1
	dayAfterHoliday		X(t)								1
										Number of features:	**69**

Table 9: Miren traffic feature vectore schema.

Concrete implementation (if even needed) would depend on the legislation requirements (how much in advance the classification of a street could/would be changed and if changing it on-line would not be sufficient). The prediction horizon would also partly depend on the minimal interval of the profile change (which is 15 minutes at the moment).

3.3.5 Energy Stock Market (EPEX)

Description: As the spot market closes at 12:00 each day, we need to have predictions calculated at 11:00 each day, for one day in advance, hour-by-hour.
Time: 0 time refers to the time of prediction generation and t refers to the time of the prediction.
Models: 24

				Aggregates							
	Name	UoM	Value (t)	Aggr(t)	MA	EMA	MIN	MAX	SUM	VAR	N
Sensor:	energy_price	EUR/MWh	X(0,-d,-2d)	A(0)	w,m		w	w		m	8
	energy_quantity	MWh	X(0,-d,-2d)	A(0)	w,m		w	w		m	8
Weather:	temperature	°C	X(0)	A(0)	w		w	w		m	30
6 stat.	windspeed	m/s	X(0)	A(0)	d, w			d			12
	winddir	°									0
	humidity	%	X(0)	A(0)	w,m			w		w	30
	pressure	mbar	X(0)	A(0)	w						12
	cloudcover	%	X(0)	A(0)	w					w	18
Weather	temperature	°C	X(t)								6
forecast:	windspeed	m/s	X(t)								6
	humidity	%	X(t)								6
	sky/cloudcover	%	X(t)								6
	winddirection	°	X(t)								6
Features:	weekday		X(t)								1
	dayOfWeek		X(t)								1
	month		X(t)								1
	hour										0
	working day		X(t)	A(t)					w		0
	working hour		X(t)	A(t)					d,w		0
	holiday		X(t)	A(t)					w		0
	dayBeforeHoliday		X(t)								0
	dayAfterHoliday		X(t)								0
										Number of features:	**151**

Table 10: EPEX feature vector schema.

4 Data Mining Methods

The following section is dedicated to the short description of data mining methods that are viable for usage in the modelling of the pilot systems. Most of the methods are described only briefly. Our intention is to use these methods and not to study them in depth. Initial testing results, however, have indicated that model trees are the most successful method to be used with the pilots initially tested.

We have dedicated quite some effort of this deliverable to researching, implementing and testing such a method. A subsection dedicated to the Hoeffding trees is therefore much more detailed.

4.1 Methodology for Evaluation of the Methods and Models

The following subsection has been prepared with the goal to extend the QMiner evaluation module with a full set of possible error measures and to create a complete overview on the area, which is not present in the literature or on the internet.

4.1.1 Error Measures

When comparing different prediction methods a basic tool one needs is the error measure. The error measure can often be the decisive factor in the process of choosing the appropriate prediction method. In [27] a study is presented, where correlations among different rankings were calculated. Median correlation between different error measures in the study was only 0.40, which confirms the hypothesis above.

The same source gives the following guidelines for the use of different measures:

- Ensure that measures are not affected by scale (for example – when the value of predicted phenomena is near 0 – for example with temperature in the unit of degrees Celsius or Fahrenheit).
- Ensure error measures are valid.
- Avoid error measures with high sensitivity to the degree of difficulty.
- Avoid biased error measures.
- Avoid high sensitivity to outliers.
- Do not use R-squared to compare forecasting models.
- Do not use RMSE for comparison across series.

In table below the following quantities are used:

$$e_t = y_t - f_t,$$

$$p_t = \left(\frac{y_t - f_t}{y_t}\right), \text{ and}$$

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n} |y_i - y_{i-1}|},$$

where y_t is the measurement at time t, f_t prediction (forecast) at time t, n the number of prediction points. Note that e_t is the error of the forecast, p_t is the percentage (relative) error (some literature uses this measure in real percentage units, that is multiplied by 100, but we do not). The value q_t denotes a scaled error, proposed by [26].

Abbr.	Name	Formula	Description

ME	Mean error	$\frac{1}{n}\sum_{t=1}^{n} e_t$	*ME is likely to be small, as positive and negative errors tend to offset one another [25]. This measure can only tell us whether a forecast bias exists in the model.*
MAE	Mean absolute error	$\frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$	*MAE removes the original disadvantage of the ME with the introduction of the absolute value.*
MSE	Mean squared error	$\frac{1}{n}\sum_{t=1}^{n} e_t^2$	*MSE is also not strained with positive/negative error compensation like MAE, but it is a bit more difficult to interpret.*
MPE	Mean percentage error	$\frac{1}{n}\sum_{t=1}^{n} p_t$	
MAPE	Mean absolute percentage error	$\frac{1}{n}\sum_{t=1}^{n} \lvert p_t \rvert$	
sMAPE	Symmetric mean absolute percentage error	$2\sum_{t=1}^{n} \frac{\lvert e_t \rvert}{f_t + y_t}$	*This alternative to MAPE is limited to 2, but behaves better with low value items in the series. Low items can otherwise have infinitely high error rates that skew the overall error rate.*
MASE	Mean Absolute Scaled Error	$\frac{1}{n}\sum_{t=1}^{n} q_t$	*Proposed in [26]. Authors claim it is independent of the scale of the data, it is less sensitive to outliers as RMSSE and can be more easily interpreted. It is also less variable on small samples than MdASE.*
MAEP	Mean Absolute Error Percent	$\frac{\sum_{t=1}^{n} e_t}{\sum_{t=1}^{n} y_t}$	*MADP is preferable to MAPE as it does not skew error rates approaching zero.*
MRAE	Mean Relative Absolute Error		

Table 11: Different error measures based on mean.

Abbr.	Name	Formula	Description
R^2	Coefficient of Determination	$1 - \frac{\sum_{t=1}^{n}(f_t - \bar{y})^2}{\sum_{t=1}^{n}(y_t - \bar{y})^2}$	
PB	Percent Better		*Percent of cases where our method behaves better than a naïve (baseline) method (last or random walk). A baseline method found in the literature is the random walk method, in older literature (prior to 2000) also the last measurement method is used.*

Table 12: Special error measures.

There are numerous error measures [25][26][27] and many more than mentioned in Table 11 and Table 12. All of the measures using mean (which is sum, divided by the number of the data points, *n*) can also use the median (those are denoted by Md) or the geometric mean (denoted by G). For the mean and median measures the root operation is also applicable (e.g. RMSE is a widely used measure).

Basic measure	Mean	Median	Geometric Mean	Root mean	Root median
Error	ME	MdE	GME	-	-
Absolute error	MAE	MdAE	GMAE	-	-
Squared error	MSE	MdSE	GMSE	RMSE	RMdSE
Percentage error	MPE	MdPE	GMPE	-	-
Absolute percentage error	MAPE	MdAPE	GMAPE	-	-
Symmetric absolute percentage error	MASE	MdASE	GMASE	-	-
Symmetric squared error	MSSE	MdSSE	GMSSE	RMSSE	RMdSSE
Absolute scaled error	MASE	MdASE	GMASE	-	-
Absolute error percent	MAEP	MdAEP	GMAEP	-	-
Relative absolute error	MRAE	MdRAE	GMRAE	-	-

Table 13: Table of *derived* error measures.

Most of the measures can also be used as relative measures to a comparing method. Those measures are denoted by *Rel* [26] or *Cum* [27]. For example: $RelMAE = \frac{MAE}{MAE_b}$, where *b* stands for a benchmark method. Certain authors have also used logarithmic scale for relative measures. For example $\mathrm{LMR} = \log(RelMSE)$.

There are 34 error measures in the Table 13. Each of these measures can be used as a relative measure or further applied with the log function. All of the functions can be used in the Percent Better method. This means that we have noted 136 different functions in this subsection and – of course – the list is not yet complete.

The conclusion is that multiple error measures should be used, when determining the best candidates for a method/model. Different measures can also give different insight into possible problems with the models.

4.1.2 Choice of Error Measures for NRG4Cast

Although evaluation of the models should not be taken lightly in any scenario and especially not in the streaming scenario, there are certain properties of the NRG4Cast models that do not require the special caution mentioned in the paragraphs above. All the models in the NRG4Cast scenarios are prepared with the same dataset, evaluation takes place in the same interval and so on.

A standard set of measures has been taken into account:

- Mean Error (ME) – for checking possible bias of the models
- Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) – two main measures for evaluating the models
- Mean Squared Error (MSE) – has same relevance as RMSE, but the latter can be interpreted easier
- R^2 has been checked out of curiosity. We found that R^2 was as good a measure for the models, as were RMSE or MAE.

4.1.3 Error Measures in a Stream Mining Setting

The methods selection has been realized in an off-line manner. There was no need to implement the data stream evaluation measures. The problem of evaluating learning algorithms on a changing data stream is however discussed in the subsection 4.9.

4.2 Fine tuning of parameters

Certain methods are quite robust (linear regression and moving average), whereas others are hardly dependent on the choice of the parameters. Quite often a greedy scan over the parameter space is needed, to identify the relevant subspaces that need detailed exploration. As gradients of the method cannot be calculated directly, a bisection-like method is needed for finding the error measure minimum.

The golden-rule minimization has been implemented in the QMiner and used for fine tuning of parameters near the optimal spot. The following method provides minimization over only one parameter. Even if used consecutively on all the relevant parameter it does not guarantee convergence to the most optimal spot (even in the selected subspace).

This method has been used to optimize parameters for SVMR and NN.

```
function golden_minimization(func, min, max, tol, nmax) {
    var n = 1;
    var amse = func([min]);
    var bmse = func([max]);

    var a = min;
    var b = max;

    var phi = (1 + Math.sqrt(5)) / 2; // golden ratio

    while ((n < nmax) && (((b - a) / 2) > tol)) {
        x1 = b - (b - a) / phi;
        x2 = a + (b - a) / phi;

        x1mse = func([x1]);
        x2mse = func([x2]);

        if (x1mse > x2mse) {
            a = x1;
        } else {
            b = x2;
        }
    }

    return (a + b) / 2;
}
```

Figure 11: Golden ratio minimization algorithm, implemented in JavaScript for the QMiner platform.

Following are the subsections describing the possible methods in general. All of the methods have been used in the preliminary experiments that are documented in the Appendix of this document.

4.3 PCA

Short description of the method [12]:
Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated

variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables. The method was originally presented in [18].

Expected usage of the method:
PCA is expected to be used mainly in the phase of feature vector generation.

4.4 Naïve Bayes

Short description of the method [13]:
In machine learning, naïve Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. Naïve Bayes is a popular (baseline) method for text categorization, the problem of judging which category documents belonging to (spam or legitimate, sports or politics, etc.), with word frequencies as the features. With appropriate pre-processing, it can compete (in this domain) with more advanced methods including support vector machines.

Expected usage of the method:
Naïve Bayes is expected to be used in the classification phase, after the eventual discretisation of the dependent variable. It is expected to perform best on features generated by the PCA method.

4.5 Linear Regression

Short description of the method [14]:
In statistics, linear regression is an approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. In linear regression, data is modelled using linear predictor functions and unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, linear regression refers to a model in which the conditional mean of y, given the value of X, is an affine function of X. Less commonly, linear regression could refer to a model in which the median, or some other quantile of the conditional distribution of y, given X, is expressed as a linear function of X. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of y, given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.

Expected usage of the method:
Linear regression is expected to be used in the modelling phase in an attempt to generate an accurate linear model, which will predict the desired dependent variable from multiple independent ones – in this case multiple linear regression will be used. Moreover, simple linear regression could be used to examine the effects a single independent variable can have on the dependent variable.

4.6 SVM

Short description of the method [15]:
In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyse data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in

space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces. They were originally presented as support vector networks in [19].

Expected usage of the method:
SVM is expected to be used in the modelling phase both to predict the original dependent variable and also after its discretisation.

4.7 Artificial Neural Networks (ANN)

Short description of the method [16]:
In computer science and related fields, artificial neural networks (ANNs) are computational models inspired by an animal's central nervous system (in particular the brain), which is capable of machine learning as well as pattern recognition. Artificial neural networks are generally presented as systems of interconnected "neurons" which can compute values from inputs. For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a function (determined by the network's designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally, an output neuron is activated. This determines which character was read. Like other machine learning methods - systems that learn from data - neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition.

Expected usage of the method:
ANNs are expected to be used as an alternative modelling method to the other described methods.

4.8 Model Trees

Short description of the method:
Model trees are a sort of tree-based piecewise linear models. They combine decision trees with linear regression in such a way that a decision tree is initially constructed to partition the learning space. Linear regression is later used to fit the data from each partition. Model trees were first introduced in [21] and later extended in [22].

As we found out that model trees were quite effective in our preliminary evaluation (see Appendix) of the methods, we have made quite some effort to implement the Hoeffding trees in the QMiner open source platform. Description of the work is presented in the next subsection.

Expected usage of the method:
Model trees are expected to outperform the traditional linear regression method on our data.

4.9 Incremental Regression Tree Learner

This section describes the incremental regression tree learning algorithm implementation. The algorithm has been partially implemented within the NRG4Cast project and therefore this subsection goes in to much more detail than the overviews above. We present a very brief overview of theoretical foundations and then focus on implementation details.

4.9.1 Theoretical Introduction

Regression trees are well-known in the machine learning community. Intuitively, a regression tree represents a partition of the dataset so that elements that belong to the same partition have similar values (small variance) and elements from different partitions have different values. In general, this is a hard problem and in practice one usually uses greedy algorithms, such as [30], to learn regression trees.

In the data stream setting (road traffic counters, electric energy sensors, and so on) data arrives continuously and we have no control over the speed and order of arrival of stream elements. The size of the stream is unbounded for all practical purposes and we cannot fit the whole stream in the main memory. Classic regression tree learning algorithms are not applicable because they violate these constraints.

Recently, Ikonomovska et al. [28][29] adopted ideas from [31][32] to scale up one of the classical regression tree learning algorithm to data stream setting.

The algorithm uses standard deviation reduction [30] as an attribute evaluation measure. The selection decisions are based on a probabilistic estimate of the ratio of the standard deviation reduction of the two best-performing candidate splits.

Suppose S is the set of examples accumulated at a leaf of the tree. The standard deviation reduction for a d-valued discrete attribute A at this leaf is defined as sdr(A) = sd(S) - $p_1$sd(S_1) - ... - $p_d$sd(S_d), where S_i is the set of examples for which attribute A has the i-th value. Here the value p_i=|S_i|/|S| is the proportion of examples with the i-th value at attribute A and sd(S) denotes the standard deviation of the values of the target variable from the set S.

Let r = sdr(A) / sdr(B) be the real ratio and let r = SA / SB be the estimated ratio. Then Pr[r - r ≤ ε] ≥ 1 - δ, where ε = sqrt(log(1 / δ) / (2n)) and n is the number of examples in the leaf. If SA / SB < 1 - ε, then sdr(A) / sdr(B) < 1 with probability at least 1 - δ. (Note that sdr(A) / sdr(B) < 1 means attribute A is better than attribute B.) See [28][29] for more details.

```
Algorithm HoeffdingTree(S, δ, nm)
    Let T be an empty root node

    procedure Process (x) { // Update the tree using stream example x
        Traverse x down the tree T until it hits a leaft l
        Update sufficient statistics of nodes on the traversed branch
        Update unthresholded perceptron's weight vector
        if (n mod nm = 0) { // Recompute heuristics every nm examples
            Compute heuristic estimates for all attributes using sufficient statistics of
            leaf l
            Let SA and SB be the best and the second-best scores
            if (SB/SA)² < 1-log(1/δ)/(2n) { // Attribute A is "the best" with probability at
            least 1-δ
                Split the leaf l // Leaf l becomes a node with a children, if A has a values
            }
        }
    }

    function Predict (x) { // Predict value of example x
        Traverse x down the tree T until it hits a leaft l
        // Returns mean (iI) mean or (ii) uses unthresholded preceptron
        Use leaf model hL to compute prediction y=hL(x)
        return y
    }
```

Figure 12: Very rough outline of the *HoeffdingTree* algorithm variant for incremental learning of regression trees [28].

An interested reader can find more details regarding this family of algorithms in [28]. In the following sections we focus on our implementation.

4.9.2 Implementation

Our implementation is an extension of the classification Hoeffding tree learner [31][32], which was implemented as a part of the MobiS [39], OpComm, and Xlike projects and uses the same data stream and algorithm parameter format. The algorithm is available in QMiner [40].

To adapt the algorithm for regression, we need to do a nontrivial modification of the Hoeffding test, because we can use neither the information gain, nor the Gini index as an attribute heuristic measure. Instead, we follow [28][29] and use standard deviation reduction [30]. To find the best attribute, we look at the ratio of the standard deviation reductions of the two best-performing attributes. We use the Hoeffding bound [35] to confidently decide whether the ratio is less than 1 - ε, where ε=sqrt(log(1 / δ) / (2n)) and 1 - δ is the desired confidence. When this is the case, we have found the best attribute with probability at least 1 - δ. (Note that this does not mean the split will significantly improve predictive accuracy of the tree – all it means is that the attribute is probably the best, although it may not make sense to make the split.)

Consider a scenario when we have two equally good attributes with "very similar" standard deviation reductions. In such case, the ratio will be "almost 1" and the algorithm will be unable to make the split. To solve this, we introduce a tie-breaking parameter τ, typically τ=0.05, and consider two attributes equally good whenever ε<τ and the splitting criterion is still not satisfied [28]. The intuition is that when two attributes perform almost equally well, we do not care on which one we split.

The algorithm needs to efficiently (i.e. "fast enough") estimate standard deviation reduction of each attribute in every leaf periodically. We achieve this using a (numerically stable) incremental algorithm for variance [37] (p. 232) and formulas [36].

To handle numeric attributes, we implemented an E-BST approach as suggested in [28][29], and adapted the histogram-based approach described in [38]. We describe this in detail in the following subsections.

General Description

We give a brief description of the algorithm in the next paragraph, assuming the reader is familiar with the batch regression [30] or classification [43] tree learners.

The algorithm starts with an empty leaf node (the initially empty root node). Each time a new example arrives, the algorithm sorts it down the tree structure, updating necessary statistics at internal nodes. When the example hits the leaf, the algorithm updates statistics at the leaf and computes standard deviation reductions (SDRs) of all unused attributes. (Discrete attributes that are used along the given branch cannot be reused in the leaf of the branch. Note that this is not the case for numeric attributes.) If the attribute with the highest estimated SDR is "significantly better" than the second-best attribute, the algorithm splits the leaf on the best-performing attribute. (By "sorts down the tree" we mean that the algorithm checks what attribute the current node splits on, and passes the example to the appropriate subtree, according to the value of the attribute of the current example.) The algorithm uses Hoeffding's inequality to ensure that the attribute it splits on is "the best" with desired probability (technically, with probability at least 1 - δ, for a user-defined parameter 0 < δ < 1).

It seems that setting δ = 1e-6, grace period to 300, and τ = 0.005 performs reasonably well.

Handling Numeric Attributes

When decision tree splits a leaf on a d-valued discrete attribute, it creates d new leaves that become children of that leaf. If the attribute is numeric, there is no way to make such a split. The usual solution is to discretize numeric attributes in the pre-processing step. This is clearly unacceptable in the data stream model. Instead, we perform an on-the-fly discretization using the histogram-based approach and binary-search tree approach.

The idea behind the histogram-based approach is to initialize a histogram with a constant number of bins (we use a hard-coded constant of 100 bins) in each leaf of the tree, for each numeric attribute. Each histogram bin has a unique key, which is one of the attribute values. We use the first 100 unique attribute values to initialize bins of the histograms. All subsequent stream examples that pass the leaf with this histogram will affect the closest bin of the histogram. In each bin we incrementally update the target mean, target variance, and the number of examples, using the algorithm suggested by Knuth [37]. (The algorithm is numerically stable.) To determine a split point, we use formulas [36] that allow us to compute the variance of a union of bins from variances we keep in bins. This suffices to determine the best split point. The problem with this approach is that it is sensitive to the order of arrival of examples (skewed distributions are problematic), that is not clear how much bins one should take, etc. The advantage is that the approach works very fast and uses only a constant amount of memory (independent of the data stream).

Another option is the so-called E-BST (extended binary search tree) discretization, proposed by [29]. Essentially it is a binary search tree with satellite data (satellite data are statistics needed to estimate standard deviation reduction for each split point) for each numeric attribute in every leaf of the tree. The keys are unique values of the numeric attribute(s?). Each node holds the number of examples with the attribute value less than, or equal to the key of the node, sum of the target values of these examples, and the sum of squares of the target values of these examples. Similar statistics are stored for examples with attribute values greater than the key of the node. These three quantities suffice to compute the standard deviation (see [28][29]). Determining the best split point corresponds to in-order traversal of the binary search tree [29]. The problem is that this technique is memory-intensive (it is essentially a batch method, as it remembers everything), and has potentially slow worst-case insertion time (linear in the number of keys). (Note that insertion can be made fast using balanced binary search trees, such as AVL tress or red-black trees [41] (p. 308), with worst-case insertion times logarithmic in the number of keys. To save memory [28][29] suggest disabling bad splits. A split h_i is bad if $sdr(h_i)/sdr(h_1) < r - 2\varepsilon$, where $r=sdr(h_2) / sdr(h_1)$ and h_1 and h_2 are the best and the second-best split, respectively.

Stopping Criteria

Note that the algorithm, as described, doesn't care whether it makes sense to make the split. All it cares is whether the attribute that looks the best is the best. So it is important we stop growing the tree at some point. We address this via several threshold parameters.

Our implementation controls growth via the standard deviation reduction threshold parameter (sdrTresh) and the standard deviation threshold parameter (sdTresh). We only split the leaf, if the standard deviation of the target variables of the examples in the leaf exceeds sdTresh and if $sdr(A) \geq sdrTresh$, where A is the attribute with the highest standard deviation reduction. We assume $sdTresh \geq 0$ and $sdrTresh \geq 0$. By default (if the user doesn't set the parameters) we have sdTresh = 0 and sdrTres = 0. The implementation also controls the number of nodes in the tree. When the tree size exceeds maxNodes - 1 (maxNodes is a user-defined threshold), the learner stops growing the tree. By default maxNodes=0 and in this case there is no restriction on the size of the tree.

We typically want small threshold values, for instance sdrTresh = 0.1, or even sdrTresh = 0.05, to prevent useless splits and make sure we are not limiting the algorithm too much. In general, however, the value of the threshold parameters depends on the following scenario: We might want small, interpretable trees (higher threshold, to prevent growth), or we might want to let the tree grow and “maximize” prediction accuracy (lower threshold, to allow growth).

Change Detection

When a process that generates stream examples changes over time, we say the data stream is *time-changing*. When the current model no longer reflects the concept represented by the stream examples, we say that *concept drift* has occurred [34].

In classification, the CVFDT algorithm [32] periodically scans the tree for nodes that do not pass the Hoeffding test anymore – at each such node, it start growing an alternate tree. Whenever the best-performing tree at that node is one of the alternate trees, the algorithm uses it in place of the main one, deleting all other trees at that node. Note that waiting for the alternate tree to outperform the main one enables granular local adaption of the current hypothesis.

Instead of adapting sufficient statistics according to sliding window, we implemented the Page-Hinkley (abbr. PH) test, as described in [28][29][33][34]. The main idea is to monitor the evolution of error at each node of the tree. If the data stream is stationary, the error won't increase as the tree grows. If the error start increasing, we start growing an alternate tree at that node, since this is a sign that the model no longer reflects concept in the stream. We track the error of all nodes using prequential error estimation (see next section) and all PH-Period examples, we periodically compute Q-statistic of the error of the main tree, and the best-performing alternate tree. If the Q-statistic is positive (meaning the original tree has higher error than the alternate one), we swap the alternate tree with the main one and delete all other trees at that node.

We now describe the Page-Hinkley test (adapted from [28]). The PH test detects abrupt changes in the average of a Gaussian signal. At any point in time, the test considers a cumulative sum m(T) and the minimal value of the cumulative sum M(T) = $\min_{t=1,2,\ldots,T}$ m(t), where T is the number of observed examples. The cumulative sum is defined as the cumulative difference between the monitored signal x_t and its current mean x(T), corrected with an additional parameter α:

$m(T) = \sum_{i=1}^{T}(x_i - \bar{x}(T) - \alpha)$, where

$\bar{x}(T) = \frac{1}{T}\sum_{i=1}^{T} x_i$.

The parameter α denotes the minimal absolute amplitude change that we wish to detect, and should be adjusted according to the expected standard deviation of the signal.

The PH test monitors the difference PH(T)=m(T)-M(T) and triggers an alarm whenever PH(T)>λ for a user-defined parameter λ, which corresponds to the admissible false alarm rate.

Our implementation takes an additional parameter phInit (typically phInit = 500) and starts using the PH test for the change detection at the node after the node saw at least phInit examples, so that the mean "stabilizes". We compute the mean x(T) using an/the incremental algorithm [37].

Evaluation and Comparison of Stream Learning Algorithms

In this section we briefly discuss how to evaluate and compare stream learning algorithms.

Classic evaluation techniques are inappropriate in the data stream setting, especially when one is dealing with time-changing data streams. The reason for this is concept drift, which refers to an online supervised learning scenario (in our case mining regression trees from the data stream), where the relation between the input data (in our case a vector of attributes) and the target variable (in our case a numerical "label") changes over time [34].

Classic measures give equal weight to all errors. However, when dealing with time-changing data streams, we are mainly interested in the recent performance of the model.

Gama et al. [33] suggest using prequential fading error estimation, also known as "test-then-train", defined as follows. Let A be the learning algorithm, let y_i be the target value at time point i and let y_i be the value the learner predicted. We then define the loss function L_A (i) = L(y_i, y_i). Given a fading factor 0 < α <= 1, typically α = 0.975, we define S_A(i) = L_A (i) + αS_A (i-1). Whenever the learner receives a new example from the stream, it computes the loss for the example, updates the error, and then uses the example to train the model. (Hence the name "test-then-train".) Note how the factor α controls which errors we consider relevant – a small α corresponds to taking into account only very recent errors, while α = 1 corresponds to taking into account all errors. The loss function L_A (i) is usually a squared difference L_A (i) = (y_i - y_i)^2, an absolute difference L_A (i) = |y_i - y_i|, or something similar.

Let A and B be learning algorithms and let S_A (i) and S_B(i) be their losses at time point i. The Q-statistic at time point i is defined as Q_i (A,B) := log(S_A(i) / S_B(i)). One can interpret it as follows:

- Q_i(A,B) > 0 indicates A outperforms B at time point i;
- Q_i(A,B) < 0 indicates B outperforms A at time point i;
- Q_i(A,B) = 0 indicates a tie.

If the Q-statistic value is extremely small, we can hardly say that one learner is better than the other. One way to address this is by introducing a small threshold and indicate a tie if |Q_i(A, B)| does not exceed the threshold. If the Q-statistic is positive "most of the time", we say A performs better than B; similarly, if Q-statistic is negative "most of the time", we say B performs better than A.

Sometimes we can see that one learner dominates the other by eyeballing the graph. When this is not the case, we can apply the Wilcoxon test [42]: the null hypothesis says that the vector of Q-statistics (Q_1(A, B), Q_2(A, B), ...) comes from a distribution with median zero. Whenever we reject the null hypothesis, one of the learners is better, and the sample median tells us which one.

4.9.3 Algorithm Parameters

Our implementation comes with many parameters to guide the learning algorithm. Below is a brief description of each parameter.

- The parameter gracePeriod is a positive integer that corresponds to nm in Figure 2. Because computing heuristic estimates (in our case standard deviation reductions of all the attributes) is the most expensive operation, the algorithm does these every nm examples. We typically set gracePeriod between 200 and 300.
- The parameter splitConfidence is a real number from the open unit interval that corresponds to 1-δ in Figure 2. Intuitively, it is the probability that the split made by the algorithm is the same as the split that the batch learner would make on the whole stream. We typically set splitConfidence to 1e-6.
- The parameter tieBreaking is a real number from the open unit interval. When the attribute with the highest heuristic estimates have similar scores, the algorithm can't tell them apart – the quotient SA/SB will be very close to 1 and the algorithm might never make split. In practice we don't care on what attribute we split if the two have similar heuristic estimates. We solve this using the tieBreaking parameter.
- The parameter driftCheck is used in certain change-adaption modes for classification. The algorithm will check split validity of the node every driftCheck examples to see whether the split is no longer valid.
- The parameter windowSize is a positive integer that denotes the size of the sliding window of recent stream examples that the algorithm keeps in the main memory. The regression tree that the algorithm maintains reflects the concept represented by these most recent examples.
- The parameter conceptDriftP is a boolean value that tells the algorithm whether to use change detection or not.
- The parameter maxNodes is a positive integer that denotes the maximum size of the tree. The algorithm stops growing the tree, once the tree has at least maxNodes nodes.
- The parameter regLeafModel is a string that tells the algorithm which leaf mode to use. Currently there are two leaf models available: (i) when regLeafModel=mean the algorithm predicts mean value of examples at given a leaf; (ii) when regLeafModel=linear the algorithm fits unthresholded perceptron in the leaf.

- The parameters sdThreshold and sdrThreshold are the minimum standard deviation and the minimum standard deviation reduction, respectively, needed for the algorithm to split the attributes – when SD and SDR are less than these thresholds, the algorithm will not consider making the split.
- The parameters phAlpha and phLambda are Page-Hinkley test parameters that corresponds to α and λ in the text above, respectively.
- The parameter phInit is the minimal number of examples needed in the subtree for the algorithm to run change-detection on that subtree.

5 Results from method selection experiments

The following methods have been compared in the experiments:

- Linear regression (LR)
- Support Vector Machine Regression (SVMR)
- Ridge Regression (RR)
- Neural networks (NN)
- Moving average multiple models (MA)
- Hoeffding trees (HT)

For algorithms, where this makes sense (LR, RR, HT), some feature pruning has also been done. As our predicted features are immune to the concerns mentioned in Section 4.1.1, we have been looking at the following measures:

- Mean Error (ME)
  Showing us whether there is some bias introduced to our models.
- Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE)
  Showing us the average/expected value of the prediction.
- Mean Squared Error (MSE)
- The R^2 measure

The best model has been chosen taking into account all of the measures.

According to the feature selection, there are the following universal denominations in the text:

- ALL - all features are used (as mentioned in Section 3.3.)
- AR – autoregressive – variable (to be predicted) and its historical/aggregated values
- S – sensor data – all the sensor data
- W – weather – weather values were used
- F – forecast data – all the forecasts
- P – static properties

For example:

> *LR-AR+W+S or LR-ARWS means linear regression method with autoregressive, weather and sensor features.*

If no parameters values are mentioned with the model, **default parameters** have been taken. They **are marked** in the Notes for each of the model **in the first subsection (EPEX)** below. If other parameters have been used, they are unambiguously shortened and the values are added in the parenthesis. In the case of neural networks the first number/sequence of numbers describes the inside layers of the neural network. For example (12-4-3) would mean that the neural network has 5 layers. Starting with the input layer of the size of input parameters, followed by three inside layers with 12, 4, and 3 neurons, respectively and one output layer with 1 parameter (it is the scalar that we want to predict in the NRG4Cast).

We have also tried to interpret some of the results, which is not in the scope of this deliverable. More in-depth analysis will be provided in D5.2 in year 3 of the NRG4Cast project.

5.1 EPEX

Valid fused data interval: 4.5 years (from April 2009 until October 2014)

Learning period: 3 years

Evaluation period: 1.4 years

Total number of features: 133

Number of models: 24

Feature to predict: Energy Prices (spot-ger-energy-price)

Requirement: Models need to be run every day at 11:00. They need to predict energy prices for the next day – hour by hour.

A bit surprisingly our models work quite well at predicting spot market values, as can be seen in Figure 13.

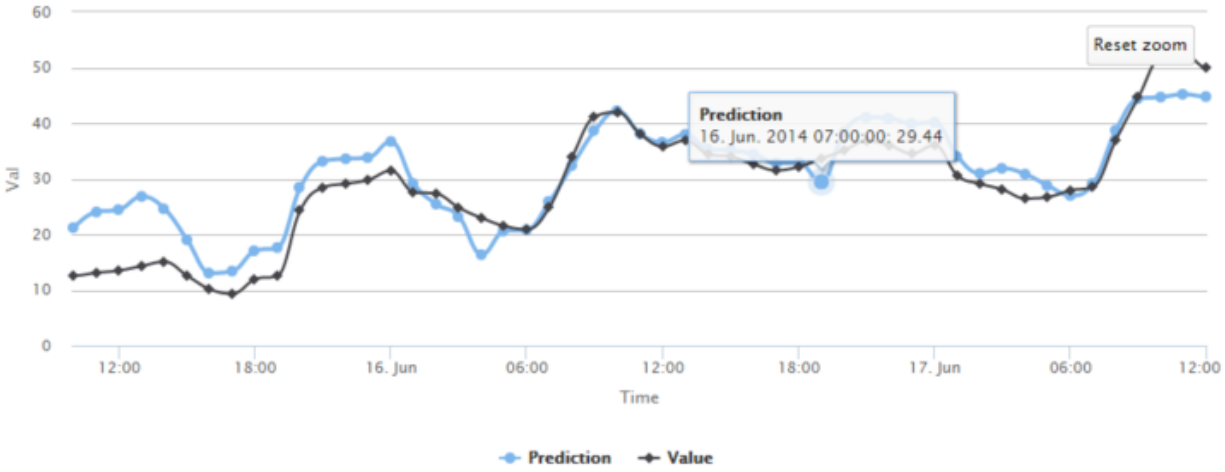


Figure 13: Example of prediction for EPEX problem (LR-ALL).

Results from the experiments can be seen in Table 14. One of the most safe algorithms behaves the best here – linear regression. LR shows that there are possible problems (either with data, its relevance or with over-fitting) with weather data. The best model uses auto-regressive and sensor values, weather prediction, and additional properties. It was a little bit worrying that neural networks were not competitive here at all. We have had quite some problems with the SVMR in the beginning too, but a wider scan of the parameter space results steered us in a better direction. The LR is, however, still the dominant method here.

Model	ME	MAE	MSE	RMSE	R2
LR-AR+S+F+P	-0,53	6,22	73,7	8,59	0,71
LR-ALL	-0,28	6,31	74,7	8,64	0,70
SVMR-ALL (c=0.037, eps=0.034)	1,01	6,93	79,9	8,94	0,63
SVMR-ALL (c=0.02, eps=0.04)	-3,07	7,23	92,2	9,60	0,63
LR-AR+S+F	-0,22	7,55	106,0	10,29	0,58
LR-AR+S+P	-0,73	7,49	106,4	10,32	0,58
SVMR-ALL (c=0.02, eps=0.1)	-0,73	8,64	124,6	11,16	0,51
SVMR-ALL (c=0.01, eps=0.1)	-0,32	8,82	129,7	11,39	0,49
LR-AR+S+W	-0,13	8,62	135,6	11,64	0,46
LR-AR+S	-0,54	8,66	137,9	11,74	0,45
LR-AR	0,15	9,06	149,8	12,24	0,41
HT-AR+S+F+P (sc=1E-1, tb=1e-4)	-2,29	9,65	179,9	13,41	0,29
NN-AR+S+F+P (4; lr=0.05)	0,17	10,03	181,0	13,45	0,28
NN-AR+S+P (4-3;lr=0.05)	0,03	10,21	187,6	13,70	0,26

NN-AR+S+F+P (4-3; lr=0.05)	0,00	10,23	188,1	13,71	0,25
HT-AR+S+F+P (sc=3E-1, tb=1e-4)	-5,25	10,13	191,1	13,82	0,24
HT-AR+S+F+P	-1,76	10,28	192,6	13,88	0,24
HT-ALL	-1,39	10,33	193,0	13,89	0,23
HT-AR+S+F+P (sc=3E-2, tb=1e-4)	-0,93	10,10	195,8	13,99	0,22
HT-AR+S+F+P (sc=1E-2, tb=1e-4)	-0,75	10,09	196,4	14,02	0,22
HT-AR+S+P	-0,87	10,36	197,5	14,05	0,22
NN-AR+S+F+P (12-4-3; lr=0.1)	0,08	10,74	212,1	14,56	0,16
HT-AR	-0,07	10,71	216,4	14,71	0,14
HT-AR+S	-0,12	10,84	219,7	14,82	0,13
HT-AR+S+F+P (sc=9E-1, tb=1e-4)	-7,31	11,28	220,8	14,86	0,12
NN-AR+S+F+P (4; lr=0.1)	0,11	11,08	228,4	15,11	0,09
NN-AR+S (1)	0,16	11,12	232,4	15,25	0,08
NN-AR+S (2)	0,11	11,55	249,7	15,80	0,01
MA (365)	-8,36	12,50	263,7	16,24	-0,05
NN-AR+S (3)	0,00	12,50	301,6	17,37	-0,20
NN-ALL (5-3)	0,01	12,69	311,9	17,66	-0,24
NN-AR+S+P (4-3)	0,00	12,73	321,1	17,92	-0,27
NN-AR+S+F+P (4-3)	0,03	12,88	328,1	18,11	-0,30
NN-ALL (15-5-3)	0,08	13,00	338,1	18,39	-0,34
NN-ALL (30-5-3)	0,10	13,01	341,1	18,47	-0,35
NN-AR+S+F+P (12-3)	0,11	13,00	346,6	18,62	-0,38
NN-AR+S+F+P (4)	0,03	14,60	440,5	20,99	-0,75
NN-AR+S (4)	0,02	14,85	481,3	21,94	-0,91
NN-AR+S+P (4)	0,01	15,21	485,5	22,03	-0,93
NN-AR+S (5)	0,07	20,67	1003,9	31,68	-2,98
NN-ALL (5)	0,07	21,11	1083,5	32,92	-3,30
NN-AR+S+F+P (4; lr=0.2, m=0.6)	-0,09	21,25	1599,0	39,99	-5,34

Table 14: Comparison of models in EPEX use-case.

5.1.1 Linear Regression Notes

Interestingly, linear regression has proven to be quite a good method for this problem. With added weather forecast data the algorithm improved significantly. Below is an overview of hourly linear regression models in a setting LR-ALL. Three most interesting values from the table are also depicted in the Figure 14.

model	ME	MAE	MSE	RMSE	R^2
0	-0,419	4,387	31,952	5,653	0,635
1	-0,316	4,402	34,626	5,884	0,635
2	-0,293	4,542	34,789	5,898	0,499
3	0,103	5,224	48,325	6,952	0,426
4	0,349	5,694	65,577	8,098	0,406
5	0,367	5,694	69,961	8,364	0,385
6	0,434	5,433	55,255	7,433	0,484
7	-0,073	5,434	54,332	7,371	0,449
8	0,368	6,365	76,987	8,774	0,511
9	0,206	6,712	79,055	8,891	0,715

10	0,157	6,575	78,669	8,870	0,739
11	-0,821	6,875	84,909	9,215	0,689
12	-1,057	7,073	81,458	9,025	0,658
13	-1,007	7,162	83,096	9,116	0,623
14	-0,946	7,314	95,662	9,781	0,574
15	-0,782	7,009	84,890	9,214	0,601
16	-0,858	7,076	96,282	9,812	0,604
17	-0,303	7,060	107,119	10,350	0,601
18	-0,252	6,800	101,064	10,053	0,626
19	-0,573	7,234	98,444	9,922	0,714
20	-0,923	8,362	119,407	10,927	0,661
21	-0,644	7,078	89,728	9,472	0,621
22	0,432	6,232	63,785	7,987	0,562
23	0,152	5,810	56,355	7,507	0,562

Table 15: Comparison of models for LR-ALL.

The chart below shows that models during the night are more accurate. This is, however, expected as spot market prices are more stable during the night (there are less unforeseen phenomena). The absolute value of prices is also much smaller during the night.

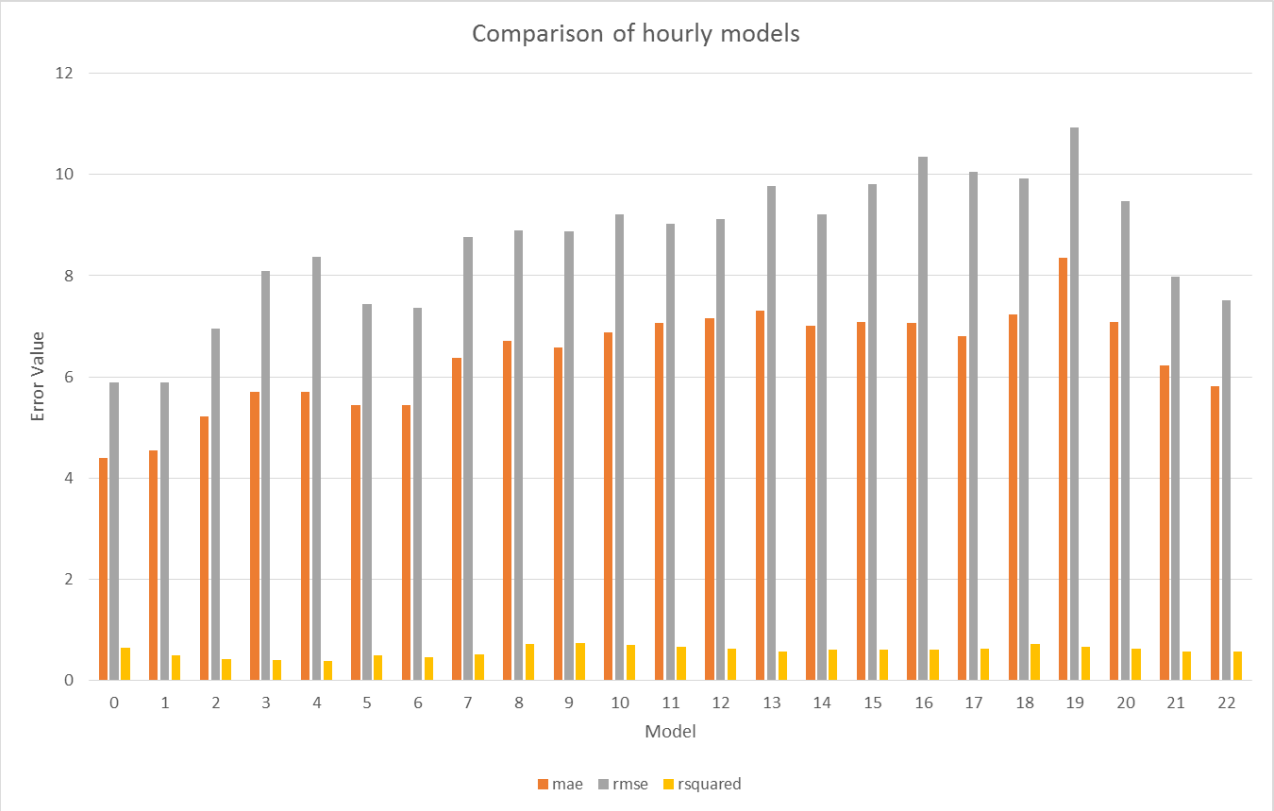


Figure 14: MAE, RMSE and R^2 per hourly LR-ALL model in the EPEX use case.

Heat map from the Figure 15 tells an interesting story. Red and green fields depict the feature values that influence the model outcomes the most. Most dominant features are bolded; the most dominant are also coloured in red. On the Y axis we have different features and on the X axis we have all the hourly models – one by one.

Figure 15: Heat map of linear regressions weights for full feature vectors in the EPEX use case.

It is interesting to see that wind bearing values are quite a significant feature. Much more than the wind speed. This confirms our hypothesis that wind energy is the dominant energy price changing actuator. With the wrong wind direction wind turbines do not function.

It is interesting to observe

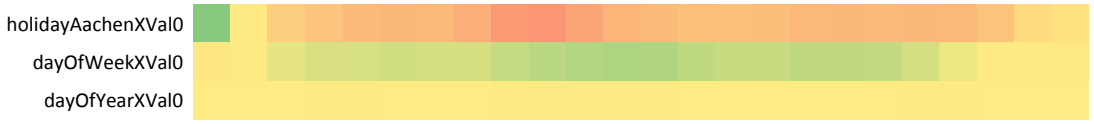


Figure 16. Although the static properties do not play any significant role in the full feature set, they are quite significant in the scenario where they are the only supporting features. This is something that is expected and quite nice to see. It shows that a working days or holidays are quite important features, especially between the working hours (columns from 8 to 18 in the figure below). Also the part of the year plays a significant role as well as the day of week.

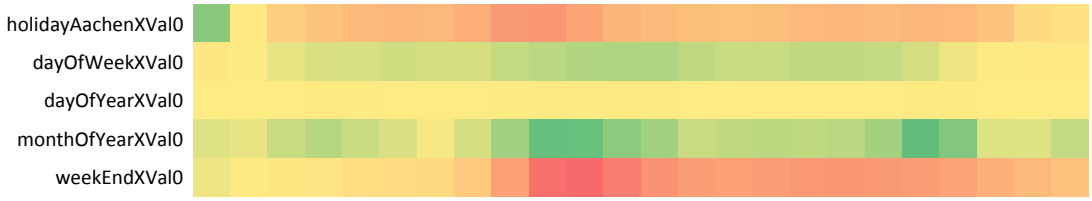


Figure 16: Heat map with values of LR weights for ARSP case in EPEX use case.

5.1.2 Moving Average Notes

Moving average usually behaves better in the setting with a bigger prediction horizon. In the EPEX scenario this is not the case.

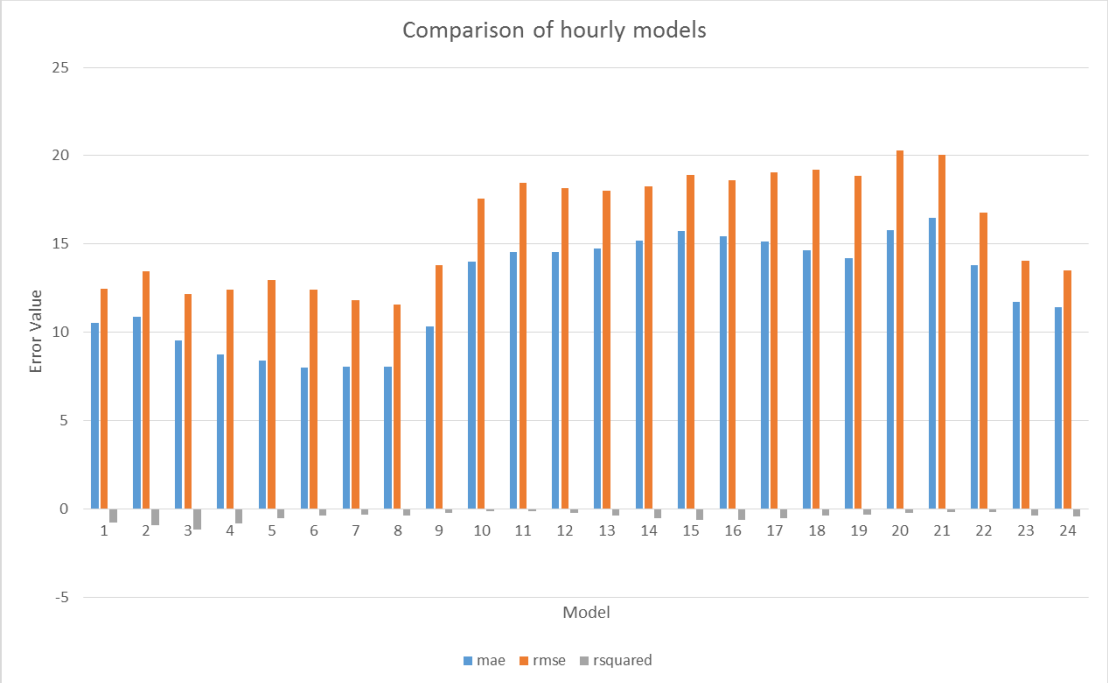


Table 16: The moving average model comparison.

5.1.3 Hoeffding Tree Notes

Default parameters for Hoeffding trees were:

- gracePeriod: 2,
- splitConfidence: 1e-4,
- tieBreaking: 1e-14,
- driftCheck: 1000,
- windowSize: 100000,

- conceptDriftP: true,
- clsLeafModel: "naiveBayes",
- clsAttrHeuristic: "giniGain",
- maxNodes: 60,
- attrDiscretization: "bst"

The algorithm was of course dominant to the moving average algorithm, but it was not competitive with LR or SVMR. The illustration of a tree can be found in the figure below. More in depth analysis would make sense in the case where HT manages to be one of the top methods for predicting a certain phenomenon. The HT algorithm in QMiner is able to export the tree structure in a standard graph DOT format, which can be visualized with many tools online and offline.

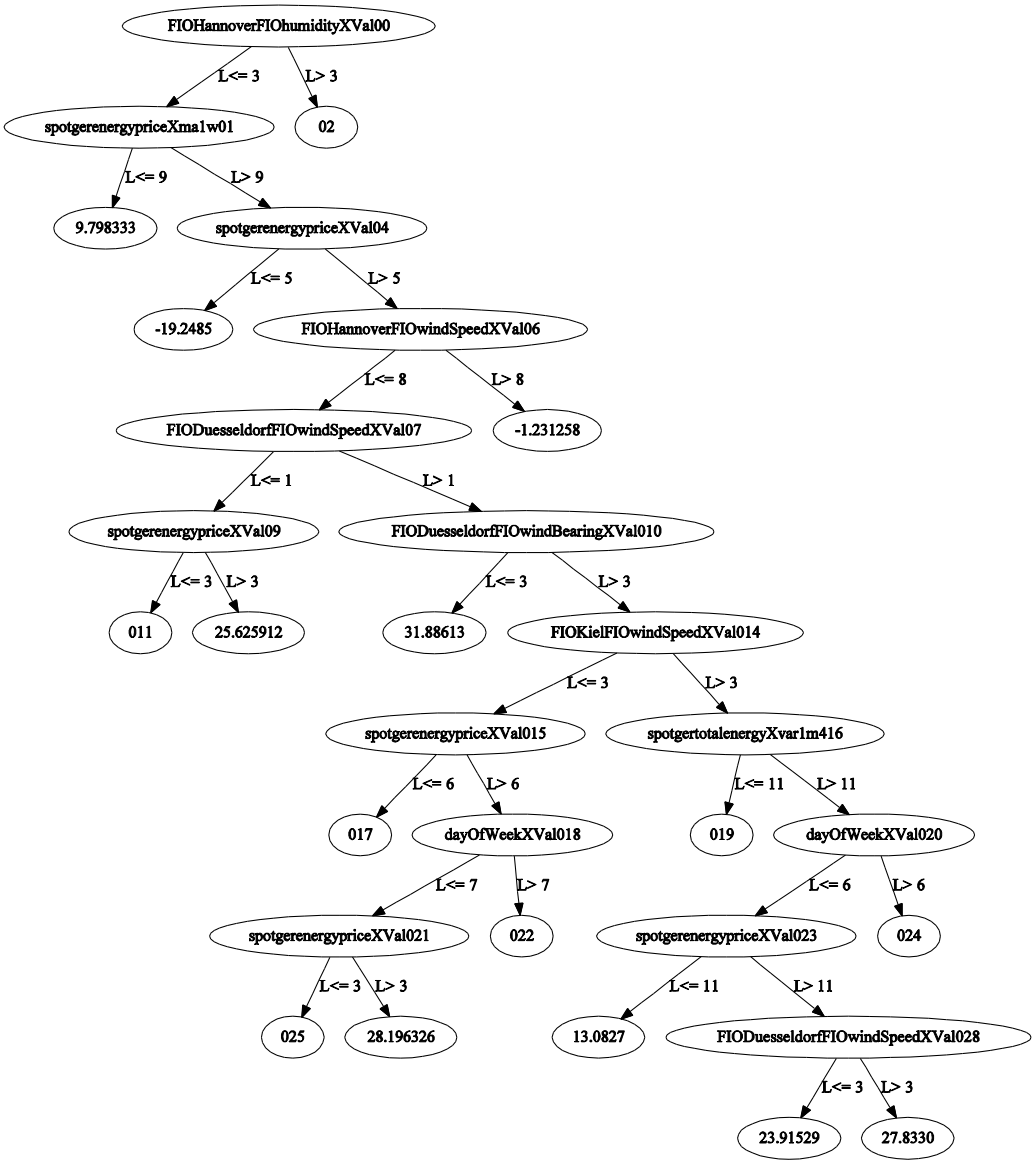


Figure 17: The Hoeffding Tree for HT-ARSFP in the default parameters scenario.

5.1.4 Neural Networks Notes

Default parameters are:

- learnRate: 0.2
- momentum: 0.5

Neural networks have been set with the linear transfer function output layers. The inside layers as well as the input layer have the usual tangens hyperbolicus transfer function set. This means that normalization of the feature vectors is required and the normalization of the out values is not.

NN has proven itself to be quite an unstable method with a vast parameter space to explore. We had little luck finding any useful model using the NN method in the EPEX scenario.

5.1.5 SVM Regression Notes

Default parameters for SVMR are:

- C: 0.02,
- eps: 0.05,
- maxTime: 2,
- maxIterations: 1E6

The parameter C is a measure of fitting (if it is too small, it could cause under-fitting and if it is too big over-fitting). The parameter eps defines the difference between the prediction and the true value that is still not considered an error. So, we can understand this parameter as a measure of noise in the data. A nice description of the SVM parameters can be found in the footnotes[10].

5.2 CSI

Valid fused data interval: 3.3 years (from June 2011 until October 2014)

Learning period: 2 years

Evaluation period: 1.3 years

Total number of features: 48

Number of models: 24

Feature to predict: building consumption without cooling (turin-building-CSI_BUILDING-buildingconsumptionnocooling)

In the CSI use case SVMR has been the dominant method. The neural networks and HT also produced comparable results. There was an interesting finding with testing the SVMR. Normally one would normalize features between MIN/MAX, but if we normalized the target value with a factor, smaller than its MAX we received better results for the model. A phenomena worth some additional exploration.

The sample prediction can be seen in Figure 18 and comparison of the models in Table 17.

[10] http://www.svms.org/parameters/

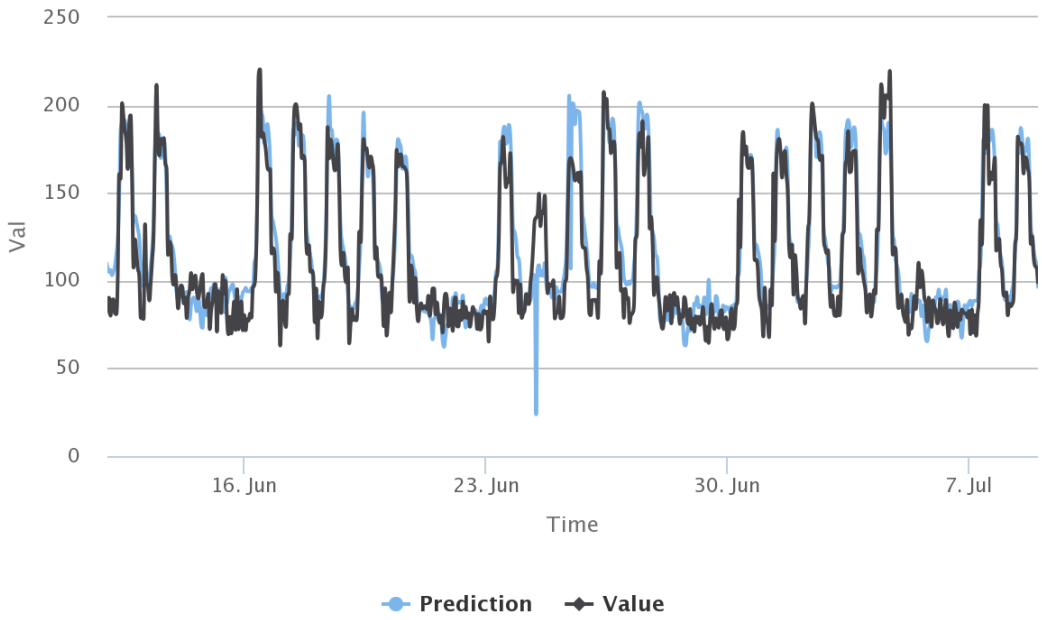


Figure 18: An example of prediction for CSI use-case.

Model	ME	MAE	MSE	RMSE	R^2
SVMR-ARFP(eps=0.015;norm=250)	-2,74	11,71	272,1	16,50	0,84
SVMR-ARFP(eps=0.005;norm=250)	-2,78	11,72	273,6	16,54	0,84
SVMR-ARFP(eps=0.05;norm=175)	-2,69	11,83	275,4	16,59	0,84
SVMR-ARFP(eps=0.05;norm=150)	-2,59	11,77	275,4	16,60	0,84
SVMR-ARFP(eps=0.03;norm=150)	-2,72	11,74	275,5	16,60	0,84
SVMR-ARFP (eps=0.05;norm=200)	-2,69	11,89	276,1	16,62	0,84
SVMR-ARFP(eps=0.05;norm=100)	-2,51	11,80	280,5	16,75	0,84
SVMR-ARFP(eps=0.05;norm=250)	-2,86	12,09	281,0	16,76	0,84
SVMR-ARP (eps=0.05;norm=200)	-1,96	12,01	285,2	16,89	0,83
SVMR-ARFP (eps=0.05; norm=300)	-3,11	12,38	288,4	16,98	0,83
SVMR-ARP (eps=0.05;norm=300)	-2,51	12,50	296,8	17,23	0,83
LR-ARFP	-3,24	12,45	322,5	17,96	0,81
LR-ARP	-3,46	12,62	331,0	18,19	0,81
SVMR-ALL (eps=0.05;norm=300)	-1,96	13,61	348,7	18,67	0,80
LR-ARSFP	-0,78	13,35	382,0	19,54	0,78
LR-ARSP	-0,81	13,44	389,7	19,74	0,77
NN (6,lr=0.02)	0,32	12,54	395,9	19,90	0,77
HT-ARSFP	-2,69	13,74	400,7	20,02	0,77
NN (4,lr=0.02)	0,18	12,65	407,6	20,19	0,76
NN (5,lr=0.02)	0,24	12,69	409,9	20,25	0,76
NN (7,lr=0.02)	0,40	12,78	414,2	20,35	0,76
HT-ARP	-2,61	13,51	414,7	20,36	0,76
NN (8,lr=0.02)	0,30	12,70	416,7	20,41	0,76
HT-ARFP	-2,40	13,77	424,2	20,60	0,75
NN (6, lr=0.03)	-0,12	13,31	446,0	21,12	0,74
HT-ARP (sc=1e-2, tb=1e-4)	-1,13	13,53	512,7	22,64	0,70
NN (6,lr=0.01)	0,79	15,00	558,2	23,63	0,67
NN (6-3, lr=0.02)	-0,10	16,81	634,2	25,18	0,63
NN (6-4, lr=0.02)	-0,16	18,09	715,8	26,75	0,58

LR-ARF	-0,37	19,57	768,8	27,73	0,55
LR-AR	1,02	19,77	789,7	28,10	0,54
NN (4-3, lr=0.02)	-0,09	19,92	846,6	29,10	0,51
LR-ARS	1,87	20,45	879,4	29,65	0,49
LR-ALL	-0,94	13,99	896,6	29,94	0,72
NN (10-4-3,lr=0.02)	-0,18	20,97	917,8	30,30	0,46
MA (7)	0,01	21,79	954,4	30,89	0,44
MA (30)	-0,05	22,47	999,8	31,62	0,42
MA (365)	-2,02	23,88	1093,6	33,07	0,36
HT-ARF	-3,35	23,78	1121,3	33,49	0,35
HT-AR	-6,71	24,92	1182,4	34,39	0,31

Table 17: Error measures for different models in the CSI use-case.

5.2.1 Linear Regression Notes

The feature relevance illustration can be found in Figure 19. Autoregressive features seem very important here (values and moving averages).

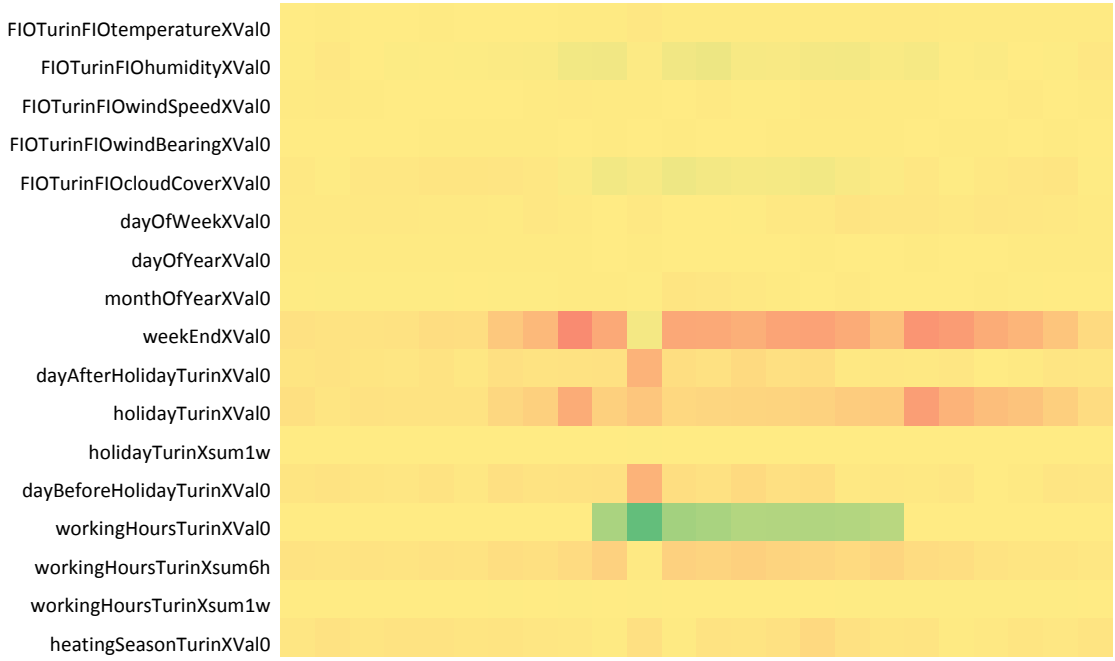


Figure 19: Feature relevance in LR-ALL for CSI use-case.

Many autoregressive aggregates seem to be irrelevant (min, max, variance). Building the total consumption can also be considered an autoregressive parameter.

Comparison of all LR-ALL models is depicted in Figure 20. There is a curious maximum at the 8th model (8:00). This exception is linked to the beginning of the work day and might have many causes. It could be explained with some phenomenon (like starting work day habits change during the year), or there might be a problem with the data.

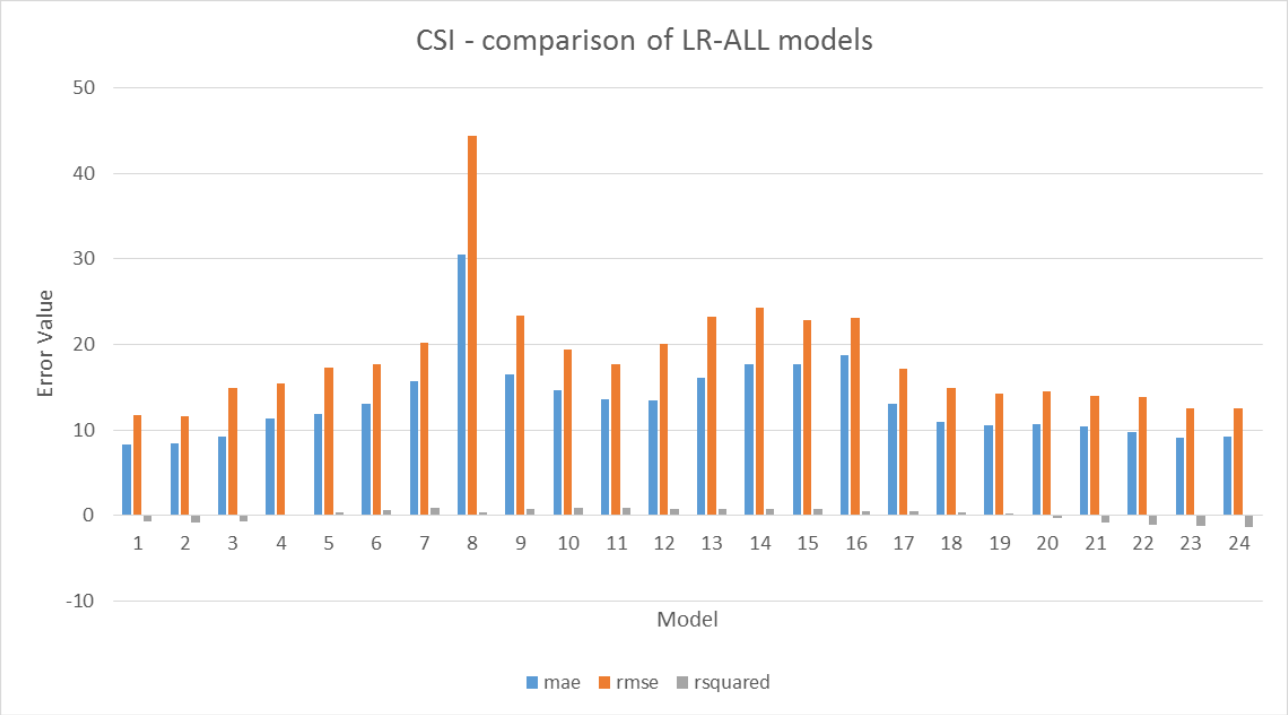


Figure 20: Comparison of models for the CSI use-case LR-ALL.

5.2.2 Hoeffding Tree Notes

Figure below show a nicely shaped Hoeffding tree for one of the models for the CSI use case. The main criteria in the tree being occupancy of the offices, which is determined by working hours, or by the weekend value.

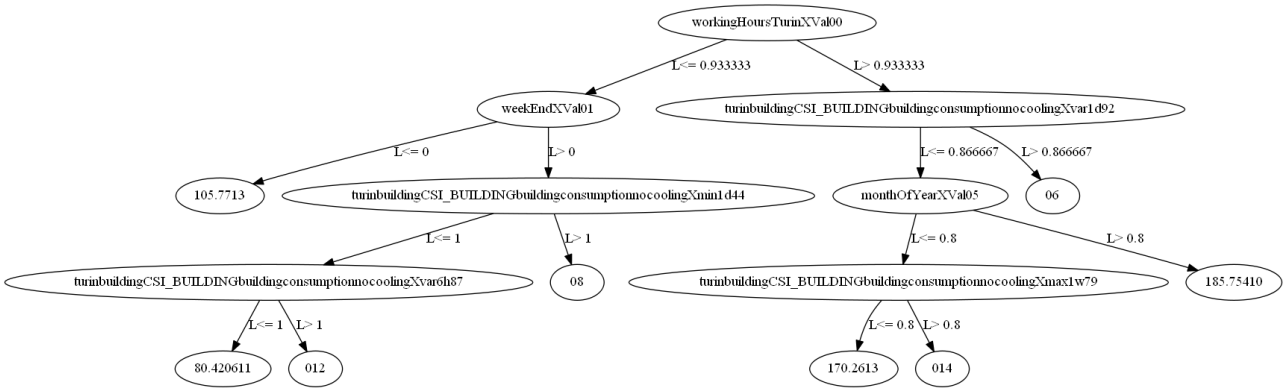


Figure 21: A Hoeffding tree example for the ARP feature set for the 12th model.

5.2.3 SVM Regression Notes

Figure 22 shows very good performance of the SVMR model on data of one week. However, some local peaks are not well modelled and there is also a visible problem with the predictions for Monday.

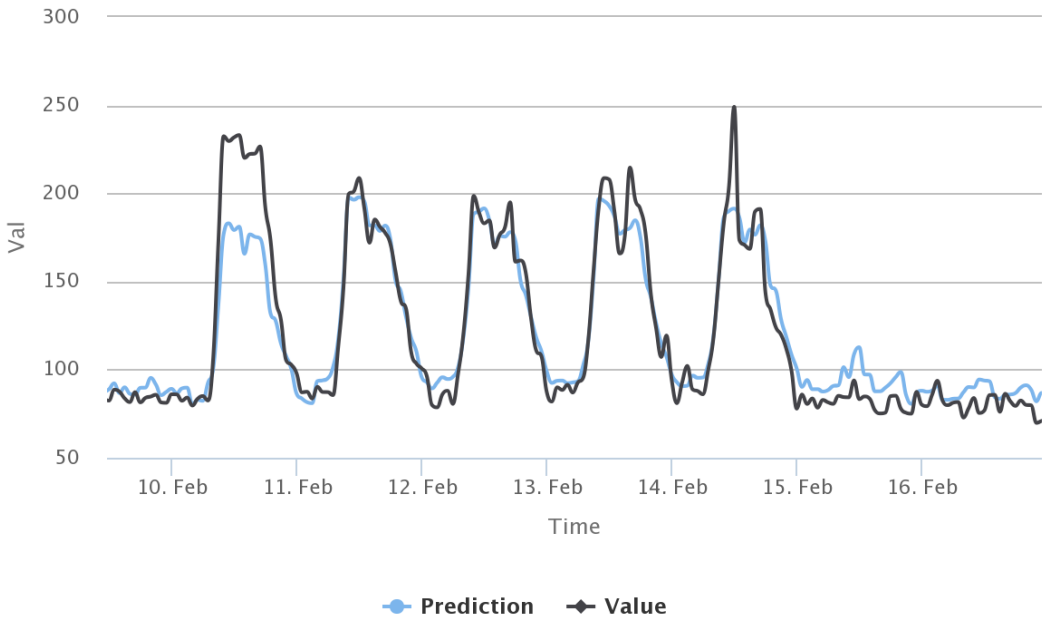


Figure 22: SVMR (norm = 250, e = 0.015) – example of prediction vs. true value.

5.3 IREN

Valid fused data interval: 1.9 years (January 2013 until October 2014)

Learning period: 1.1 year

Evaluation period: 0.5 years

Total number of features: 43

Number of models: 24

Feature to predict: thermal plant production hour-by-hour (nubi-plant-IREN_THERMAL-Thermal_Production)

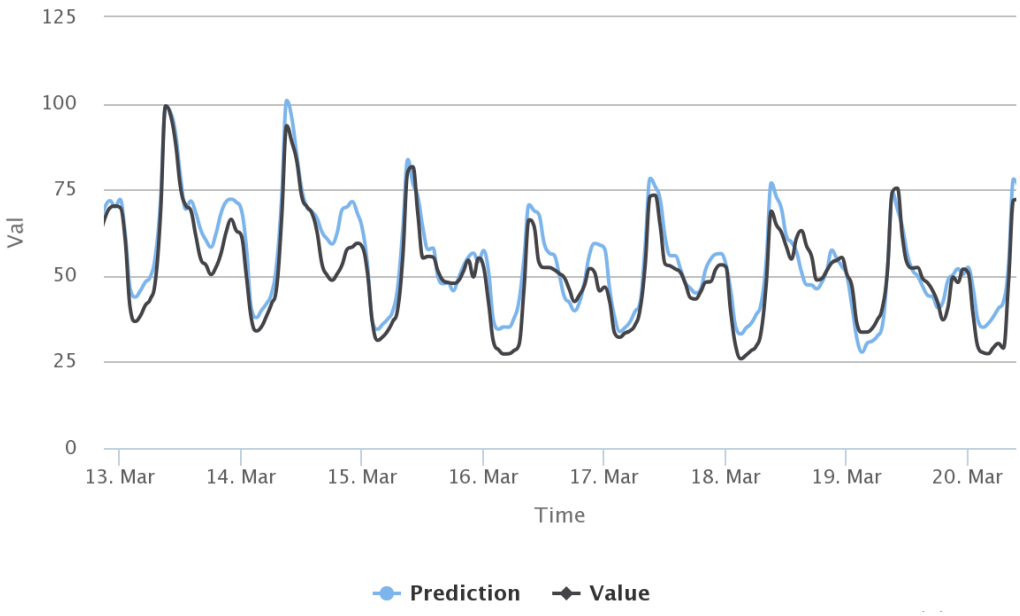


Figure 23: The IREN use-case prediction example.

In this use case we do not use all the available data. The most important for IREN is the heating season data. This is why the last part of the data is not used.

Comparison of models is available in Table 18. Linear regression performs the best again. It is however only slightly better than a naïve moving average method. Hoeffding trees give no usable results in this use case.

Model	ME	MAE	MSE	RMSE	R^2
LR-ALL (non-normalized)	-1,27	11,25	323,8	17,99	0,79
LR-ALL	-0,66	11,11	303,3	17,41	0,80
LR-AR	-0,08	11,49	321,7	17,94	0,79
LR-ARF	-0,46	11,38	318,0	17,83	0,79
LR-ARP	-0,23	11,37	310,9	17,63	0,79
LR-FP	-5,55	18,26	585,1	24,19	0,61
MA (365)	15,86	29,41	1316,7	36,29	0,13
MA (30)	-2,92	16,80	547,2	23,39	0,64
MA (7)	-1,06	12,20	329,3	18,15	0,78
MA (4)	-0,70	11,78	323,8	17,99	0,79
MA (3)	-0,57	11,60	326,5	18,07	0,78
MA (2)	-0,44	11,60	350,8	18,73	0,77
HT-ALL (sc=1e-2,tb=1e-4)	15,56	33,65	1755,6	41,90	-0,16
HT-ALL	13,73	33,11	1625,2	40,31	-0,08
NN (4, lr=0.017)	-0,32	13,35	398,5	19,96	0,74
NN (4, lr=0.01)	-1,03	14,09	413,5	20,33	0,73
NN (4, lr=0.025)	-0,17	13,09	394,0	19,85	0,74
NN (3, lr=0.025)	-0,32	13,12	391,0	19,77	0,74
NN (5, lr=0.025)	0,02	13,07	400,1	20,00	0,73
NN (6, lr=0.025)	0,00	13,23	416,9	20,42	0,72
NN (7, lr=0.025)	0,00	13,20	401,7	20,04	0,73
NN (4-3, lr=0.025)	-0,13	13,03	384,8	19,62	0,74

NN (5-3, lr=0.025)	-0,09	13,04	393,8	19,84	0,74
NN (4-6-3, lr=0.025)	-0,09	13,04	393,8	19,84	0,74
NN (4-6-3, lr=0.04)	-0,10	12,35	347,7	18,65	0,77
NN (4-6-3, lr=0.05)	-0,43	12,48	360,8	18,99	0,76
SVMR (c=0.03, e=0.02, norm = 200)	0,19	13,07	370,4	19,25	0,75
SVMR (c=0.04, e=0.03, norm = 200)	0,08	13,09	370,8	19,26	0,75
SVMR (c=0.04, e=0.02, norm = 200)	0,15	12,97	370,3	19,24	0,75
SVMR (c=0.04, e=0.01, norm = 200)	0,15	12,97	370,0	19,24	0,75
SVMR (c=0.06, e=0.01, norm = 200)	0,22	12,92	372,3	19,30	0,75

Table 18: IREN use-case comparison of models.

5.3.1 Linear Regression Notes

Next two figures illustrate our experiments with the linear regression. Comparison of hourly models gives an already well known picture. Interesting is the table of relevance of certain features. Certain features remain relevant in all the models, but they often change their sign. If it is humid near noon the models will predict lower thermal production, but in the late afternoon/evening they will predict that the production will be higher.

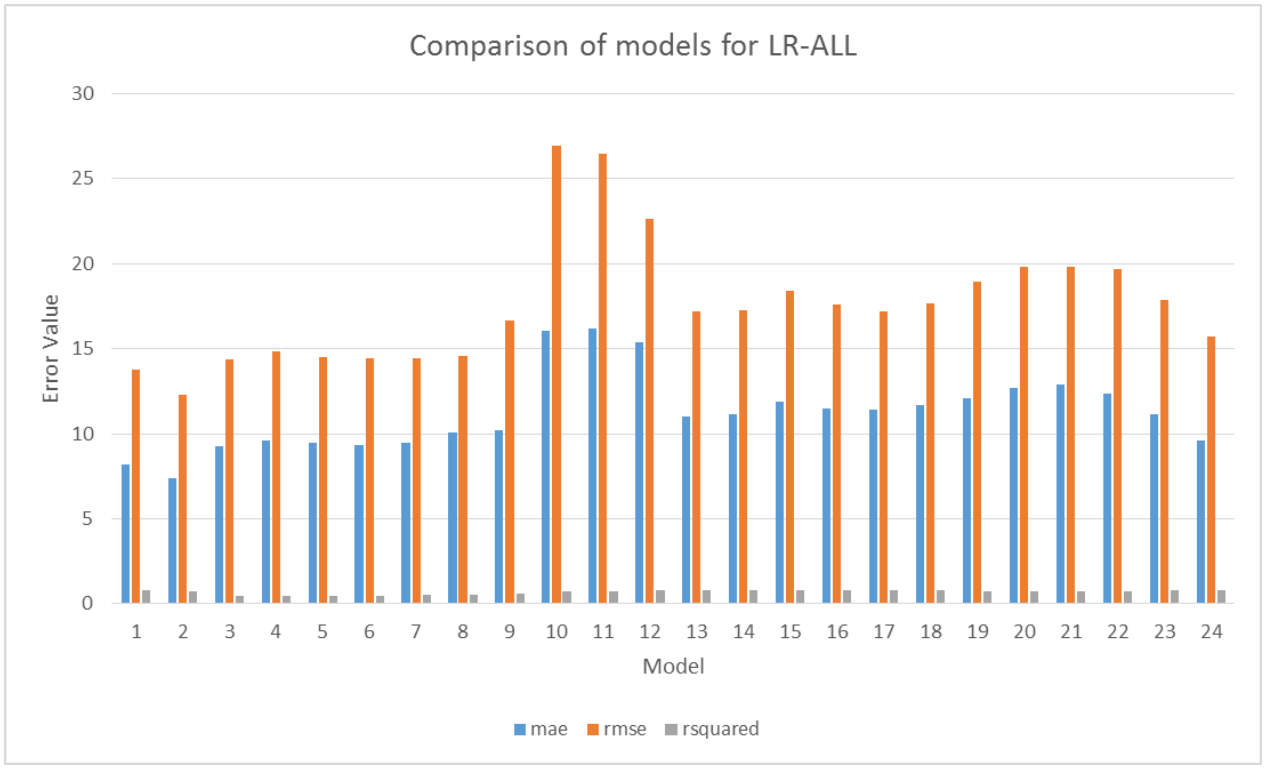
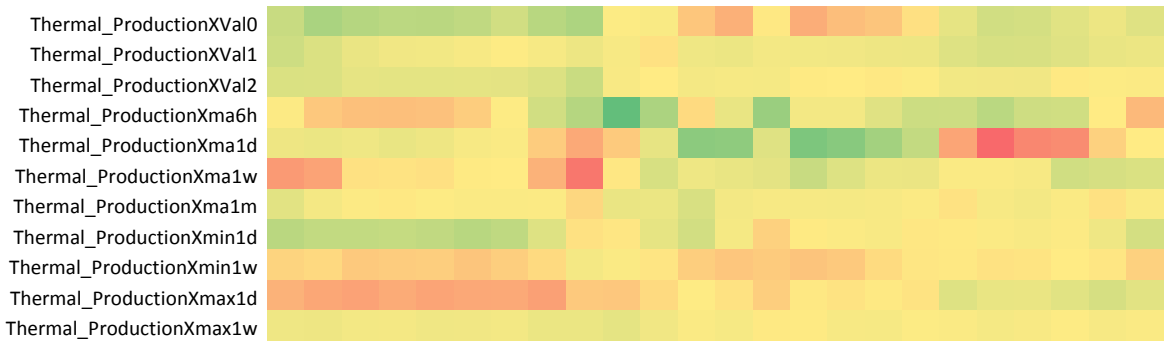


Figure 24: Comparison of LR-ALL models.



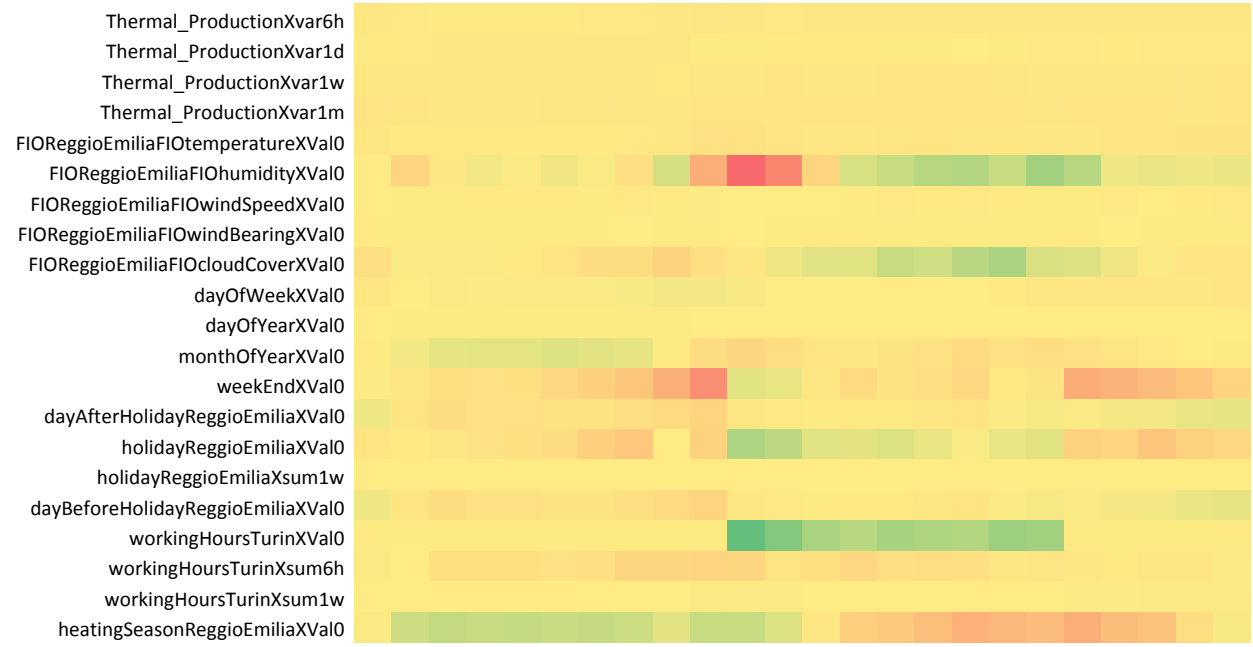


Figure 25: Relevance of different features in the IREN use case for LR-ALL.

Note a different scale is used for Thermal_Production features and other features. Autoregressive features have lower significance.

5.4 NTUA

Valid fused data interval: 5 years (from January 2010 until October 2014)

Learning period: 3 years

Evaluation period: 1.8 years

Total number of features:

Number of models: 24

Feature to predict: average power demand for Lampadario building (ntua-building-LAMPADARIO-last_average_demand_a)

At the first glance predictions in the NTUA scenario have big problems. For some periods they are quite good (see Figure 26), but for other (more extreme cases) not so much (see Figure 27). There seem to be many exceptions (days off, strike, etc.), which are not handled well in the additional properties data. Further in-depth data analysis is needed regarding those issues.

In general the model scores are quite good. MAE for LR-ALL is 4.24, which is in the range of other models. However many periods are missing from the data (consumption for those periods is calculated as 0). These intervals represent holidays, when the data was not recorded. A relatively good fit probably fixes the score.

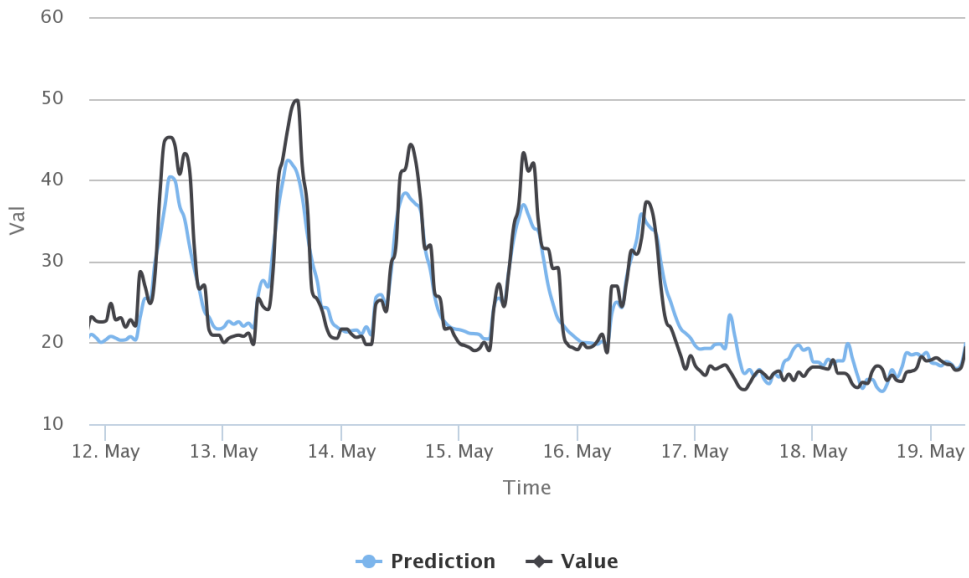


Figure 26: Good predictions in the NTUA use-case (LR-ALL).

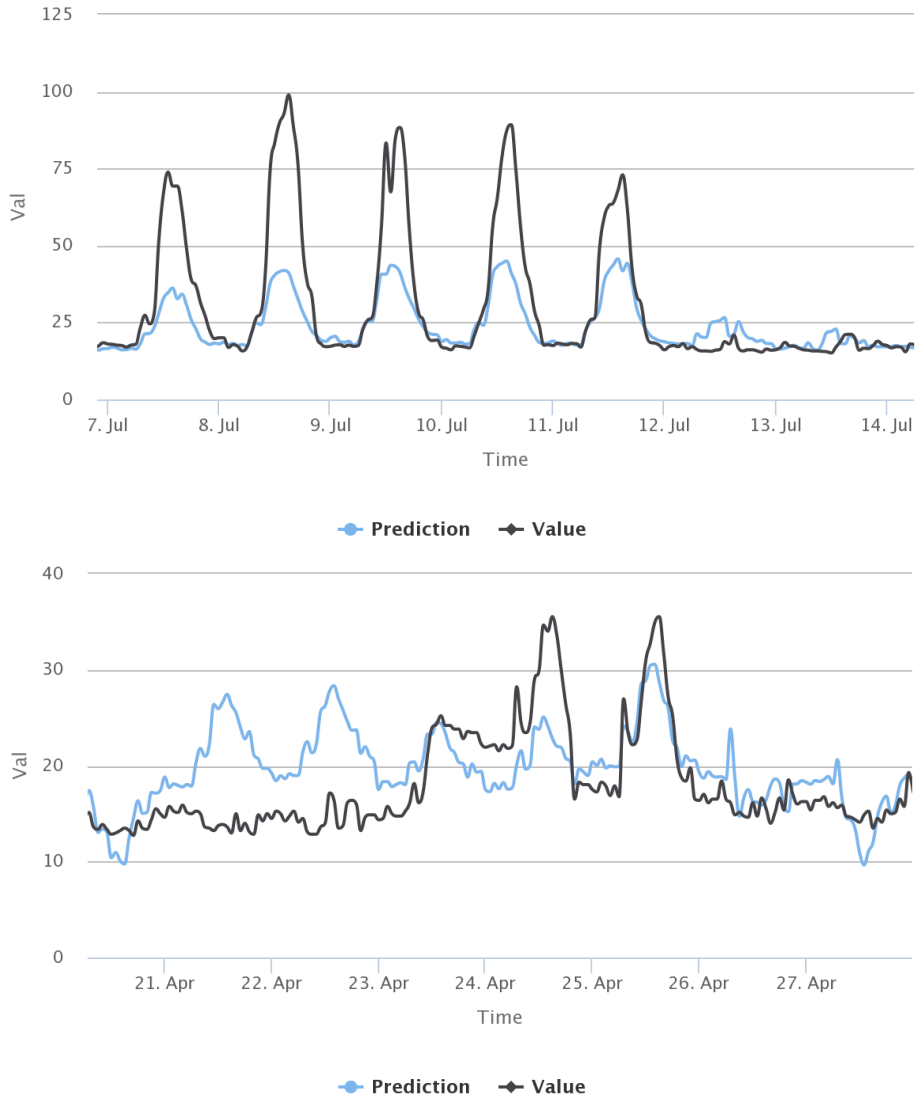


Figure 27: Bad prediction of peaks (above) and bad additional properties data (below) in the NTUA scenario (LR-ALL).

Some experiments have been made with the SVMR and the NN, but no better fit was found for these two basic problems. Comparing the methods in such a setting does not make much sense. More time should instead be invested into feature engineering.

6 Optimal Flow for Data Mining Methods

Modelling in the streaming scenario with different types of data is not a trivial issue. There are many details that need to be taken into account in order to provide a working streaming prototype. Firstly, we need to identify different kinds of data coming into the system.

- **Sensor Data**
  This is streaming data in the "classical" sense of the word. The system receives data in an orderly fashion. There are a few exceptions, though. Data is not coming as it is being generated. Often systems implement some sort of buffering (to avoid overhead, network congestions, and similar) or there are just some technical issue preventing data to be received in a true on-line fashion. We need our system to deal with such exceptions
- **Prediction (weather) Data**
  Prediction data is different in the way that predictions can change through time. For example: weather forecast for a day after tomorrow will be refined tomorrow and different values will have to be taken into account. Many streaming mechanisms do not work in such a scenario. The data we have is also not aligned with the measurement, but usually extends to and beyond the prediction horizon.
- **Properties Data**
  Properties data is the data concerning the time of day, week, the day of year, holidays, working days, weekends, moon phase, etc. This is the data that can be pre-calculated and is usually pushed into the prediction engine at once (in the initial data push).

Each type of the data requires different handling!

To handle such diversity we broke the data mining component into two types: the Data instance and the Modelling Instance. In the NRG4Cast Year 2 scenario we are using one Data Instance and multiple Modelling Instances as depicted in Figure 28.

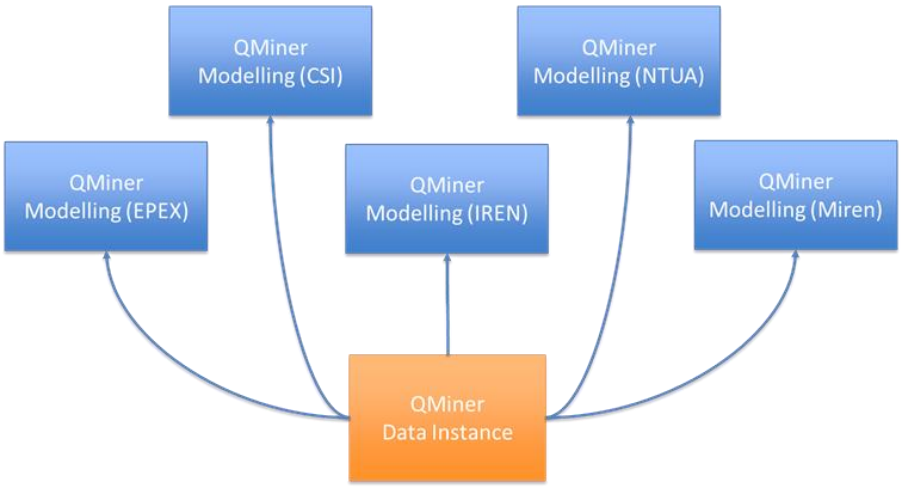


Figure 28: Data and Modelling instances of QMiner in the NRG4Cast Y2 scenario.

Data Instance includes the following components:

Push (time sync) Component
This component overcomes the problems, caused by unsynchronized arrival of sensor, prediction, and properties data. This component is invoked for a group of data streams arriving to the Data Instance. The component determines the lowest possible timestamp, where data exists in the Data Instance. Then it pushes items from the entire stream in a timely fashion, that is one by one, where all the items follow the correct timeline. This makes it possible for the Modelling Instance to

implement normal streaming algorithms on top of the data stream. The pushed data includes the measurement data and aggregates.

The Modelling Instance includes the following components:

- **The Store Generator**
  The Modelling instance needs to provide stores for all the data it will be receiving, as well as for all the merged data streams. This includes merged stores by the group of sensors and a meta-merged store with all the data.
- **The Load manager**
  The Load manager component is the one that invokes the Push Component. It provides the push component with the list of relevant data streams and the timestamp of the last received measurement. The load manager is loading the following data separately: sensors, properties, and forecasts.
- **The Receiver**
  The Receiver listens to the data sent by the Push Component. Its sole purpose is to write the data in the appropriate stores. It also needs to take additional care that no record is overwritten.
- **Merger**
  The Merger Component is a universal component that takes a group of data streams (these groups consist of one type of the following stream types: sensor data, properties, and predictions), with arbitrary timestamps and joins all the measurements in a single store (table). The Merger only works with data items that do not break the timeline. The result of the merger is a huge table with data for each single timestamp in the source data.
- **The Re-sampler**
  The Merger data needs to be resampled to the relevant interval. In NRG4Cast this interval is mostly 1 hour. All the other measures are irrelevant. Different interpolation methods can be used to provide the relevant record (previous, linear). The records are written in a corresponding data storage.
- **The Meta-merger**
  As the dynamics of the different groups of data (sensor data, predictions, and properties) are different, the data is received at different times. The Meta-merger provides a full data record composed from all three types merged and resampled stores.
- **The Semi-automated modeller**
  The Modeller is described in more detail below.

Figure 29 shows data flow for modelling in the streaming data scenario. As described above, there are two instances of the QMiner present in such a scenario. A so-called **Data Instance** (which calculates the aggregates) and the **Modelling Instance** (which is in charge of more complex functionalities).

The data enters the analytical platform at the data instance and gets written in the Measurement store. The stream aggregators are attached to the measurement store and they calculate the predefined aggregates. When they are calculated, they are written to the aggregate store.

Further use of all the data in the Data Instance is managed by the Push component.

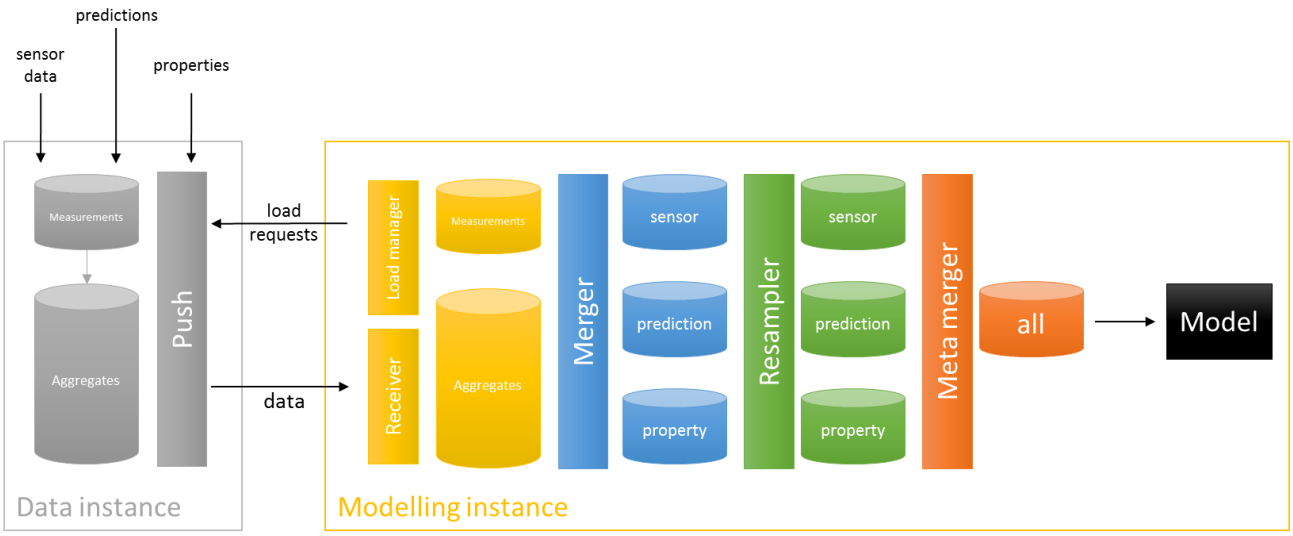


Figure 29: Data flow for modelling in the streaming data scenario.

7 Prototype Description

Code repository: https://github.com/klemenkenda/nrg4mine

Branches:

- Master (the Data Instance)
- Modelling (the Modelling Instance)

7.1 Aggregate Configuration

In QMiner we speak of two kinds of stream aggregates that are relevant for handling the streaming data. These are tick aggregates (based only on the last received value) and buffer aggregates (based on a bunch of measurement in the last interval). The goal of the prototype was among other things to define these aggregates with a simple configuration structure. With `tickTimes` we define all the relevant timestamps for the tick aggregates, with `tickAggregates` we do the same for the buffer aggregates. Once we have the relevant timestamps, we attach aggregates to them with `tickAggregates` and `bufAggregates`.

The tick aggregates are relatively cheap, as they only require one step. The buffer aggregates are much more problematic, as they work on the whole interval. It is sometimes difficult to compute buffer aggregates for longer time periods.

```
// config tick aggregates
tickTimes = [
    { name: "1h", interval: 1 },
    { name: "6h", interval: 6 },
    { name: "1d", interval: 24 },
    { name: "1w", interval: 7 * 24 },
    { name: "1m", interval: 30 * 24 },
    { name: "1y", interval: 365 * 24 }
];

tickAggregates = [
    { name: "ema", type: "ema" }
];

// config winbuff aggregates
bufTimes = [
    { name: "1h", interval: 1 },
    { name: "6h", interval: 6 },
    { name: "1d", interval: 24 },
    { name: "1w", interval: 7 * 24 },
    { name: "1m", interval: 30 * 24 },
    { name: "1y", interval: 365 * 24 }
]

bufAggregates = [
    { name: "count", type: "winBufCount" },
    { name: "sum", type: "winBufSum" },
    { name: "min", type: "winBufMin" },
    { name: "max", type: "winBufMax" },
    { name: "var", type: "variance" },
    { name: "ma", type: "ma" }
]
```


7.2 Model Configuration

Our goal was to introduce a general schema, which would take care of the time-series modelling inside the QMiner. There are many sub-steps in creating a model, or even in running a model and our goal was to put the configuration of the model in one place and then take care of all the other functionality (loading the data, merging it, creating the feature space, creating the feature vectors, preparing the models, learning, and predicting), based on this configuration.

Some of the flexibility with feature generation has been lost with such an approach temporarily, but all the improvements in the future should be easy and, more importantly, available not only in one, but in all the modelling scenarios.

We have two types of models: those who care about loading the data (`master: true`) and those who just use shared data stores (`master: false`). Each model is labelled with an `id` and a `name`. The data source is specified in the `storename`, which is actually a prefix to the set of stores connected to the model (stores for the merged sensor data, merged prediction data and the merged property data, as well as the store for the meta-merged data, which can store the full feature vector used for the model).

The properties `dataminerurl` and `callbackurl` represent the links to the Data Miner REST interface for pushing and modelling instance of the REST interface, respectively.

The Re-sampler can trigger a function every time a new record is received. With this mechanism we can implement prediction triggering in an on-line fashion. Scheduling is defined in the `type` property. The main part of the configuration structure is the definition of the data sources (due to historical reasons) named `sensors`. This is a set of data sources, representing sensors, predictions, and properties (called features in this configuration). In this configuration each sensor is represented by its `name`, a set of relative timestamps `ts` (in the units of resample interval `resampleint`), the relevant aggregates `aggrs` (names are based on the ids from the aggregate configuration), and the type of the data stream `type` (“sensor”, “prediction”, and “feature”). Note that predictions do not have corresponding aggregates, as they are not considered a classical stream and the aggregating mechanisms can only deal with incremental additions and not insertions at an arbitrary time.

The phenomena and the prediction horizon are defined in `prediction`, used method in `method`, method specific parameters in `params`, and the interval used for resampling in `resampleint`.

```
// definition of the model

modelConf = {
    id: 1,
    name: "EPEX00h",
    master: true,
    storename: "EPEX",
    dataminerurl: "http://localhost:9789/enstream/push-sync-stores",
    callbackurl: "http://localhost:9788/modelling/",
    timestamp: "Time",
    type : {
        scheduled: "daily",
        startHour: 11
    },
    sensors: [

        /* sensor features */
        { name: "spot-ger-energy-price", ts: [0, -24, -48], aggrs: ["ma1w", "ma1m", "min1w",
              "max1w", "var1m"], type: "sensor" },
```

```
        { name: "spot-ger-total-energy", ts: [0, -24, -48], aggrs: ["ma1w", "ma1m", "min1w",
              "max1w", "var1m"], type: "sensor" },
        { name: "WU-Duesseldorf-WU-cloudcover", ts: [0], aggrs: ["ma1w", "var1w"], type:
              "sensor" },

       ...

       /* weather forecast */
        { name: "FIO-Berlin-FIO-temperature", ts: [24], type: "prediction" },
        { name: "FIO-Berlin-FIO-humidity", ts: [24], type: "prediction" },
        { name: "FIO-Berlin-FIO-windSpeed", ts: [24], type: "prediction" },
        { name: "FIO-Berlin-FIO-windBearing", ts: [24], type: "prediction" },
        { name: "FIO-Berlin-FIO-cloudCover", ts: [24], type: "prediction" },

       ...

       /* properties */
        { name: "dayBeforeHolidayAachen", ts: [24], aggrs: [], type: "feature" },
        { name: "holidayAachen", ts: [24], aggrs: [], type: "feature" },
        { name: "dayOfWeek", ts: [24], aggrs: [], type: "feature" },
        { name: "dayOfYear", ts: [24], aggrs: [], type: "feature" },

       ...
    ],
    prediction: { name: "spot-ger-energy-price", ts: 24 },
    method: "linreg", // linreg, svmr, ridgereg, nn, ht, movavr
    params: {
        /* model relevant parameters */
    },
    resampleint: 1 * 60 * 60 * 1000
};
```

7.3 Classes

7.3.1 TSmodel

The TSmodel is the main class of the NRG4Cast modelling solution, as it represents an abstract view on the models. It includes functions to support all the modelling tasks and takes a configuration structure of the model as an input. The properties and methods of the class are described below.

```
/* PROPERTIES / CONFIGURATIONS */
this.conf              // model config

this.lastSensorTs      // last timestamp of pulled sensor data
this.lastFeatureTs     // last timestamp of pulled features
this.lastPredictionTs  // last timestamp of pulled weather predictions

this.mergerConf;       // merger conf
this.resampledConf;    // resampled store configuration
this.pMergerConf;      // merger conf for weather predictions
this.fMergerConf;      // merger conf for features

this.ftrDef;           // feature space definition
```

```
this.htFtrDef;          // Hoeffding tree feature space definition

this.mergedStore;       // merged store
this.resampledStore;    // resampled store
this.pMergedStore;      // weather predictions merged store
this.fMergedStore;      // additional features merged store

this.ftrSpace;          // feature space

this.rec;               // current record we are working on
this.vec;               // feature vector, constructed from record


/* MODELLING FUNCTIONS */

// METHOD: predict()
// Make the prediction
this.predict = function (offset);

// METHOD: createFtrVec()
this.createFtrVec = function ();

// METHOD: initModel()
// Init model from configuration
this.initModel = function ();

// METHOD: initFtrSpace()
// Init feature space
this.initFtrSpace = function ();

// METHOD: findNextOffset(offset)
// Finds next suitable offset from the current offset up
this.findNextOffset = function (offset);


/* CONFIG & LOAD FUNCTIONS */

// METHOD: getMergerConf - sensors
// Calculates, stores and returns merger stream aggregate configuration for the model
// configuration
this.getMergerConf = function ();

// METHOD: getFMergerConf - features
// Calculates, stores and returns features merger stream aggregate configuration for the model
// configuration
this.getFMergerConf = function ();

// METHOD: getPMergerConf - weather predictions
// Calculates, stores and returns weather prediction merger stream aggregate configuration
// for the model configuration
this.getPMergerConf = function ();
```

```
// METHOD: getMergedStoreDef
// Returns merged store definition, based on mergerConf (sensor, feature, prediction)
this.getMergedStoreDef = function (pre, mergerConf);

// METHOD: getResampledAggrDef
// Returns resampled store definition
this.getResampledAggrDef = function ();

// METHOD: makeStores
// Makes appropriate stores for the merger, if they do not exist.
this.makeStores = function ();

// METHOD: getFields
// Get array of fields in the merger.
this.getFields = function ();

// METHOD: getFtrSpaceDef
// Calculate ftrSpaceDefinition from model configuration.
this.getFtrSpaceDef = function ();

// METHOD: getHtFtrSpaceDef
// Calculate htFtrSpaceDefinition from model configuration - for Hoeffding trees regression.
this.getHtFtrSpaceDef = function ();

// METHOD: getLearnValue
// Get Learn Value for specified offset.
this.getLearnValue = function (store, offset);

// METHOD: getOffset
// Get offset for a specified timestamp
this.getOffset = function (time0, store);

// METHOD: getRecord
// Get record for specified offset (meta-merger)
this.getRecord = function (offset);

// METHOD: loadData
// Loads data from Data Instance (separated by groups – sensor data, predictions, properties)
this.loadData = function (maxitems);

// METHOD: updateTimestamps
// Updates last timestamps from the last records in the stores
this.updateTimestamps = function ();

// METHOD: initialize
// Initialize sensor stores (if needed), initialize merged and resampled store if needed.
this.initialize = function ();

// METHOD: updateStoreHandlers
// Updates handles to the 4 stores (3x merged + 1x resampled). Useful if we restart the
// instance.
this.updateStoreHandlers = function ();
```

7.3.2 pushData

The pushData class takes care of pushing relevant data in the timeline. This function is implemented in the Data Instance and is invoked by the Modelling Instance.

```
// CLASS: pushData
// Pushes all the data from relevant inStores from a particular data/timestamp up.
pushData = function (inStores, startDate, remoteURL, lastTs, maxitems);

// Find and returns first datetime field from store
getDateTimeFieldName = function (store);

// Find and return all datetime fields in store
getDateTimeFieldNames = function (stores);

// Returns index with lowest timestamp value from currRecIdxs array
findLowestRecIdx = function (currRecIdxs);

// prepare time-windowed RSet from the store
prepareRSet = function (store, startDateStr, lastTs);

// prepare time-windowed RSets from the stores
prepareRSets = function (stores, startDate, lastTs);
```

7.4 Visualizations

7.4.1 Sensor Data Availability

This visualization shows us, which data is available at any moment. When hovering over data, the exact date interval is shown.

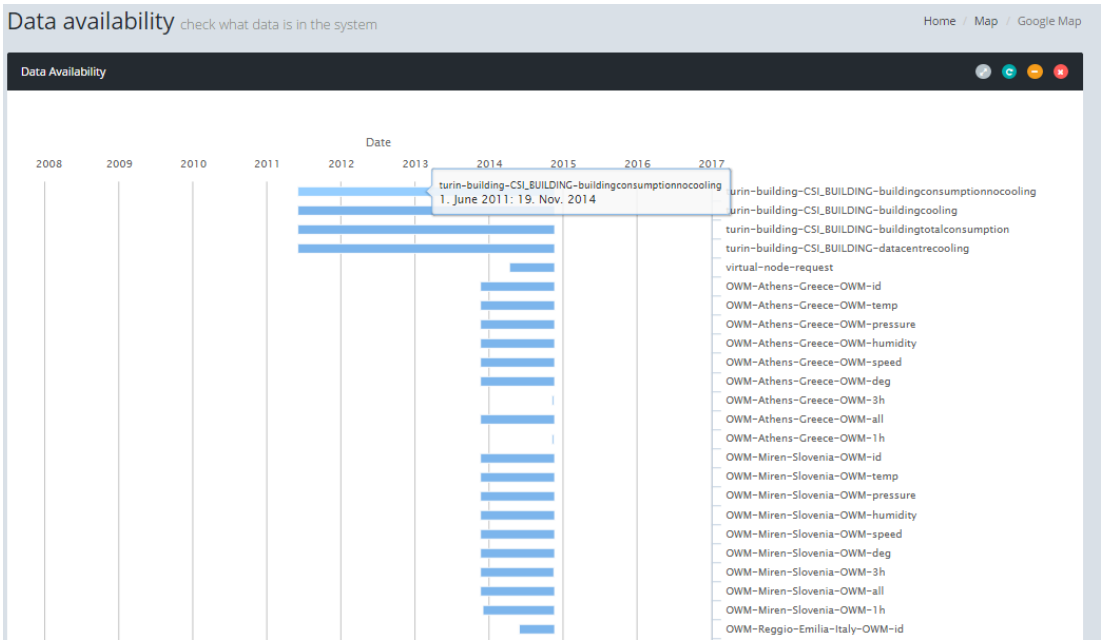


Figure 30: Some of the data available while writing this.

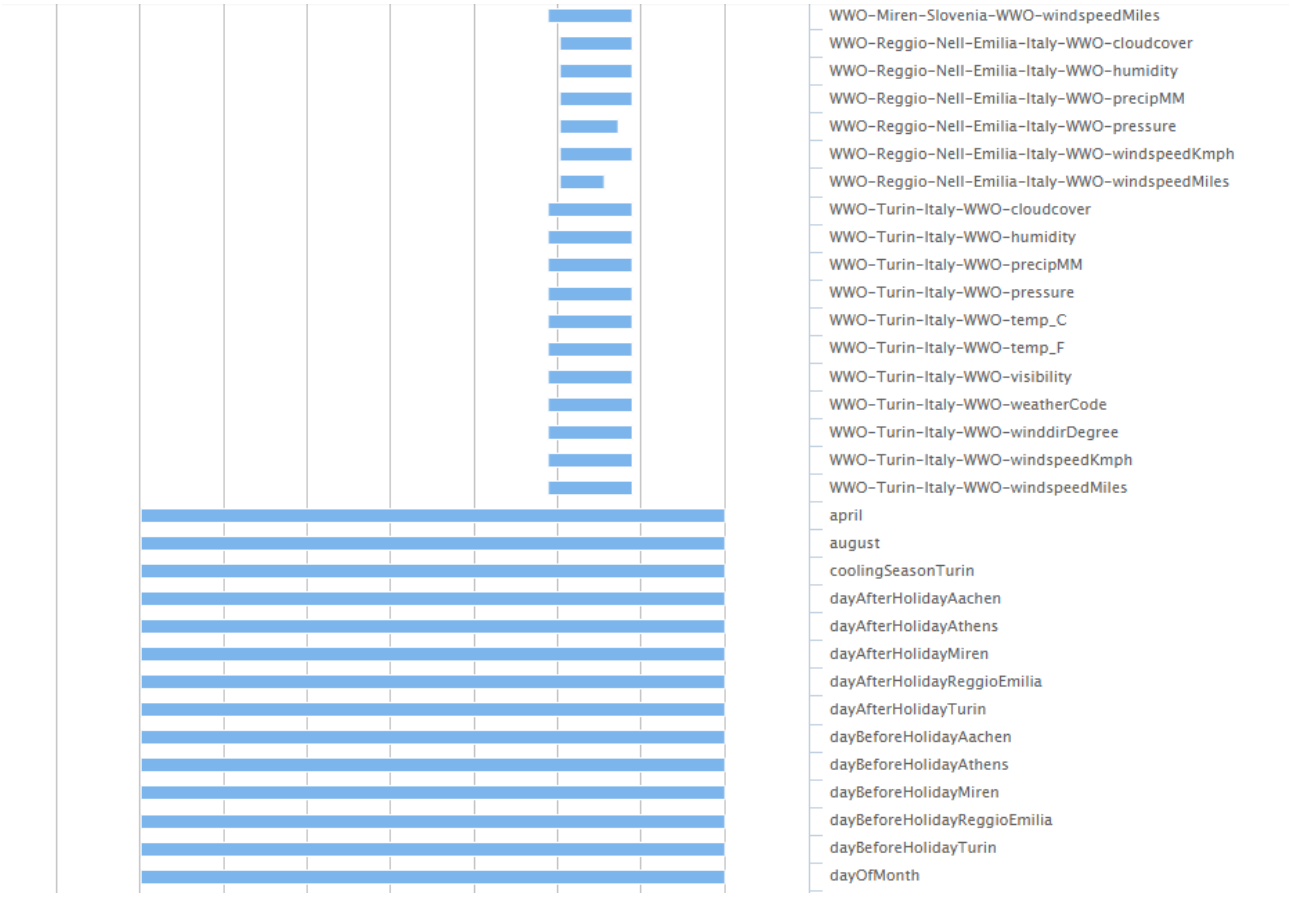


Figure 31: Some sensors have a lot of data and some very little.

Data availability is the Achilles heel of many EU projects related with data mining. There are many steps between the source data (at the pilot) and the end-user. In the NRG4Cast scenario there is the transition mechanism from the pilot to the OGSA-DAI, there is a person, who takes care about the imports (possible human error), then there is the transfer mechanism between OGSA-DAI platform and QMiner Data Instance. QMiner and the servers, where it resides, have had quite some stability issues in the past and reloads were needed; sometimes wrong data was loaded, sometimes the streaming (timeline demand) has prevented some historical data to load.

7.4.2 Custom Visualizations

The custom visualisations application enables visualising highly customisable data from any sensor available. It uses the Highcharts[11] library for drawing graphs and therefore it is possible to zoom into the graph and to export/print the picture.

The graph options include:

- Selecting the sensor
- Setting the start and end date of data samples
- Setting the sampling interval
- Setting the aggregate type

[11] http://www.highcharts.com/

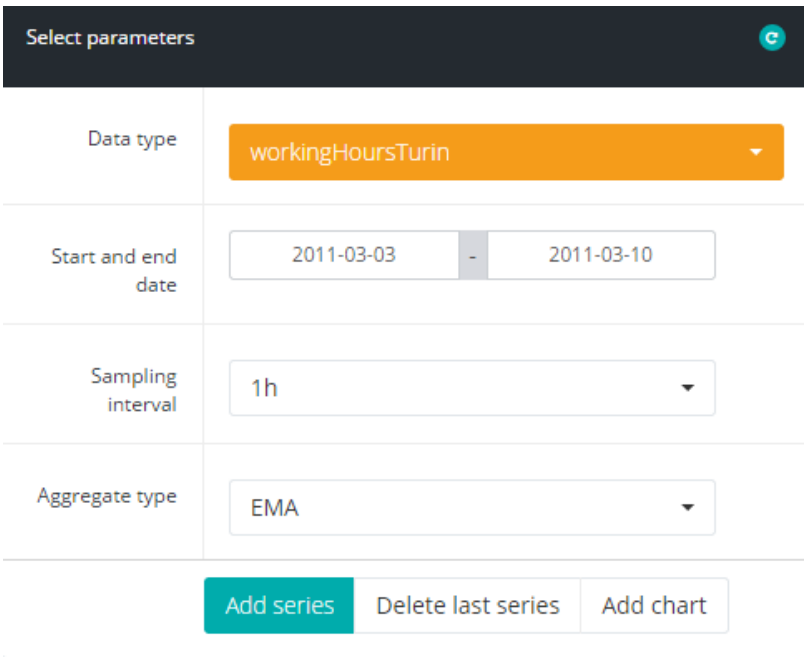


Figure 32: Selecting sensors and all available parameters.

Possible sampling intervals (determines the interval on which the aggregates are computed):

- **1 hour**
- **6 hours**
- **1 day**
- **1 week**
- **1 month**
- **1 year**
- **Raw**

If the sampling interval is set to **Raw** no aggregates are computed and the aggregate option is disabled. To prevent data to become too large, we impose limitations on date interval, according to the chosen sampling interval.

The aggregate options:

- **EMA** (exponential moving average)
- **MA** (moving average)
- **MIN** (moving minimum)
- **MAX** (moving maximum)
- **CNT** (moving count – of measurements inside the moving windows)
- **SUM** (moving window sum)
- **VAR** (moving variance)

The buttons speak for themselves.

It is possible to draw multiple series on a single chart. The application will automatically obtain all information needed for drawing and for each unit of measurement a new y axis will be created on the chart (each series will show which axis it belongs to). If values of series with a same unit are too different, a new y-axis will also be created for better visibility. Series can also be deleted from the chart (FILO), along with any redundant axis.

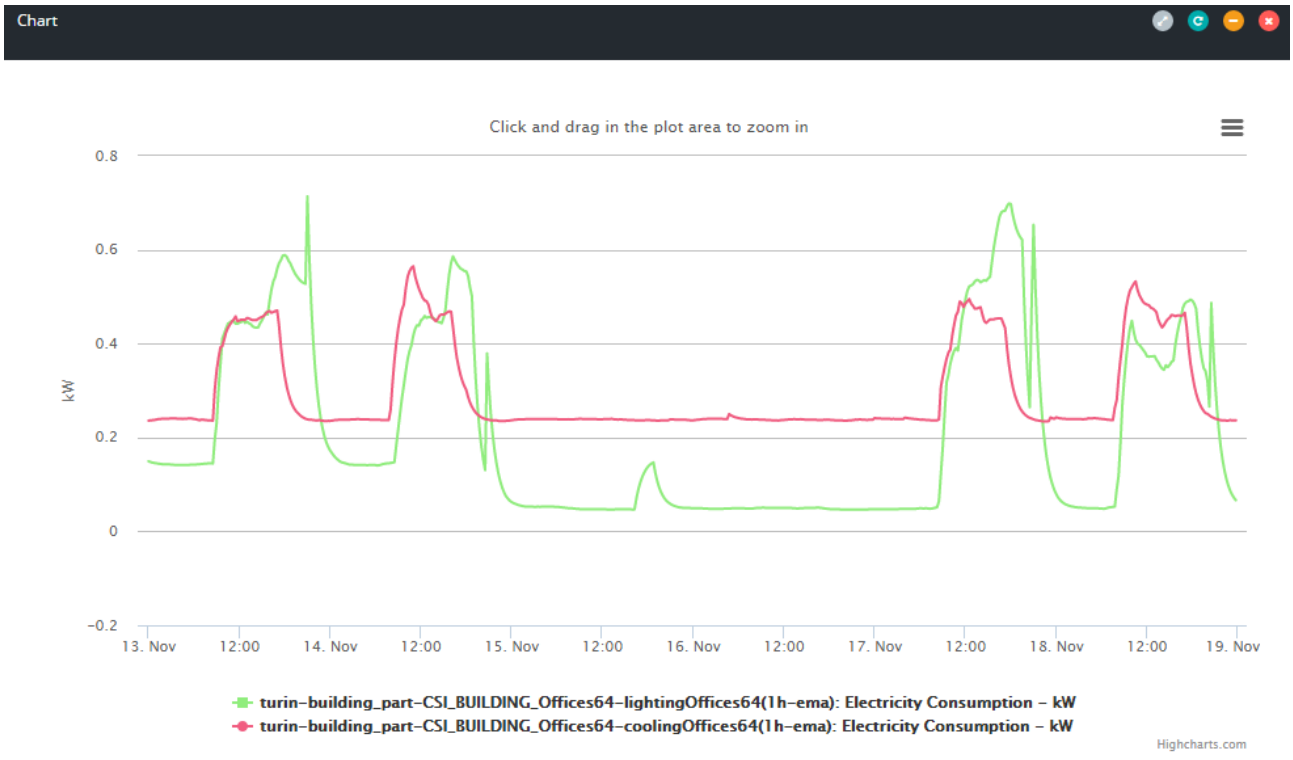


Figure 33: Two series that lay on the same y-axis.

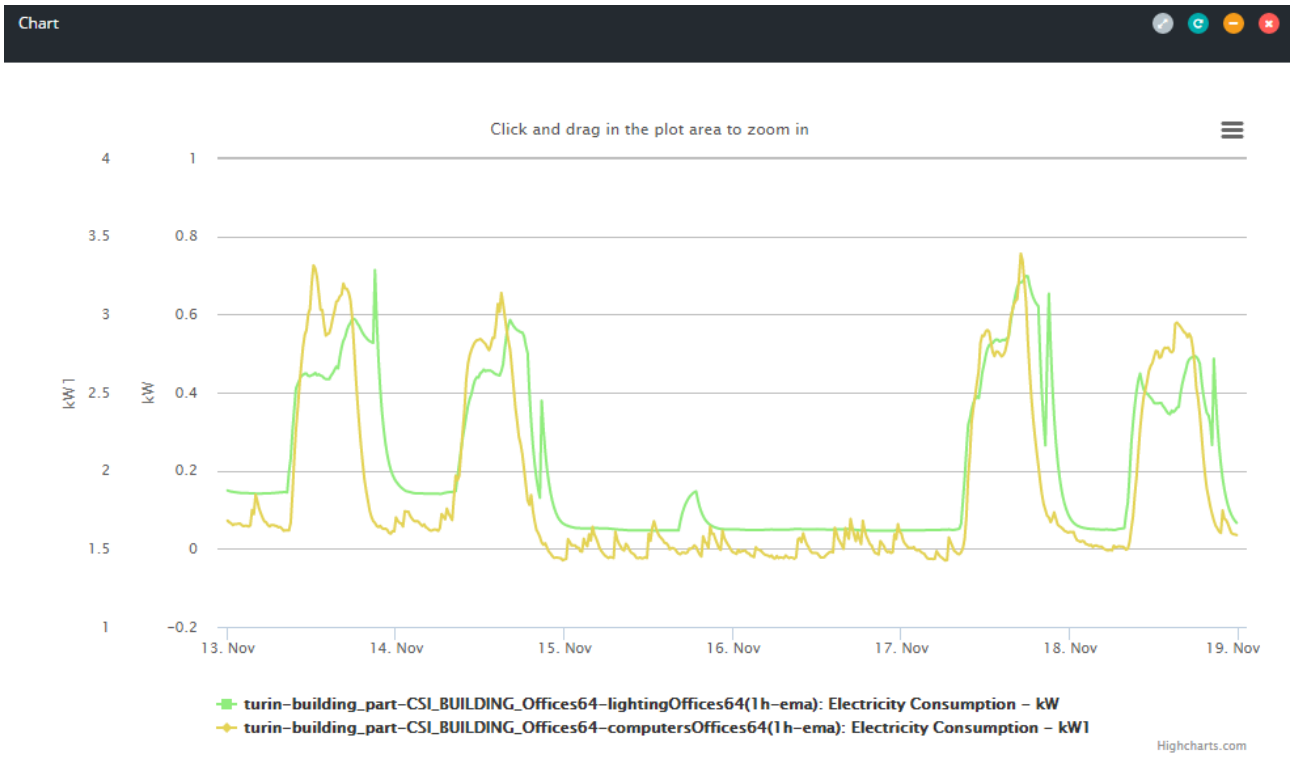


Figure 34: When the difference is too big a new axis is created.

When the chart is empty, the date interval is automatically pre-set to **dateOfLastData – 7 days: dateOfLastData.** If, however, the chart is not empty we would probably want to compare the series and if the selected series has any data in this date interval, we allow this. Otherwise we alert the user the sensors don't have comparable data and ask him to manually adjust the date interval.

It is possible to look at data availability as described in section Sensor Data Availability.

It is also possible to have multiple charts (up to five) open at the same time, but after a new chart is created, it is no longer possible to add series to the previous charts. Regardless it is a nice feature, as we can look at different visualisations simultaneously. A chart can be deleted with the red (x) button, which reduces maximum chart number by one.

Figure 35: Two charts open at the same time.

7.4.3 Exploratory Analysis

This application was created with analysing data correlation in mind. The user can choose up to four sensors, the date interval, sampling interval, and the aggregate type. The application than draws an **n x n** (where **n** is the number of sensors) graph matrix, with all combinations of sensors representing the x and the y axis. The data points can be coloured in a customisable way (in code, file: qminer.explore.js, line: 384-, some options were preprogramed).

The date is handled as in the previous section, assuming the already selected sensors are already "drawn" on the chart.

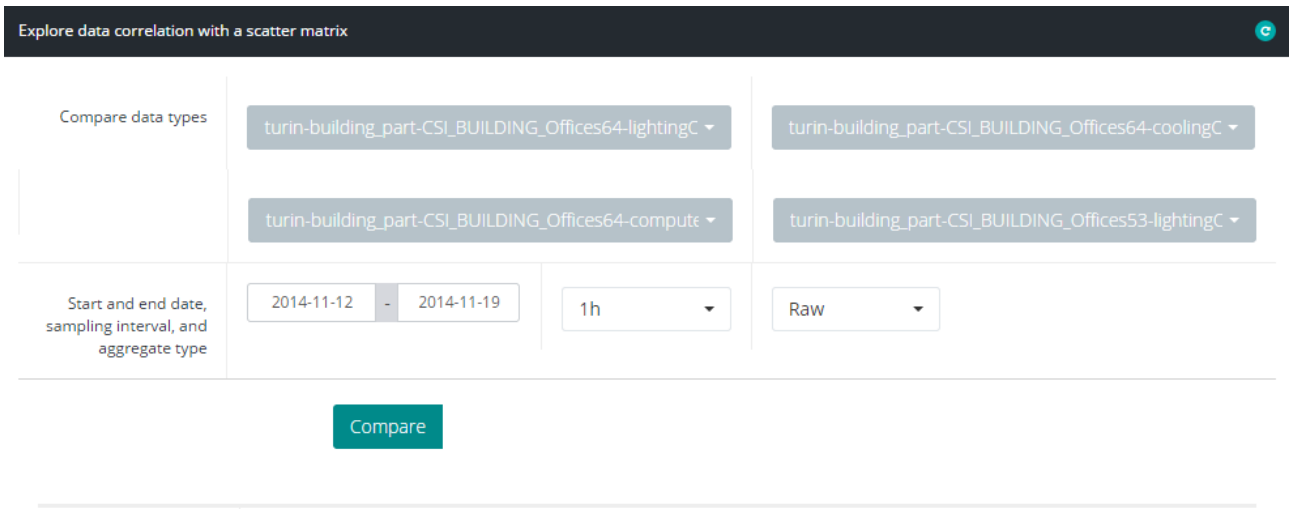


Figure 36: Possible options.

The sampling interval and the aggregate type options are the same as in the previous section. The only difference is **Raw** now resides in the aggregate type list. Why? Because we (possibly) need to draw a lot of graphs we limit the number of points. Firstly we only take one point from each sampling interval (e.g. only one point each day). If the number of points still exceeds our limitation, we randomly sample points up to the limit size and therefore it makes no sense to include **Raw** data, without a specified sampling interval.

After the graphs are drawn we can exclude any of the sensors from the matrix (and therefore reducing the matrix from **n x n** to **n – 1 x n - 1**).

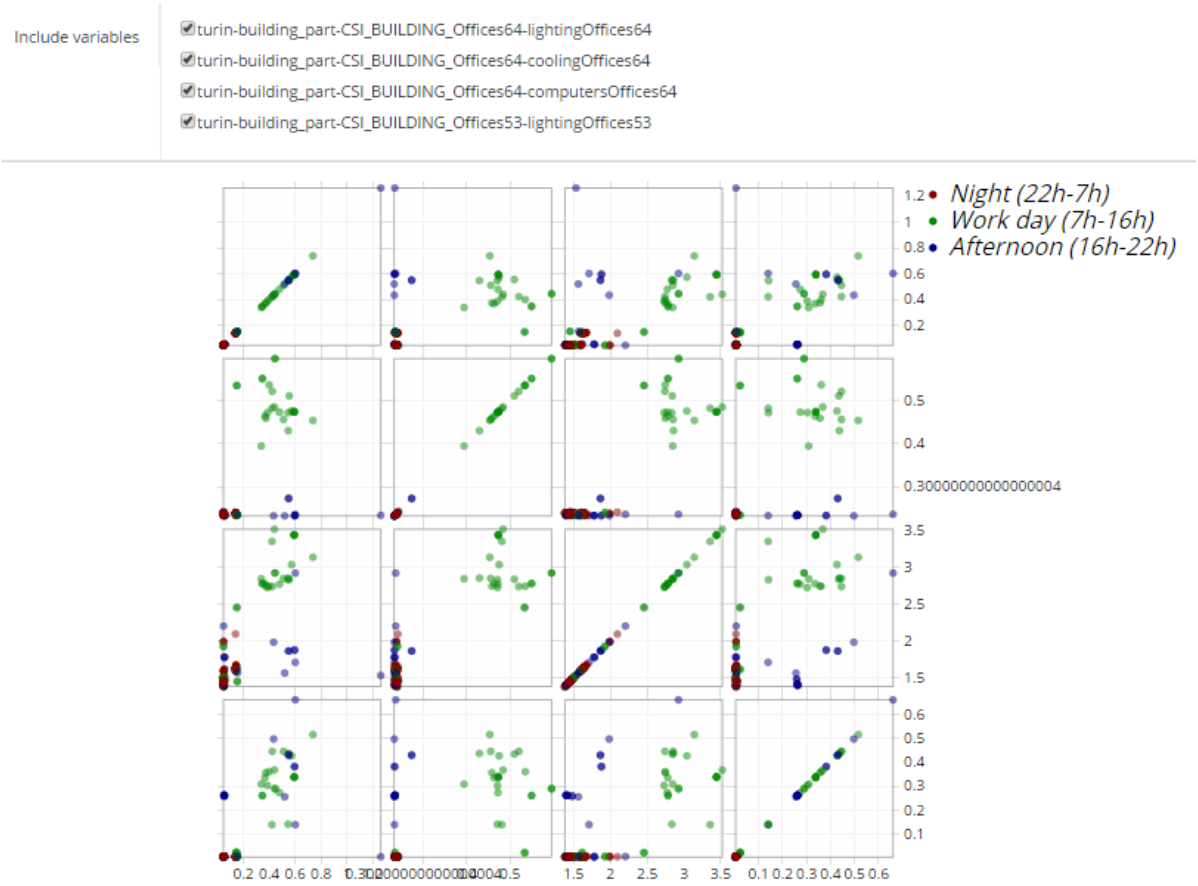


Figure 37: A drawn 4x4 scatter matrix, the data points are coloured by hours in the day.

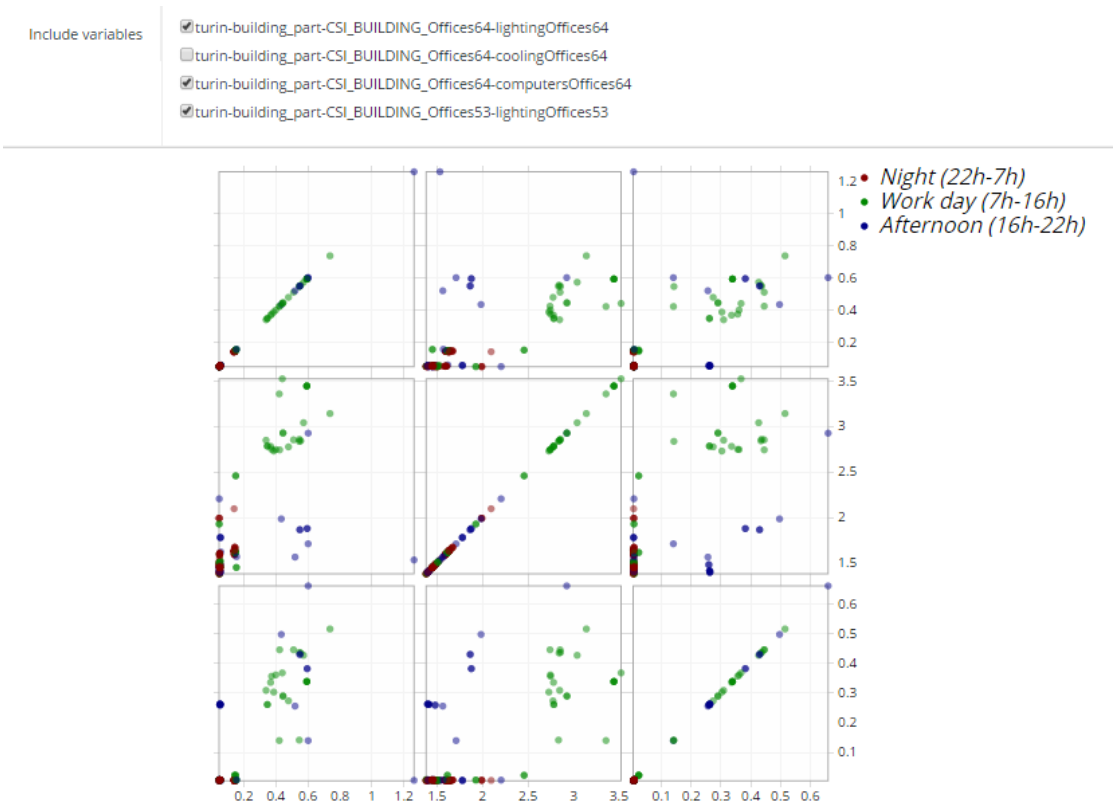


Figure 38: Exclusion (temporary) of one of the sensors.

It is possible to select points on the chart (any) and only those points will be highlighted on all the charts. To reset this, just click on any part of the chart.

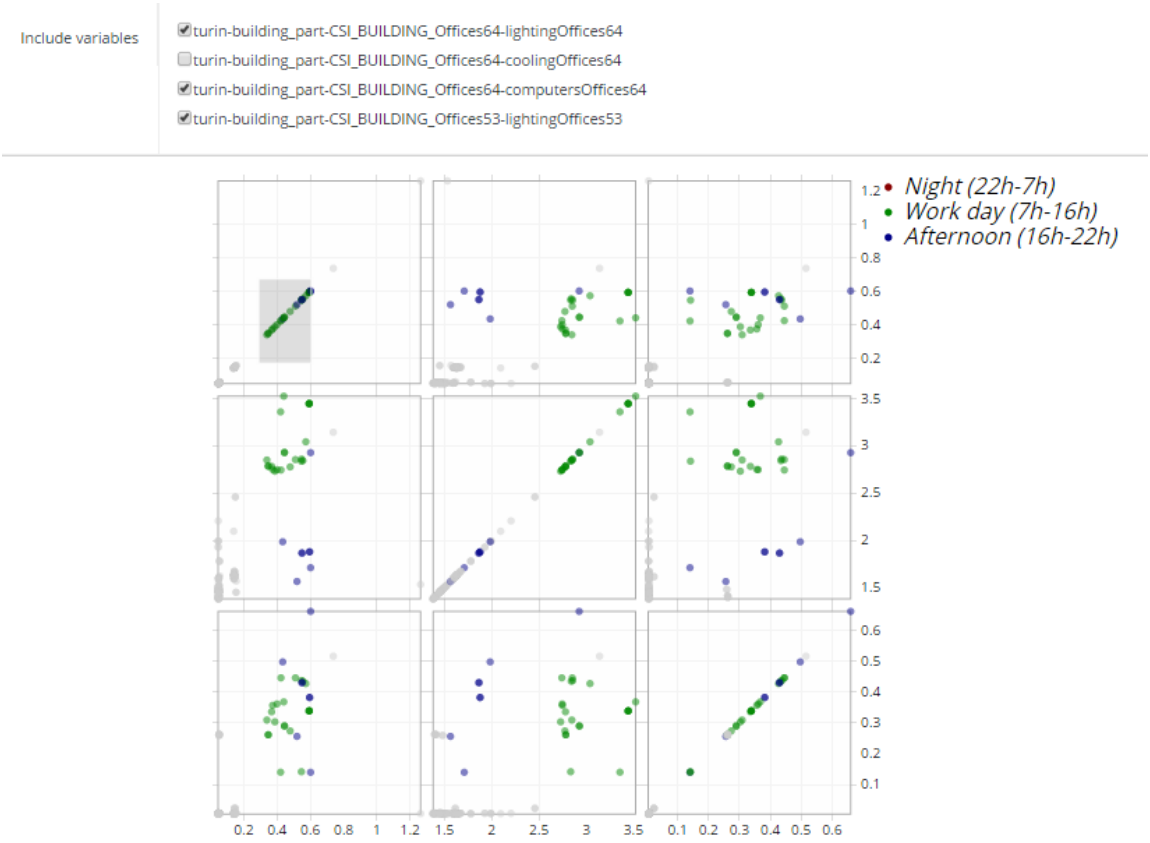


Figure 39: Selecting a few points.

8 Conclusions and Future Work

Prototype from this deliverable represents a working platform for the heterogeneous multivariate data streaming setting. It is able to perform modelling in an off-line and on-line scenario. With minor additions in the 3rd year of NRG4Cast the developed platform should cover all the needs for modelling in the project.

Quite some efforts have been put into feature generation and handling different kinds of data sources throughout the vertical of the NRG4Cast platform. An important discovery, extracted from the experience with the implementation, was that there are significant differences in handling different types of data in the stream modelling setting: streaming sensor data, streaming forecast data, and static additional features data.

The use-cases have been described and the feature vectors defined. The modelling has been tested with 5 different prediction methods. The best method was selected for each of the use-cases and results for 3 pilot scenarios have been produced, as well as an EPEX spot market price prediction algorithm. A first-glance qualitative and quantitative analysis show good results with a relative error between 5 and 10%. These results seem good, but further analysis from the use-cases is needed to put these results into perspective.

The results of this task will be utilized in the task T5.2 (Data-driven prediction methods environment). Additional models need to be prepared and some of the models need to be extended to the additional instances (additional buildings, etc.). One of the objectives of the T5.2 task is also to evaluate and interpret the models presented in this deliverable. A superficial interpretation was already conducted here. Based on the analysis some inconsistencies in the provided data have been discovered and they need to be addressed.

The current prototype infrastructure is able to handle features that are generated directly from the records in the data layer (although the QMiner platform itself is able to also take arbitrary JavaScript functions to generate the features). An extension is needed to enable the user to create features that combine one or more records and thus derive more complex features (linear combinations, products, ratios, transformations, etc.).

References

[1] K. Kenda, J. Škrbec and M. Škrjanc. Usage of the Kalman Filter for Data Cleaning of Sensor Data. In proceedings of IS (Information Sociery) 2013, Ljubljana, September 2013.

[2] K. Kenda, J. Škrbec, NRG4CAST D2.2 – Data Cleaning and Data Fusion – Initial Prototype. NRG4CAST, May 2013.

[3] K. Kenda, J. Škrbec, NRG4CAST D2.3 – Data Cleaning and Data Fusion – Final Prototype. NRG4CAST, November 2013.

[4] R. E. Kalman. A new approach to linear filtering and prediction problem. Journal of basic Engineering, 82(1):35-45, 1960.

[5] Y. Chamodrakas et al., NRG4CAST D2.4 – Data Distribution Prototype. NRG4CAST, November 2013.

[6] T. Hubina et al., NRG4CAST D1.4 – Final Toolkit Architecture Specification. NRG4CAST, February 2014.

[7] http://en.wikipedia.org/wiki/Wind_power_in_Germany (accessed on March 5th, 2014).

[8] G. Corbetta et al. Wind in Power – 2013 European statistics. The European Wind Energy Association. February 2014.

[9] T. Hubina et al., NRG4CAST D1.6 – Final Prototype of Data Gathering Infrastructure, February 2014.

[10] http://en.wikipedia.org/wiki/European_Energy_Exchange (accessed on March 5th, 2014)

[11] http://en.wikipedia.org/wiki/Wind_power (accessed on March 5th, 2014)

[12] http://en.wikipedia.org/wiki/Principal_component_analysis (accessed on June 18th, 2014).

[13] http://en.wikipedia.org/wiki/Naive_Bayes_classifier (accessed on June 18th, 2014).

[14] http://en.wikipedia.org/wiki/Linear_regression (accessed on June 18th, 2014).

[15] http://en.wikipedia.org/wiki/Support_vector_machine (accessed on June 18th, 2014).

[16] http://en.wikipedia.org/wiki/Artificial_neural_network (accessed on June 19th, 2014).

[17] T. Gül, T. Stenzel. Variability of Wind Power and Other Renewables – Management options and strategies, IEA, June 2005.

[18] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" (PDF). Philosophical Magazine 2 (11): 559–572.

[19] Cortes, C.; Vapnik, V. (1995). "Support-vector networks". Machine Learning 20 (3): 273.

[20] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.

[21] Ross J. Quinlan: Learning with Continuous Classes. In: 5th Australian Joint Conference on Artificial Intelligence, Singapore, 343-348, 1992.

[22] Y. Wang, I. H. Witten: Induction of model trees for predicting continuous classes. In: Poster papers of the 9th European Conference on Machine Learning, 1997.

[23] T. Gül, T. Stenzel. Variability of Wind Power and Other Renewables – Management options and strategies, IEA, June 2005.

[24] http://people.cs.uct.ac.za/~ksmith/articles/sliding_window_minimum.html (accessed on July 31st, 2014).

[25] S. Makridakis, S. C. Wheelwright, R. J. Hyndman. Forecasting: Methods and Applications, John Wiley & Sons, Inc. 1998.

[26] R. J. Hyndman, A. B. Koelher. Another look at measurest of forecast accuracy. International Journal of Forecasting, 679-688, 2006.

[27] J. Scoot Armstrong. Principles of Forecasting: A Handbook for Researchers and Practitioners. Kluwer Academic Publishers, Dordrecht, 2001.

[28] Elena Ikonomovska. Algorithms for Learning Regression Trees and Ensembles from Time-Changing Data Streams. PhD thesis. 2012.

[29] Elena Ikonomovska, Joao Gama, and Saso Dzeroski. Learning model trees from evolving data streams. Data Mining and Knowledge Discovery. 2010.

[30] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. Classification and Regression Trees. Chapman and Hall/CRC. 1984

[31] Pedro Domingos and Goeff Hulten. Mining High-Speed Data Streams. KDD. 2000.

[32] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining Time-Changing Data Streams. KDD. 2001.

[33] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On Evaluating Stream Learning Algorithms. Machine Learning. 2013.

[34] Joao Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. Concept Drift Adaption: A Survey. ACM Computing Surveys. 2014.

[35] Wassily Hoeffding. Probability Inequalities for Sums of Independent Bounded Random Variables. American Statistical Association. 1963.

[36] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. Algorithms for Computing the Sample Variance: Analysis and Recommendations. The American Statistician. 1983.

[37] Donald E. Knuth. The Art of Computer Programming, Volume 2: Seminumerical Algorithms. Third edition. Addison-Wesley. 1997.

[38] Bernard Pfahringer, Geoffrey Holmes, and Richard Kirkby. Handling Numeric Attributes in Hoeffdnig Trees. PAKDD. 2008.

[39] Luka Bradesko, Carlos Gutierrez, Paulo Figueiras, and Blaz Kazic. MobiS: Deliverable D3.2. October 2013.

[40] Blaz Fortuna and Jan Rupnik. Qminer. URL http://qminer.ijs.si/

[41] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. Introduction to Algorithms. MIT Press. 2009.

[42] Frank Wilcoxon. Individual comparisons by ranking methods. Biometrics Bulletin. 1945.

[43] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Third edition. Prentice Hall. 2009.

A. Appendix – Ad-hoc QMiner contributions

During the project the QMiner analytical platform has become open source and is available via the GitHub repository[12] . With transition to open-source the platform developed quite a lot in the 2014, which was followed by intensive rewrites of the NRG4Cast software, built on top of QMiner and also NRG4Cast project contributed quite some code to the QMiner.

NRG4Cast contributions to the repository during this time were:

1. Extension of streaming aggregates functionality (serialization of the aggregates, addition of the following aggregates: TWinBufCount, TWinBufSum, TWinBufMin, TWinBufMax, and updates of aggregates TVar and TMa).
2. TFilter.

A.1. Implementation of the sliding window minimum and maximum

The sliding window minimum (maximum) calculation is a bit less trivial task than it looks at the first glance. The problem requires:

- Removing all the obsolete elements (can be more when we don't have a guaranteed fixed interval with incoming measurements) from the array
- Adding the new element into the array
- Calculating the smallest value of the array

A naïve solution would calculate the minimum from scratch with each new measurement, but some fast optimizations are possible in cases, where the incoming measurement is smaller than the previous minimum, or when the outgoing values are larger than the previous minimum. When the outgoing value is the actual minimum, it gets rather complicated, as one would need to go through the list of all the values in the time window.

But this task can be performed in a smarter way [24] using the sorted deque (double-ended queue). If we take care on inserting the values into the deque in a smart way, we can significantly simplify the steps of the algorithm.

For example: if we have a set {1, 6, 4, 8, 8, 3} and our next measurement is 4 than all the measurements that we received before this moment and are greater than 4 will never ever again be candidates for the sliding window minimum. Therefore they can be discarded. Note that if the deque would be sorted by this point, we could remove the last *k* (larger) of the elements from the end of the deque. The new measurement would then be added at the end and the deque would still remain sorted.

The same idea could be used to remove the elements. The elements in deque are not only sorted by the value, but also by the time of arrival (timestamp). The first *n* elements with smaller timestamp than the time-windows limit can be removed from the front of the deque.

[12] https://github.com/qminer/qminer

B. Appendix – The list of Additional Features

A list of all additional features to help models has been compiled. Each feature is mapped to the pilot, where it should be used. A feature is identified by the name. The table also provides information on the source of the data (whether this is a static calculation, a webservice, or similar), the start and end dates, and a textual description of the data. Features represented in grey are not yet imported or implemented.

ID	Name	Source	Start date	End date	CSI	IREN	NTUA	ENV	FIR	EPEX	Description
1	day of the week	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Day of the week in numeric format (0 - Monday, 6 - Sunday)
2	DOW - Monday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Monday (0 - no; 1 - yes)
3	DOW - Tuesday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Tuesday (0 - no; 1 - yes)
4	DOW - Wednesday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Wednesday (0 - no; 1 - yes)
5	DOW - Thursday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Thursday (0 - no; 1 - yes)
6	DOW - Friday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Friday (0 - no; 1 - yes)
7	DOW - Saturday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Saturday (0 - no; 1 - yes)
8	DOW - Sunday	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Sunday (0 - no; 1 - yes)
9	Month	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Month in the numeric format (0 - January; 11 - December)
10	M - January	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	January (0 - no; 1 - yes)
11	M - February	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	February (0 - no; 1 - yes)
12	M - March	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	March (0 - no; 1 - yes)
13	M - April	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	April (0 - no; 1 - yes)
14	M - May	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	May (0 - no; 1 - yes)
15	M - June	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	June (0 - no; 1 - yes)
16	M - July	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	July (0 - no; 1 - yes)
17	M - August	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	August (0 - no; 1 - yes)
18	M - September	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	September (0 - no; 1 - yes)
19	M - October	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	October (0 - no; 1 - yes)
20	M - November	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	November (0 - no; 1 - yes)
21	M - December	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Decemer (0 - no; 1 - yes)
22	Day of the month	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Numeric (1 - 31)
23	Day of the year	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Numeric (1 - 366)
24	Season	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Numeric (1 - Spring, 4 - Winter)
25	Heating season/IREN	calculation/CSI	1.1.2005	1.1.2017		X					Numeric (0 - no; 1 - yes)
26	Heating season/CSI	calculation/CSI	1.1.2005	1.1.2017	X						Numeric (0 - no; 1 - yes)
27	Weekend	calculation/CSI	1.1.2005	1.1.2017	X	X	X	X	X	X	Weekend day (0 - no; 1 - yes)
28	Holiday/it	calculation/CSI	1.1.2005	1.1.2017	X	X					Holiday (0 - no; 1 - yes).
29	Holiday/si	calculation/CSI	1.1.2005	1.1.2017				X			Holiday (0 - no; 1 - yes).
30	Holiday/gr	calculation/CSI	1.1.2005	1.1.2017			X				Holiday (0 - no; 1 - yes).
31	Holiday/de	calculation/CSI	1.1.2005	1.1.2017						X	Holiday (0 - no; 1 - yes).
32	Day before holiday/it	calculation/CSI	1.1.2005	1.1.2017	X	X					Day before holiday (0 - no; 1 - yes)
33	Day before holiday/si	calculation/CSI	1.1.2005	1.1.2017				X			Day before holiday (0 - no; 1 - yes)
34	Day before holiday/gr	calculation/CSI	1.1.2005	1.1.2017			X				Day before holiday (0 - no; 1 - yes)
35	Day before holiday/de	calculation/CSI	1.1.2005	1.1.2017					X	X	Day before holiday (0 - no; 1 - yes)

36	Day after holiday/it	calculation/CSI	1.1.2005	1.1.2017	X	X					Day before holiday (0 - no; 1 - yes)
37	Day after holiday/si	calculation/CSI	1.1.2005	1.1.2017				X			Day before holiday (0 - no; 1 - yes)
38	Day after holiday/gr	calculation/CSI	1.1.2005	1.1.2017			X				Day before holiday (0 - no; 1 - yes)
39	Day after holiday/de	calculation/CSI	1.1.2005	1.1.2017					X	X	Day before holiday (0 - no; 1 - yes)
40	day/night/ENV	calculation/CSI	1.1.2005	1.1.2017				X			Day time (0 - night, 1 day); could be float.
41	day/night/FIR	calculation/CSI	1.1.2005	1.1.2017					X		Day time (0 - night, 1 day); could be float.
42	day/night/Gemany (center)	calculation/CSI	1.1.2005	1.1.2017						X	Day time (0 - night, 1 day); could be float.
43	day/night/NTUA	calculation/CSI	1.1.2005	1.1.2017			X				Day time (0 - night, 1 day); could be float.
44	day/night/CSI	calculation/CSI	1.1.2005	1.1.2017	X						Day time (0 - night, 1 day); could be float.
45	day/night/IREN	calculation/CSI	1.1.2005	1.1.2017		X					Day time (0 - night, 1 day); could be float.
46	moon phases	calculation/CSI	1.1.2005	1.1.2017				X			Moon phase (0 - 360) in degrees.
47	lunch time/CSI	calculation/CSI	1.1.2005	1.1.2017	X						Lunch time (0 - no lunch time, 1 - lunch time).
48	lunch time/NTUA	calculation/CSI	1.1.2005	1.1.2017			X				Lunch time (0 - no lunch time, 1 - lunch time).
49	lunch time/IREN	calculation/CSI	1.1.2005	1.1.2017		X					Lunch time (0 - no lunch time, 1 - lunch time).
50	rush hour/FIR	calculation/CSI	1.1.2005	1.1.2017					X		Rush hour (0 - no rush hour, 1 - rush hour).
51	working hours/CSI	calculation/CSI	1.1.2005	1.1.2017	X						Numeric (0 - no; 1 - yes)
52	working hours/NTUA	calculation/CSI	1.1.2005	1.1.2017			X				Numeric (0 - no; 1 - yes)
53	occupancy/NTUA lab	calculation/CSI	1.1.2005	1.1.2017			X				%
54	occupancy/NTUA solar	calculation/CSI	1.1.2005	1.1.2017			X				%
55	radiation/NTUA solar	webservice	1.1.2005	1.1.2017			X				
56	radiation/IREN	webservice	1.1.2005	1.1.2017		X					
57	solar radiation/CSI	webservice	1.1.2005	1.1.2017	X						
58	solar radiation/FIR	webservice	1.1.2005	1.1.2017					X	X	
59	solar radiation/Germany	webservice	1.1.2005	1.1.2017						X	May be more substations/measurement points in Germany.
60	part-timers schedule/CSI	calculation/CSI	1.1.2005	1.1.2017	X						Working hours of part-time workers (0 - no; 1 - yes).
61	student holidays/NTUA	calculation/CSI	1.1.2005	1.1.2017			X				Holidays (0 - no; 1 - yes)
62	temperature	sensorfeed/JSI	1.1.2005	1.1.2017						X	Temperature in deg. Celsius (different locations).
63	humidity	sensorfeed/JSI	1.1.2005	1.1.2017						X	Humidity in % (different locations).
64	pressure	sensorfeed/JSI	1.1.2005	1.1.2017						X	Pressure in mbar (different locations).
65	cloudcover	sensorfeed/JSI	1.1.2005	1.1.2017						X	Cloudcover in %.
66	visibility	sensorfeed/JSI	1.1.2005	1.1.2017						X	Visibility in km.
67	wind speed	sensorfeed/JSI	1.1.2005	1.1.2017						X	Windspeed in km/h.
68	wind direction	sensorfeed/JSI	1.1.2005	1.1.2017						X	Wind direction in degrees.
69	forecast - temperature	sensorfeed/JSI	1.1.2005	1.1.2017						X	Forecasted temperature in deg. Celsius.
70	forecast - wind speed	sensorfeed/JSI	1.1.2005	1.1.2017						X	Forecasted windspeed in km/h.
71	forecast - wind direction	sensorfeed/JSI	1.1.2005	1.1.2017						X	Forecasted wind direction in degrees.
72	forecast - cloudcover	sensorfeed/JSI	1.1.2005	1.1.2017						X	Forecasted cloudcover in %.
73	forecast - humidity	sensorfeed/JSI	1.1.2005	1.1.2017						X	Forecasted humidity in %.
74	forecast - pressure	sensorfeed/JSI	1.1.2005	1.1.2017						X	Forecasted pressure in mbar.

C. Appendix – The list of Sensors

ID	Pilot	Source	Sensor name (UID)	Phenomena	Availability	Start date	Description	Refresh	Frequency	UoM
	CSI	webservice	turin-building-CSI_BUILDING-datacentrecooling	datacentrecooling	1.11.2013	1.6.2011		1 hour	15 min	kWh
		webservice	turin-building-CSI_BUILDING-buildingtotalconsumption	buldingtotalconsumption	1.11.2013	1.6.2011		1 hour	15 min	kWh
		webservice	turin-building-CSI_BUILDING-buildingcooling	buildingcooling	1.11.2013	1.6.2011		1 hour	15 min	kWh
		webservice	turin-building-CSI_BUILDING-buildingconsumptionnocooling	buildingconsumptionnocooling	1.11.2013	1.6.2011		1 hour	15 min	kWh
		webservice		officeconsumption_N	30.6.2014	15.9.2014	Sensors in typical offices (8) for consumption.	1 hour	15 min	
		webservice		thermalconsumption	1.11.2014	15.9.2014	Thermal energy consumption of building (2).	1 hour	15 min	
	IREN	FTP	nubi-plant-IREN_THERMAL-Thermal_Production	IREN thermal	15.1.2014	15.10.2012	Production for the thermal plant / plants (?).	1 day	1 hour	MWh
		FTP	nubi-substation-*-FLOW_TEMPERATURE	forwardwatertemp	14.5.2014	2.7.2014	From mail from Giulia/Yannis.	1 day	1 hour	C
		FTP	nubi-substation-*-PRIMARY_RETURN_TEMPERATURE	backwardwatertemp_primary	14.5.2014	2.7.2014	For 6 substations in campus Nubi.	1 day	1 hour	C
		FTP	nubi-substation-*-SECONDARY_RETURN_TEMPERATURE	backwardwatertemp_secondary	14.5.2014	2.7.2014		1 day	1 hour	C
		FTP	nubi-substation-*-FLOW	waterflowrate	41773	41822		1 day	1 hour	
		FTP	nubi-substation-*-OUTSIDE_TEMPERATURE	outdoortemp	14.5.2014	2.7.2014		1 day	1 hour	C
		FTP	nubi-substation-*-ROOM_TEMPERATURE	indoortemp	14.5.2014	2.7.2014		1 day	1 hour	C
		FTP	nubi-substation-*-ALARM	alarmcode	14.5.2014	2.7.2014		1 day	1 hour	boolean
	FIR	website/CSV		totaldistance	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	km
		website/CSV		vechilespeed	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	km/h
		website/CSV		stateofcharge	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	%
		website/CSV		stateofcharge_ah	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	Ah
		website/CSV		externaltemperature	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	C
		website/CSV		lon	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	°
		website/CSV		lat	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	°
		website/CSV		height	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	m

	website/CSV		ipack	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	
	website/CSV		upack	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	
	website/CSV		is_driving	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	
	website/CSV		is_charging	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	
	website/CSV		is_parking	15.10.2014	15.10.2014	Not 100%.	1 day	~1 min	
ENV	FTP	miren-lamp-*-SequenceNo	SequenceNo	15.9.2014	15.9.2014	Test site nodes.	15 min	1 min	
	FTP	miren-lamp-*-SamplesSinceLastReport	SamplesSinceLastReport	15.9.2014	15.9.2014		15 min	1 min	
	FTP	miren-lamp-*-ReportSummaValue	ReportSummaValue	15.9.2014	15.9.2014		15 min	1 min	
	FTP	miren-lamp-*-ReportNo	ReportNo	15.9.2014	15.9.2014		15 min	1 min	
	FTP	miren-lamp-*-ReportAvgValue	ReportAvgValue	15.9.2014	15.9.2014		15 min	1 min	mA
	FTP	miren-lamp-*-MinValue	MinValue	15.9.2014	15.9.2014		15 min	1 min	mA
	FTP	miren-lamp-*-MeasuredConsumption	MeasuredConsumption	15.9.2014	15.9.2014		15 min	1 min	kWh
	FTP	miren-lamp-*-MaxValue	MaxValue	15.9.2014	15.9.2014		15 min	1 min	mA
	FTP	miren-lamp-*-HopCounter	HopCounter	15.9.2014	15.9.2014		15 min	1 min	
	FTP	miren-lamp-*-DimLevelCh2	DimLevelCh2	15.9.2014	15.9.2014		15 min	1 min	%
	FTP	miren-lamp-*-DimLevelCh1	DimLevelCh1	15.9.2014	15.9.2014		15 min	1 min	%
	FTP	miren-lamp-*-CalculatedConsumption	CalculatedConsumption	15.9.2014	15.9.2014		15 min	1 min	kWh
	webservice		traffic flow	15.10.2014	1.1.2014		10 min	10 min	cars/h
	webservice		traffic speed	15.10.2014	1.1.2014		10 min	10 min	km/h
	webservice		traffic density	15.10.2014	1.1.2014		10 min	10 min	
NTUA	FTP	ntua-building-*-ast_average_demand_r	lastaveragedemand_r	30.9.2014	14.10.2009	LAMPADARIO, HYDROLICS	1 day	15 min	kW
	FTP	ntua-building-*-last_average_demand_a	lastaveragedemand_a	30.9.2014	14.10.2009	33 alltogether - el. meters (Siemens) (31.12.2014).	1 day	15 min	kW
	FTP	ntua-building-*-energy_a	energy_a	30.9.2014	14.10.2009	16 el. meters (Schneider).	1 day	15 min	kWh
	FTP	ntua-building-*-current_l3	current_l3	30.9.2014	14.10.2009	availability of the data 15. 2. 2010 for HYDROLICS	1 day	15 min	A
	FTP	ntua-building-*-current_l2	current_l2	30.9.2014	14.10.2009		1 day	15 min	A
	FTP	ntua-building-*-current_l1	current_l1	30.9.2014	14.10.2009		1 day	15 min	A
GENERAL	webservice	WWO-*-WWO-cloudcover	cloudcover	1.11.2013	14.10.2009	Percent of the clear sky. (World Weather Online)	~5min	~5min	%
weather	webservice	WWO-*-WWO-humidity	humidity	1.11.2013	14.9.2013	Relative humidity.	~5min	~5min	%
	webservice	WWO-*-WWO-precipMM	precipitation	1.11.2013	14.9.2013	Precipitation in last hour.	~5min	~5min	mm

	webservice	WWO-*-WWO-pressure	pressure	1.11.2013	14.9.2013	Air pressure.	~5min	~5min	mbar
	webservice	WWO-*-WWO-temp_C	temp_C	1.11.2013	14.9.2013	Air temperature.	~5min	~5min	C
	webservice	WWO-*-WWO-temp_F	temp_F	1.11.2013	14.9.2013	Air temperature.	~5min	~5min	F
	webservice	WWO-*-WWO-visibility	visibility	1.11.2013	14.9.2013	Visibility.	~5min	~5min	km
	webservice	WWO-*-WWO-weatherCode	weatherCode	1.11.2013	14.9.2013	Internal WWO code of type of weather.	~5min	~5min	
	webservice	WWO-*-WWO-winddirDegree	winddirDegree	1.11.2013	14.9.2013	Wind direction.	~5min	~5min	deg
	webservice	WWO-*-WWO-windspeedKmph	windspeedKmph	1.11.2013	14.9.2013	Wind speed.	~5min	~5min	km/h
	webservice	WWO-*-WWO-windspeedMiles	windspeedMiles	1.11.2013	14.9.2013	Wind speed.	~5min	~5min	mph
	webservice	OWM-*-OWM-id	weatherCode	1.11.2013	14.9.2013	Weather code of OWM. (Open Weather Map).	~5min	~5min	
	webservice	OWM-*-OWM-temp	temperature	1.11.2013	14.9.2013	Air temperature.	~5min	~5min	C
	webservice	OWM-*-OWM-pressure	pressure	1.11.2013	14.9.2013	Air pressure.	~5min	~5min	mbar
	webservice	OWM-*-OWM-humidity	humidity	1.11.2013	14.9.2013	Relative humidity.	~5min	~5min	%
	webservice	OWM-*-OWM-deg	winddirection	1.11.2013	14.9.2013	Wind direction.	~5min	~5min	deg
	webservice	OWM-*-OWM-all	cloudcover	1.11.2013	14.9.2013	Percent of the clear sky.	~5min	~5min	%
	webservice	OWM-*-OWM-3h	precipitation_3h	1.11.2013	14.9.2013	Precipitation in last 3 hours.	~5min	~5min	mm
	webservice	OWM-*-OWM-1h	precipitation_1h	1.11.2013	14.9.2013	Precipitation in last hour.	~5min	~5min	mm
	webservice	WU-*-WU-cloudcover	cloudcover	1.10.2014	1.1.2010	Percent of the clear sky. (Weather Underground)	1h	1h	%
	webservice	WU-*-WU-humidity	humidity	1.10.2014	1.1.2010	Relative humidity.	1h	1h	%
	webservice	WU-*-WU-pressure	pressure	1.10.2014	1.1.2010	Air pressure.	1h	1h	hPa
	webservice	WU-*-WU-temperature	temperature	1.10.2014	1.1.2010	Air temperature.	1h	1h	C
	webservice	WU-*-WU-winddir	winddir	1.10.2014	1.1.2010	Wind direction.	1h	1h	°
	webservice	WU-*-WU-windspeed	windspeed	1.10.2014	1.1.2010	Wind speed.	1h	1h	m/s
forecast	webservice	FIO-*-FIO-temperature	temperature	1.10.2014	1.1.2010	Air temperature. (Forecast.io)	1h	1h	C
	webservice	FIO-*-FIO-pressure	pressure	1.10.2014	1.1.2010	Air pressure.	1h	1h	hPa
	webservice	FIO-*-FIO-windSpeed	windspeed	1.10.2014	1.1.2010	Wind speed.	1h	1h	m/s
	webservice	FIO-*-FIO-windBearing	winddir	1.10.2014	1.1.2010	Wind direction.	1h	1h	°
	webservice	FIO-*-FIO-humidity	humidity	1.10.2014	1.1.2010	Relative humidity.	1h	1h	%
	webservice	FIO-*-FIO-cloudCover	cloudcover	1.10.2014	1.1.2010	Percent of the clear sky.	1h	1h	%
EPEX	webservice	spot-ger-electricity-quantity	quantity	1.11.2013	1.1.2010	Quantity of traded energy.	-	1h	MWh

webservice	spot-ger-electricity-price	price	1.11.2013	1.1.2010	Price of energy.	-	1h	EUR
webservice	spot-fra-electricity quantity	quantity	1.10.2014	1.1.2010	Quantity of traded energy.	-	1h	MWh
webservice	spot-fra-electricity-price	price	1.10.2014	1.1.2010	Price of energy.	-	1h	EUR
webservice	spot-ch-electricity-quantity	quantity	1.10.2014	1.1.2010	Quantity of traded energy.	-	1h	MWh
webservice	spot-ch-electricity-price	price	1.10.2014	1.1.2010	Price of energy.	-	1h	EUR

D. Appendix – The report on Early Experiments on the Model Selection

D.1 Data gathering, description and preparation (CSI)

Data gathering:

Two different sources were used to gather the data: the dependent variables were gathered as energy consumption data of a building in Turin and the independent variables were gathered as weather condition data from the nearest weather station (Weather Online).

Data description:

The dependent variables were recorded at a 15 minute interval from June 1st 2011 at 0:30 to June 13th 2015 at 10:45 (the time that the data was downloaded) and include:

- **`buildingconsumptionnocooling`** – the energy consumption of the building without the consumption of the cooling system
- **`buildingcooling`** – the energy consumption of the cooling system alone
- **`buildingtotalconsumption`** – the total energy consumption of the building (roughly `buildingconsumptionnocooling` + `buildingcooling`)
- **`datacentrecooling`** – the energy consumption of the cooling system including only the data centre

The independent variables were recorded at non-regular intervals (every few minutes) from November 11th 2013 at 10:57 to June 13th 2015 at 12:15 (the time that the data was downloaded) and include:

- **`WeatherCode`** – an integer coded description of the current weather
- **`Temperature`** – the outside temperature in °C
- **`Pressure`** – the atmospheric pressure in millibars
- **`Humidity`** – the humidity in %
- **`Precipitation`** – the precipitation in mm
- **`WindSpeed`** – the speed of wind in km/h
- **`WindDirection`** – the direction of the wind in azimuth degrees
- **`CloudCover`** – the coverage of the sky with clouds in %
- **`Visibility`** – the visibility on a scale from 0 to 10 (10 meaning perfect visibility, 0 meaning "complete fog")

Data preparation:

The data preparation was performed in three steps:

1. Time-alignment of the data from different sources
2. Data cleaning and outlier removal
3. Generation/removal of features

Since data coming from the two sources was not time-aligned, a time-alignment step was performed, where all data was first put into the same time frame (from November 11th 2013 at 11:00 to June 13th 2015 at 10:45), meaning that almost 2.5 years of recorded dependent variables data were dropped due to not having

corresponding recordings of the independent variables. Since independent variables were recorded at non-regular intervals, all independent variables data had to be re-calculated to a 15 minute interval. The re-calculation was performed as follows: all the recordings of independent variables that "fell" in the interval of ±7 minutes around the recorded dependent variables were averaged to that interval, except the **`WeatherCode`** and the **`WindDirection`**, where the majority value has been taken (e.g.: if the dependent variables were recorded on November 18th 2013 at 15:30, all independent variable recordings for that same date between 15:23 and 15:37 had to be accordingly "merged" into a single recording – in our case 4 recordings fall into the specified interval, namely 15:25, 15:27, 15:29 and 15:31).

After time-alignment variables were inspected for inconsistencies (outliers, missing values, inconsistent values) and they were corrected. Figure 40 depicts the distribution of the values of all independent variables in the form of histograms.

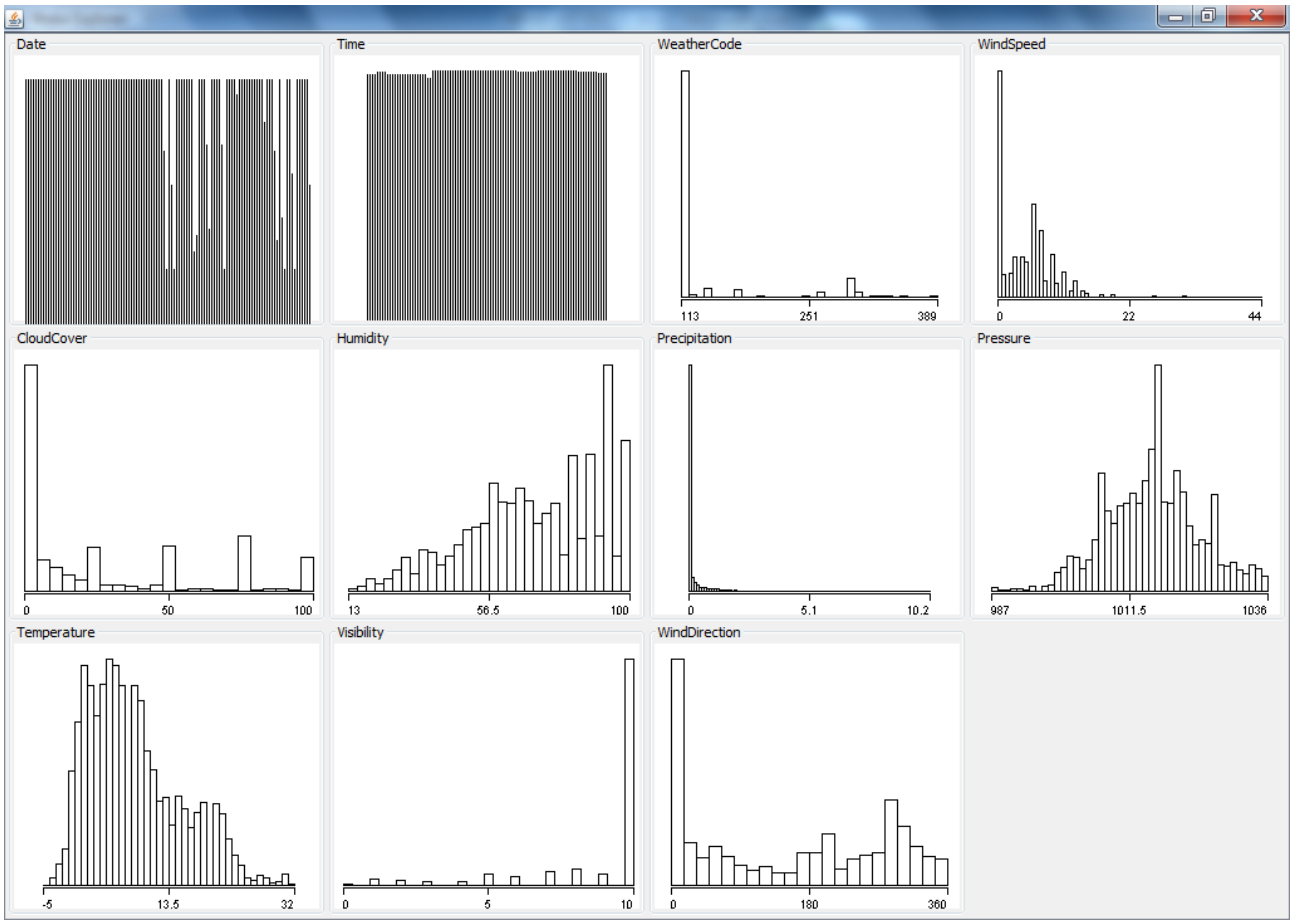


Figure 40: Histograms showing the distribution of values for independent variables

As we can see in Figure 40, some dates have fewer recordings, meaning there was no recording of the dependent variable for several 15 minute time-stamps of that day. No step was taken to "correct" this shortcoming. The other thing noticeable in Figure 40 is a surprisingly high number of recordings with **`WindDirection = 0`** and again no step was taken to address this issue.

Figure 41 shows how the 4 dependent variables change through time. Two peculiarities can be noticed from this figure:

- A sudden drop of the total energy consumption of the building on January 28th 2014 (probably due to the drop of cooling on the same day). No step was taken to account for this,
- Negative values for various types of energy consumption in some time-points. These negative values were substituted by the "unknown-value" tags that modelling algorithms will later handle accordingly.

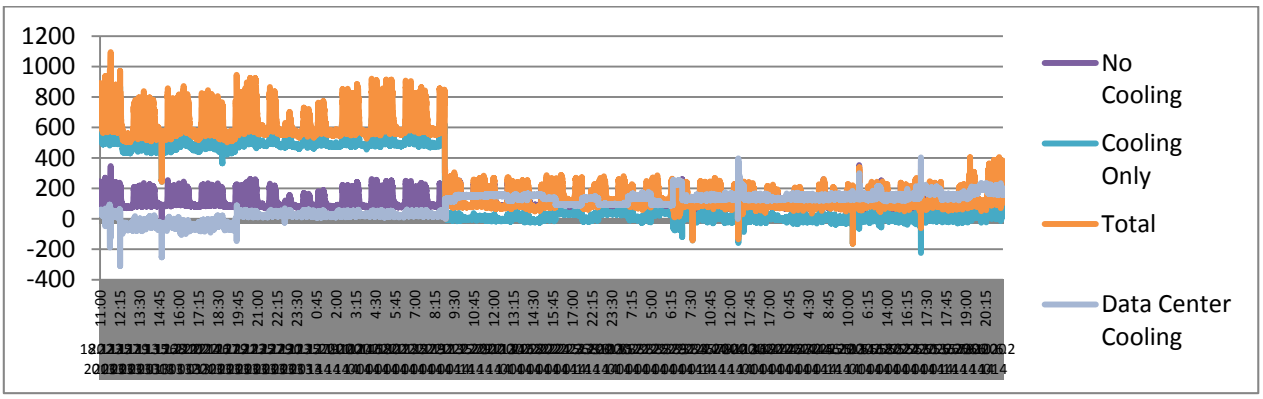


Figure 41: Changing of dependent variables through time

After this data-cleaning step an additional feature generation/removal step was undertaken, namely the actual time-stamp was replaced by 3 variables:

- **`DayOfWeek`** – an integer representation of the day of week (1 standing for Monday, …, 7 standing for Sunday);
- **`Hour`** – taking values from 0 to 23;
- **`Minute`** – representing the 15-minute interval (taking values 0, 15, 30, 45).

After this data preparation phase our data has 17,852 instances (15-minute interval recordings) and 16 variables (12 independent, of which 3 represent time and 9 representing weather condition and 4 dependent, representing various kinds of energy consumption of the building). Furthermore only the total energy consumption was retained as a single dependent variable, that is, the class attribute.

The following six subsection describe the models generated by data mining algorithms described in Section 4. All models were learned from a sample of two thirds of all available (pre-processed) data and tested on the remaining third. All algorithms were taken from the open source data mining suite WEKA [20] and ran with default parameters.

D.2 Linear Regression

The linear regression model generated from the data is the following:

```
Total =
    -20.4617 * DayOfWeek=6,3,5,1,4,2 +
     71.6284 * DayOfWeek=3,5,1,4,2 +
     10.5959 * DayOfWeek=5,1,4,2 +
      9.8938 * DayOfWeek=1,4,2 +
    -20.7519 * DayOfWeek=4,2 +
     23.2078 * DayOfWeek=2 +
      8.6967 * Hour +
    140.8433 * WeatherCode=200,356,119,386,332,176,389,335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
   -119.6562 * WeatherCode=356,119,386,332,176,389,335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
   -149.496  * WeatherCode=119,386,332,176,389,335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
    208.4965 * WeatherCode=386,332,176,389,335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
   -209.4192 * WeatherCode=332,176,389,335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
    216.049  * WeatherCode=176,389,335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
    -97.1775 * WeatherCode=335,263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
     56.7921 * WeatherCode=263,299,338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
   -121.6172 * WeatherCode=338,116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
    244.2384 * WeatherCode=116,143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
    -26.9285 * WeatherCode=143,113,323,326,293,302,122,296,266,308,329,182,317,248 +
     13.8832 * WeatherCode=113,323,326,293,302,122,296,266,308,329,182,317,248 +
    -59.5758 * WeatherCode=323,326,293,302,122,296,266,308,329,182,317,248 +
    -47.9729 * WeatherCode=326,293,302,122,296,266,308,329,182,317,248 +
    125.7415 * WeatherCode=293,302,122,296,266,308,329,182,317,248 +
     22.0138 * WeatherCode=296,266,308,329,182,317,248 +
     74.0742 * WeatherCode=266,308,329,182,317,248 +
     55.3749 * WeatherCode=308,329,182,317,248 +
     72.6822 * WeatherCode=248 +
     -3.2179 * WindSpeed +
      0.2794 * CloudCover +
     -1.1312 * Humidity +
     21.4226 * Precipitation +
      5.8183 * Pressure +
    -22.0362 * Temperature +
      3.2194 * Visibility +
     -0.2189 * WindDirection +
  -5610.3904
```

The correlation coefficient of the model is 0.5943.

D.3 SVM

The SVM model generated from the data is the following:

```
weights (not support vectors):
 +       0.0191 * (normalized) DayOfWeek=1
 +       0.0279 * (normalized) DayOfWeek=2
 +       0.0041 * (normalized) DayOfWeek=3
 -       0.0064 * (normalized) DayOfWeek=4
 +       0.0086 * (normalized) DayOfWeek=5
 -       0.0291 * (normalized) DayOfWeek=6
 -       0.0241 * (normalized) DayOfWeek=7
 +       0.1962 * (normalized) Hour
 +       0.003  * (normalized) Minute=0
 -       0.0031 * (normalized) Minute=15
 -       0.0012 * (normalized) Minute=30
 +       0.0013 * (normalized) Minute=45
 -       0.0974 * (normalized) WeatherCode=113
 -       0.0354 * (normalized) WeatherCode=116
 -       0.2952 * (normalized) WeatherCode=119
 -       0.0746 * (normalized) WeatherCode=122
 -       0.0602 * (normalized) WeatherCode=143
 -       0.1305 * (normalized) WeatherCode=176
 +       0.0341 * (normalized) WeatherCode=182
 -       0.0244 * (normalized) WeatherCode=200
 +       0.3434 * (normalized) WeatherCode=248
 -       0.1481 * (normalized) WeatherCode=263
 +       0.3448 * (normalized) WeatherCode=266
 -       0.0181 * (normalized) WeatherCode=293
 +       0.1264 * (normalized) WeatherCode=296
 -       0.0887 * (normalized) WeatherCode=299
 +       0.0885 * (normalized) WeatherCode=302
 +       0.2824 * (normalized) WeatherCode=308
 +       0.2211 * (normalized) WeatherCode=317
 -       0.065  * (normalized) WeatherCode=323
 -       0.0309 * (normalized) WeatherCode=326
 +       0.2631 * (normalized) WeatherCode=329
 -       0.1698 * (normalized) WeatherCode=332
 -       0.0506 * (normalized) WeatherCode=335
 -       0.1317 * (normalized) WeatherCode=338
 -       0.1623 * (normalized) WeatherCode=353
 -       0.079  * (normalized) WeatherCode=356
 -       0.0438 * (normalized) WeatherCode=386
 +       0.0019 * (normalized) WeatherCode=389
 -       0.075  * (normalized) WindSpeed
 -       0.0642 * (normalized) CloudCover
 -       0.0988 * (normalized) Humidity
 +       0.2727 * (normalized) Precipitation
 +       0.4664 * (normalized) Pressure
 -       0.7288 * (normalized) Temperature
 +       0.096  * (normalized) Visibility
 -       0.0832 * (normalized) WindDirection
 +       0.2717
```

The correlation coefficient of the model is 0.5667.

D.4 Model Trees

The M5 model tree algorithm [21][22] was used to model the data. It generated 736 rules, each representing a disjoint subset of the data that was further modelled using linear regression resulting in 736 linear.

The correlation coefficient of the model is 0.9371.

D.5 Artificial Neural Networks (ANN)

A variant of the ANN called the Multilayer Perceptron was used to model the data. The generated model consists of 24 nodes with corresponding weights for every input variable and a threshold. To get the fealing for this, this is how a single node looks like:

```
Sigmoid Node 1
    Inputs    Weights
    Threshold    -1.7688565370626024
    Attrib DayOfWeek=1    3.03947815886035
    Attrib DayOfWeek=2    -0.31183112230956667
    Attrib DayOfWeek=3    -0.7621087949479203
    Attrib DayOfWeek=4    1.6664049924245545
    Attrib DayOfWeek=5    2.8333331723617814
    Attrib DayOfWeek=6    3.581409770445421
    Attrib DayOfWeek=7    -1.0604938028673565
    Attrib Hour    -1.2691489794987887
```



```
Attrib Minute=0    0.8887891740804816
Attrib Minute=15    1.1076462104912703
Attrib Minute=30    0.8714059361311206
Attrib Minute=45    0.6906782273055353
Attrib WeatherCode=113    5.778358216511937
Attrib WeatherCode=116    -1.9553185138075844
Attrib WeatherCode=119    1.38608815862997
Attrib WeatherCode=122    -0.46994795377128684
Attrib WeatherCode=143    -0.42140151466252934
Attrib WeatherCode=176    0.4751867481688876
Attrib WeatherCode=182    0.23292298890049318
Attrib WeatherCode=200    0.34941673574581233
Attrib WeatherCode=248    0.19478931533246832
Attrib WeatherCode=263    0.08810761095626787
Attrib WeatherCode=266    0.3720836681698311
Attrib WeatherCode=293    -0.8306460918926067
Attrib WeatherCode=296    -2.2321462973013295
Attrib WeatherCode=299    1.3272154546714643
Attrib WeatherCode=302    -0.0952244169914875
Attrib WeatherCode=308    0.28508424890049083
Attrib WeatherCode=317    0.11656199686276643
Attrib WeatherCode=323    0.24720092794065093
Attrib WeatherCode=326    0.7243462962696049
Attrib WeatherCode=329    0.3184178971345647
Attrib WeatherCode=332    0.35451847649806967
Attrib WeatherCode=335    0.31678247177525154
Attrib WeatherCode=338    0.32762849249168124
Attrib WeatherCode=353    0.4400000874991278
Attrib WeatherCode=356    0.4604521146950918
Attrib WeatherCode=386    0.3183336940971858
Attrib WeatherCode=389    0.24036867808065643
Attrib WindSpeed    1.0499613404182127
Attrib CloudCover    4.794893581388606
Attrib Humidity    0.8219888416843283
Attrib Precipitation    1.6802240092693668
Attrib Pressure    -0.5478267615472964
Attrib Temperature    -0.7687480191255939
Attrib Visibility    7.487504651951733
Attrib WindDirection    2.5722903308465517
```

The correlation coefficient of the model is 0.7713.

D.6 Conclusions on model selection

Some basic regression data mining algorithms were tried in order to model the presented (pre-processed) data. The task at hand was to generate a model that would explain the total energy consumption of a building in Turin as being dependent on the outside weather conditions.

The algorithm that performed best was the M5 model tree that was able to explain 93.71% of the energy consumption dependency from outside weather conditions.

However, to be able to predict future energy consumption, future weather conditions are needed as well. This fact makes our modelling approach of limited use, since predicting weather in the future (weather forecast) can presently be done reliably just for a few days ahead.

New modelling methods that include time series analysis will thus be tried to overcome the described shortcoming of the analysed methods.

