XM_HeatForecast: Heating Load Forecasting in Smart District Heating Networks

Federico Bianchi*, Francesco Masillo*, Alberto Castellini, Alessandro Farinelli

Verona University, Department of Computer Science,
Strada Le Grazie 15, 37134 Verona, Italy,
{federico.bianchi, francesco.masillo, alberto.castellini,
alessandro.farinelli}@univr.it
* These authors contributed equally to this work.

Abstract. Forecasting is an important task for intelligent agents involved in dynamical processes. A specific application domain concerns district heating networks, in which the future heating load generated by centralized power plants and distributed to buildings must be optimized for better plant maintenance, energy consumption and environmental impact. In this paper we present XM_HeatForecast a Python tool designed to support district heating network operators. The tool provides an integrated architecture for *i)* generating and updating in real-time predictive models of heating load, *ii)* supporting the analysis of prediction performance and errors, *iii)* inspecting model parameters and analyzing the historical dataset from which models are trained. A case study is presented in which the software is used on a synthetic dataset of heat loads and weather forecast from which a regression model is generated and updated every 24 hours, while predictions of load in the next 48 hours are performed every hour.
Software available at: https://github.com/XModeling
Video available at: https://youtu.be/JtInizI4e_s.

Keywords - Forecasting; smart grids; predictive modeling; interpretability.

1 Introduction

Forecasting future behaviours of complex dynamical systems is a key functionality of intelligent agents involved in dynamical processes [18]. It entails the ability to learn patterns and variable relationships from past data (usually in the form of a multivariate time series), to generate a model of these patterns, and to use this model to infer future values of some variables of interest [4,5,3,?]. In the context of the recently proposed paradigm of smart grids, forecasting has gained a key role [17], since it allows to predict future loads of the network in order to improve network maintenance and efficiency [15]. A specific type of smart grids, which have proven important in recent years for environmental sustainability are District Heating Networks (DHNs) [9], in which the heat is generated in one or more centralized power plants and distributed through an insulated network of pipe system to commercial and residential buildings [10].

Short-Term Load Forecasting (STLF) [8,12,13,14], namely, the prediction of the system load over time intervals ranging from one hour to one week, is particularly important in DHN management because it supports operators in various decision-making tasks, including supply planning, generation reserve, system security and financial planning [8]. Different kinds of methods are proposed in the literature for time-series forecasting applied to DHNs and related domains. In [1], for instance, authors investigate the application of support vector regression, regression trees, feed forwards and multiple linear regression models applied to DHNs. Considering models based on deep learning, Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) [11] and convolutional LSTM [19] are among the most popular techniques for time series forecasting. Computational tools for STLF in DHNs consist of statistical and machine learning methods able to *i)* learn predictive models from past data, *ii)* update predictive models online while new data are available, *iii)* make load predictions, *iv)* analyze prediction performance, *v)* analyze model parameters. Our aim is to develop of an open source prototype for research in heating load forecasting which can easily be released in production. To the best of our knowledge the literature does not provide similar tools, while there exist commercial software.

In this paper we present a novel open source Python software for STLF in smart district heating networks. Its name is *XM_HeatForecast* and it extends the open source suite *eXplainable Modeling* (XM) [6,7]. *XM_HeatForecast* is designed to support DHN operators in all phases of load forecasting and it can be interfaced with other tools for planning and scheduling. The tool provides *i)* a module for predictive model generation based on time series, *ii)* a graphical user interface for integrated and interactive analysis of predictions, models performance and data. The current (first) version of the software uses linear autoregressive models for prediction, where variables are generated from prior knowledge on the application domain, but the tool is independent of the prediction method and the data sources. The main contribution of this work to the state-of-the-art is to presentation of the software *XM_HeatForecast* and all its components. Moreover, we provide a case study in which the tool is tested on a synthetic but realistic dataset showing the differences in prediction performance depending on the length of the historical data. Model coefficients are also investigated to show how the software supports model interpretability through integration of different kinds of information. Other original elements are the online approach by which the model is updated and the open communication interface for data exchange with other tools, such as planners.

The rest of the manuscript is organized as follows. Section 2 presents an overview of *XM_HeatForecast*. In Section 3 a detailed description of modules for model and forecast generation is provided. In Section 4 modules for visualization and analysis are presented. In Section 5 communication interface is described. Section 6 contains a complete case study. Conclusions and future directions are discussed in Section 7.

2 XM_HeatForecast: system overview

In this section we provide a high-level description of the *XM_HeatForecast* architecture, and an overview of its main modules and testing dataset.

2.1 Main modules and software structure

XM_HeatForecast has four main components, namely, *Data processor*, *Model generator*, *Forecaster* and *Graphical user interface* (GUI). In Figure 1 each part is identified by a color to highlight different functionalities (i.e., green, gray, light blue and red). In the current version the tool is designed to re-train models every 24 hours and predict future heating loads every hour. When the tool is started[1] a forecasting horizon between 24 or 48 hours and a starting instant (date/time) for the predictions are asked to the user. The values of the first parameter have been suggested by operators of a real DHN plant because of their usefulness in plant operations (e.g., production and maintenance planning). The second parameter allows to perform tests on previously acquired datasets. In this case also a refresh time can be provided by the user to speed-up the simulation. Namely, instead of waiting one hour for each update, simulations can be performed with a time interval of few seconds, as shown in the attached video.

Input/output: the input and output of the tool consists of time series data read from folders *Weather forecast*, *Data* and *Model*, and predictions stored in folder *Forecast* (see the folder icons in Figure 1). The folders contain comma separated (*csv*) files automatically read and written in real-time. **Data processor**: this module reads batches of data from folders *Weather forecast* and *Data* (green arrow 1.a in Figure 1), processes and re-stores them into folder *Data* (green arrow 1.b in Figure 1). Processed data are then loaded by *Model generator* and *Forecaster* for training and forecasting. **Model generator**: it reads past load data from folder *Data* (purple arrow 2.a), trains the model and saves it into folder *Model* (purple arrow 2.b). Past models are stored in the same folder for analysis purposes. **Forecaster**: every hour, it loads processed data and the last trained model from directories *Data* and *Model* (blue arrow 3.a), respectively, and it predicts the future heating load (blue arrow 3.b). **Performance measures**: Performance is computed at run-time when forecasting from time series of predicted load and real load. The current model is then evaluated by root mean square error on testing data (i.e., data not used for training

[1] *XM_HeatForecast* requires Python 3.X. A script is provided to automate the installation process. The script is executed by command *./install.sh*.

the model). The coefficient of determination (R^2) of the models are computed on training set and model parameters are extracted from the current model and saved into the *Performance* folder (orange arrows 4.a and 4.b in Figure 1). **GUI**: it is a graphical user-friendly interface that updates in real-time showing current predictions, errors, model parameters and historical data. **Communication interface**: module synchronization is performed at run-time by file exchange (red arrows 5.a, 5.b and 5.c in Figure 1). Figure 2 shows folder organization.

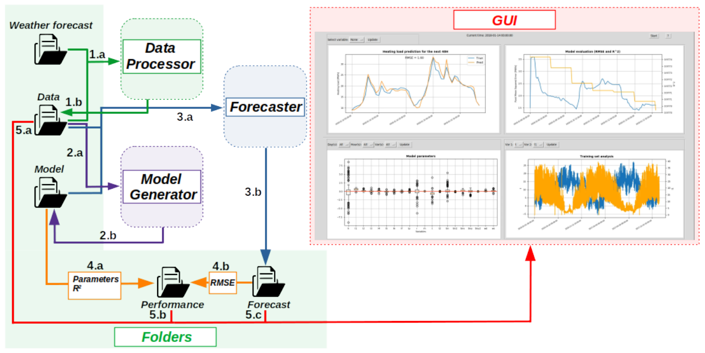


Fig. 1. Overview of the XM_HeatForecast.

2.2 Testing dataset

The current version of tool provides a dataset for testing the software, which it includes weather forecasts provided by an external forecast repository [2], calendar events (social factors) [3] and synthetic heating loads. Main weather factors that affect heating load are considered, namely temperature (T), relative humidity (RH), rainfall (R), wind speed (WS), wind direction (WD). From these signals we computed variables with strong predictive capabilities for heating load, according to the literature [2,16]. In particular, from T we compute squared of temperature T^2, moving average of temperature on last 7 days $T_{ma(7)}$, maximum of temperature T_M, squared of maximum temperature T_M^2, maximum temperature of a day ago T_{Mp} and square of maximum temperature of a day ago T_{Mp}^2. Historical load variables used by the model are heating load l_i of i days ago for $1 \leq i \leq 7$ and load peak at previous day l_p. The software is however independent of the set of variables used by the model.

[2] `https://rp5.ru/`

[3] `https://pypi.org/project/workalendar/`

2.3 Data pre-processing

The *Data processor* module transforms raw data to data in a models-readable format. Source data are *.csv* files. *Data processor* reads from these files the historical and current heat load and weather forecast, merges them and it computes the variables described in the previous subsection. Finally it saves processed data in the *Data* folder. This procedure is displayed by green arrows 1.a and 1.b in Figure 1.

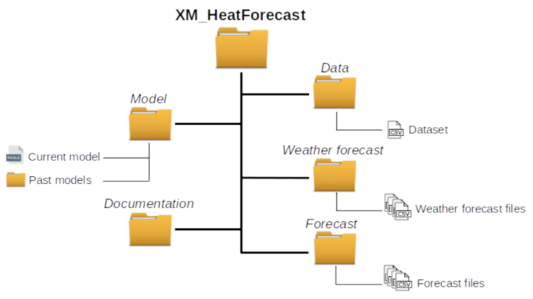


Fig. 2. Folder organization.

3 Model generation and forecasting

In this section we describe the modules that manage predictive models generation, parameters updating and forecasting of heating load.

3.1 Model generator

The *XM_HeatForecast* architecture is designed to be independent from the choice of a specific model. In the current version of the tool we used autoregressive linear models models generated by the approach proposed in [2]. In particular, the model consists of multiple linear regression equations, one for each pair (day, hour), therefore we have 168 equations in total. The model estimates future values of heating load l using a linear combination of weather variables, past loads and social factors described in the previous section. The equations have the following form: $M_{ij} = f(weather, dload, social)$, where $i \in \{1, 7\}$ indicates the day of week, $j \in \{0, \ldots, 23\}$ the hour of the day, *weather* are weather variables, *dload* includes past heating load, and *social* consists of variables related to calendar events, such as holidays.

Given a series of pre-processed data read from the *Data* folder, the *Model generator* generates the model and updates parameters during the re-training process performed every 24 hours. Each equation M_{ij} is trained using corresponding day/hour data. The current version of the software uses the Ordinary

Least Squares (OLS) function of *statsmodels*[4] (version 0.11.1) since it natively provides statistics such as parameter p-values, but other libraries (such as *scikit-learn*[5]) can be used to employ other modeling frameworks. Models are stored in the *Model* folder using the *dump* function of the *Pickle* library. The procedure is displayed by purple arrows 2.a and 2.b in Figure 1.

3.2 Forecaster

The *Forecaster* is responsible of predictions of future heating load. The architecture of the tool allows to easily select the amount of forecasting jobs and a forecasting horizon between 24 or 48 hours. In the current version we schedule a job every hour and batch of input data together with updated model are loaded by this module. According to a selected forecasting horizon h, *Forecaster* predicts the future h-hours of heating load, using for each day of the week (i) and for each hour of the day (j) the correspondent model equation M_{ij} and the data of the last week. The forecasting procedure is displayed by blue arrows 3.a and 3.b in Figure 1.

3.3 Performance evaluation

Two standard metrics are provided to evaluate performance measures, namely, the root-mean-squared error (RMSE) computed on the test set, and the coefficient of determination R^2 computed on the training set. Given an observed time-series with n observations $y_1, \ldots, y_n$ and predictions $\hat{y}_1, \ldots, \hat{y}_n$, the formula for the root-mean-squared error is: $\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2}$. The RMSE is used in two ways. First, to evaluate the error of the last prediction every time a new forecasting is performed and real heating load are available. Second, to show the error trend on a rolling basis using a sliding window of h-hours, where h is the forecasting horizon. Every time the model is re-trained the GUI is refreshed to show updated R^2 coefficients and model parameters. The performance evaluation procedure is displayed by orange arrows 4.a and 4.b in Figure 1.

4 Model and forecast visualization and analysis

The *XM_HeatForecast* graphical user interface is divided into four main sections, identified by numbers from 1 to 4 in Figure 3 and described in the following. All charts can be separately visualized in independent windows that allow zooming and saving to file.

[4] https://www.statsmodels.org
[5] https://scikit-learn.org

4.1 Visualization of heating load predictions

The first section of the GUI (top-left) contains future heating load prediction for h-hours after the current date/time instant. The current date and time is displayed in the top-central part of the interface. The chart shows the prediction in orange and the real load in blue (if available). The RMSE of the prediction is also displayed on top of the chart. The menu on top of this chart allows to select and visualize weather variables, i.e., T, RH, WS, WD and R. If a variable is selected, then the chart is refreshed to display the values of the corresponding variable during the h hours of the last prediction. This allows operators to discover relationships between heating load predictions, prediction errors and weather factors.

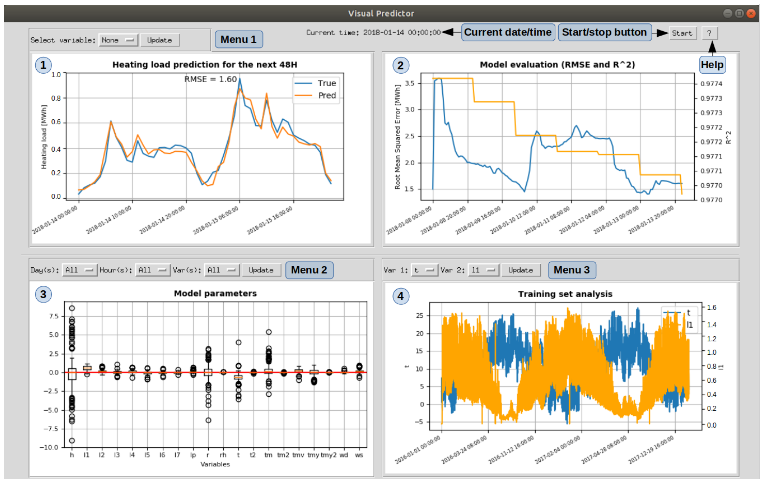


Fig. 3. Main elements of the *XM_HeatForecast* graphical user interface.

4.2 Visualization of model and prediction performance

The second section of the the GUI (top-right) displays the time evolution of two performance indicators, namely, the coefficient of determination R^2 (orange line) and the RMSE of h-hours prediction (blue line). This chart is updated every hour, namely whenever a new prediction is performed, but the R^2 statistics is updated only every 24 hours, when the model is re-trained. Figure 3 shows, for instance, 15 days of prediction errors. These chart is useful to assess the prediction quality over time, as shown in the case study of Section 6.

4.3 Visualization of model parameters

The third section of the GUI (bottom-left) shows model parameters. These values are important for DHN operators because they provide insight about variables contribution to the prediction. This insight can strongly support decision making by DHN operators. The box plot displayed in Figure 3 shows, for each variable, the distribution of related coefficient values across the 168 equations (e.g., temperature T has a negative median value, see the red horizontal line in the corresponding box plot). Menu 2, on top of this chart, allows the operator to select various combinations of the three factors *day*, *hour* and *variable*, to deepen the analysis of coefficients. By selecting a specific day (e.g., Monday) the box plot is updated by a heatmap in which rows represent variables, columns represent hours and cell colours represent coefficient values for specific day, hour and variable. Heatmaps are used to visualize coefficients when single factors are selected in menu 2. When two factors are selected a table is visualized to show related coefficient values.

4.4 Visualization and analysis of training data

The last section (bottom-right) allows the operator to select two variables of the training set (i.e., the dataset on which the model has been trained) and analyze them. In Figure 3 the temperature (blue line) and the heating load (orange line) have been selected, and their negative correlation is displayed.

5 Communication interface

In this section we describe the communication interface of the tool consisting of back-end with *Data processor*, *Model generator* and *Forecaster* and front-end which is the GUI. Each reading/writing operation performed by back-end modules is synchronized with reading of front-end. An overview of the system and a description of folders content are displayed in Figures 1 and 2.

5.1 Back-end

In the following we describe the back-end and a step-by-step process that allows the core of the tool to be synchronized with front-end. Every hour when a forecasting job is scheduled, a new file containing weather forecast for the next hour and past data are read. Past data consists of historical heating load of the last week (i.e., l_i of i days ago ($1 \leq i \leq 7$), heating load peaks at previous day l_p and calendar events such holidays H. Current weather forecast and past data are then merged and processed by *Data processor*, which returns a batch having new variables computed from current value of T (see Section 2.2). The new batch of data is saved into *Data* folder in order to be read by *Forecaster* in forecasting phase or GUI for graphical visualizations and it has 168 rows and 20 columns, which are a week of past observations for 20 variables in accordance with model requirements (see Section 3.2). Every 24 hours a whole set of historical data updated with the observations of the previous day is passed to *Model generator* for the training and the update of model parameters. *Model* folder contains current model and a chronology of past models saved as *.pickle* objects. Every hour, *Forecaster* generates an output file which is saved into *Forecast* folder. Each file contains the future *h*-hours of predicted heating load.

The metrics for the evaluation of performance are computed at run-time in two moments, the first, every 24 hours R^2 coefficients and model parameters are saved into *Performace* folder and they are updated and available as soon as a training instance is completed. Due to the type of model, a *csv* file containing R^2 coefficients has the following structure: 24 rows and 7 columns, one for each pair day of the week (j) and hour of the day (i). Each position i, j provides a coefficient of determination of model in a specific day (j) and hour (i). Model parameters are also saved in *csv* files. A file containing the parameters of a specific variable has 24 rows and 7 columns. As the previous metric, each position contains a weight for specific day of the week and hour (i, j). The second moment, every hour RMSE is computed given the last series of predicted and real load.

5.2 Front-end

GUI checks if new data are available using a polling technique on *Forecast*, *Performance* and *Data* folders. It reads last forecasting file and if available, real heating load to display the top-left graph in Figure 3, that it consists of a comparison between real and predicted load with related RMSE. It loads R^2 coefficients and RMSE values to display the graph on the top-right, which shows the performance of model since tool has been starting. It loads model parameters to display box-plot charts, heatmaps or tables with model parameters on the bottom-left. Finally, GUI reads dataset used for training the models to support data analysis on the bottom-right.

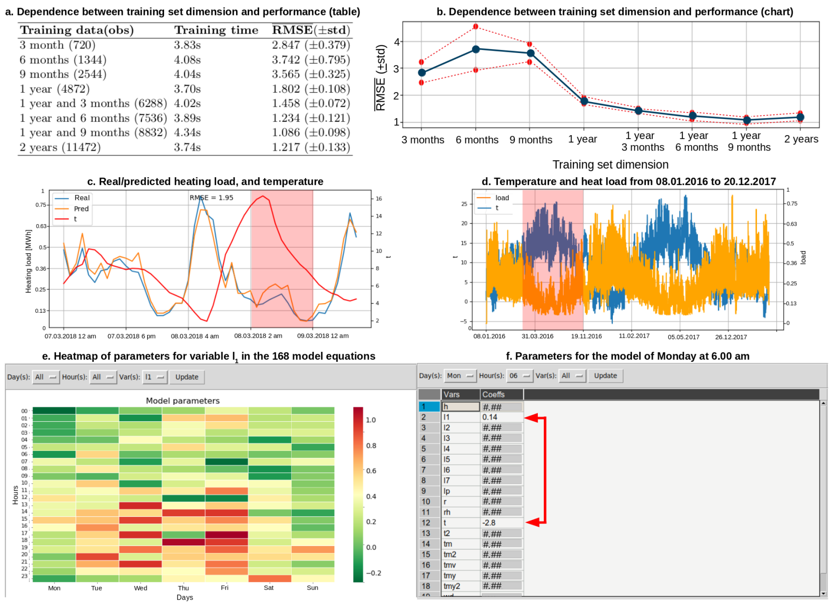
6 Case study

We provide a case study showing the standard use of *XM_HeatForecast* and the advantages it introduces from the data analysis point of view by integrating machine learning and visualization tools. The analysis here presented is based on a real-world dataset[6] provided by AGSM[7] a utility company that manages a DHN in Verona (Italy). The dataset contains variables described in Section 2.2 acquired from 01.01.2016 to 30.04.2018. In these tests we compute multi-equation linear autoregressive models with one equation for each weekday-hour, as described in Section 3.1. Further details about models are provided in [2]. The experiment focuses, in particular, on the analysis of model performance depending on the number of observations in the training set. As explained above, *XM_HeatForecast* performs model re-training every 24-hours. We evaluate eight training intervals displayed in Figure 4.a and test the model on twelve 48-hours forecasts starting from a fixed date, namely, 19.02.2018 at 12 am. The table in Figure 4.a shows, in each row, the training length (in monts and number of observations) and related $\overline{RMSE}$. The chart in Figure 4.b displays the same data

[6] The dataset is not available for privacy reasons.

[7] `https://www.agsm.it/`

on a graphical basis. Interestingly, performance reaches a plateau after about one year and a half, showing that this time horizon is needed to have have good prediction performance. We provide also the training time which is nearly constant. Another interesting analysis allowed by *XM_HeatForecast* concerns the relationship between independent variables (i.e., weather and social factors) and the heating load prediction (and error). Operators are interested to understand, in particular, the reasons of model errors to increase the knowledge about the network and improve model performance. The red line in Figure 4.c shows, for instance, a case in which the maximum temperature strongly increases between 03.07.2018 (about 10°C) and 04.07.2018 (about 16°C) (see the red area). This change seems to be the cause of a deviation between the true and the predicted load due to the slowness of the model to adapt to temperature changes. In fact, in the temperature peak the model predicts a higher load than the true one for about 7 hours, as if it expected lower temperature at that time.



Training data(obs)	Training time	$\overline{\text{RMSE}}$(±std)
3 month (720)	3.83s	2.847 (±0.379)
6 months (1344)	4.08s	3.742 (±0.795)
9 months (2544)	4.04s	3.565 (±0.325)
1 year (4872)	3.70s	1.802 (±0.108)
1 year and 3 months (6288)	4.02s	1.458 (±0.072)
1 year and 6 months (7536)	3.89s	1.234 (±0.121)
1 year and 9 months (8832)	4.34s	1.086 (±0.098)
2 years (11472)	3.74s	1.217 (±0.133)

Day(s): Mon Hour(s): 06 Var(s): All Update

	Vars	Coeffs
1	h	#.##
2	l1	0.14
3	l2	#.##
4	l3	#.##
5	l4	#.##
6	l5	#.##
7	l6	#.##
8	l7	#.##
9	lp	#.##
10	r	#.##
11	rh	#.##
12	t	-2.8
13	t2	#.##
14	tm	#.##
15	tm2	#.##
16	tmv	#.##
17	tmy	#.##
18	tmy2	#.##

Fig. 4. Example of data analysis with *XM_HeatForecast*

Finally, we show how *XM_HeatForecast* can support knowledge extraction from model parameters. The multi-equation autoregressive linear model generated by the current version of the software consists of 168 linear regression equations, one for each weekday and hour. With 20 independent features this results in 3360 coefficients, which are difficult to display efficiently. The bottom-left panel of *XM_HeatForecast* (described in Section 4.3) provides useful support on the analysis of these parameters. Selecting a variable from the menu of this panel a heatmap of model parameters is shown. For instance the heatmap displayed in Figure 4.e displays 168 coefficients of variable l_1, namely the heating load with one day lag. Heatmap columns correspond to the 24 hours of the day, and rows correspond to the 7 days of the week. The chart allows to easily observe patterns and other properties of model parameters. For instance, red cells in the figure correspond to pairs weekday-hour in which the predicted heating load highly depends on the load of the previous day at the same hour. In fact, these cells represent high coefficients (i.e., high correlation) while green cells low coefficients. Notice that Sunday and Monday have low parameters on average. In the first case this is due to the strong change of behaviour between working days and Sunday (mainly for commercial buildings), in the second case it is due to the strong change of behaviour between Sunday (weekend) and Monday (working day), which makes the load of the previous day not informative to predict future load. *XM_HeatForecast* allows also to display numerical values of model parameters, as shown in Figure 4.f, by selecting a specific day of the week and hour of the day in the menu. The figure shows coefficients for each variable on Monday at 6.00 and the red arrow indicates opposite signs of temperature T and previous day load l_1 coefficients, which highlights that the temperature T is anti-correlated with the predicted load, and the load of the previous day is little but positively correlated with the predicted load. The anti-correlation between temperature and heating load can be clearly shown also in the bottom-right panel of *XM_HeatForecast* (see Section 4.4). Figure 4.d displays the time evolution of the two variables over a time period of two years in the training set. It shows a strong seasonality with low temperature and high heating load in winter and high temperature and low heating load in spring. By zooming on the chart a similar pattern is visualized also on a daily basis, because of the temperature variation between day and night.

7 Conclusions

XM_HeatForecast integrates in a single tool both machine learning and data visualization capabilities for heating load forecasting in DHN. The tool allows to generate predictive models and analyze their parameters, predictions and performance, supporting interactive and integrated analysis in real-time. This integration provides new possibilities for DHN operators to improve their knowledge about the network and, consequently, to improve operational performance. The paper describes the tool and provides a novel case study on a real dataset where the software tools are used to collect knowledge about the real network and to show the prediction ability. Future developments concern the extensions to a other predictive modeling frameworks (e.g., LSTM) and the application to novel domains, such as behaviour prediction in intelligent robotic platforms.

Acknowledgments

The research has been partially supported by the projects “Dipartimenti di Eccellenza 2018-2022”, funded by the Italian Ministry of Education, Universities and Research (MIUR), and “GHOTEM/CORE-WOOD, POR-FESR 2014-2020”, funded by Regione del Veneto.

References

1. S. Bandyopadhyay, J. Hazra, and S. Kalyanaraman. A machine learning based heating and cooling load forecasting approach for DHC networks. In *IEEE Power Energy Soc. Innov. Smart Grid Tech. Conf. (ISGT)*, pages 1–5, Feb 2018.
2. F. Bianchi, A. Castellini, P. Tarocco, and A. Farinelli. Load forecasting in district heating networks: Model comparison on a real-world case study. In *Machine Learning, Optimization, and Data Science - LOD*, pages 553–565. Springer, 2019.
3. A. Castellini, G. Beltrame, M. Bicego, D. Bloisi, J. Blum, M. Denitto, and A. Farinelli. Activity recognition for autonomous water drones based on unsupervised learning methods. In *Proc. 4th Italian Workshop on Artificial Intelligence and Robotics (AI*IA 2017)*, volume 2054, pages 16–21, 2018.
4. A. Castellini, M. Bicego, F. Masillo, M. Zuccotto, and A. Farinelli. Time series segmentation for state-model generation of autonomous aquatic drones: A systematic framework. *Engineering Applications of Artificial Intelligence*, 90, 2020.
5. A. Castellini, F. Masillo, M. Bicego, D. Bloisi, J. Blum, A. Farinelli, and S. Peigner. Subspace clustering for situation assessment in aquatic drones. In *Proceedings of the Symposium on Applied Computing, SAC 2019*, page To appear. ACM, 2019.
6. A. Castellini, F. Masillo, R. Sartea, and A. Farinelli. eXplainable Modeling (XM): Data analysis for intelligent agents. In *Proc. 18th Int. Conference on Autonomous Agents and MultiAgent Systems*, AAMAS ’19, page 23422344. International Foundation for Autonomous Agents and Multiagent Systems, 2019.
7. eXplainable Modeling code. https://github.com/XModeling/XM, 2019.
8. S.N. Fallah, M. Ganjkhani, S. Shamshirband, and K.-W. Chau. Computational intelligence on short-term load forecasting: A methodological overview. *Energies*, 2019.
9. T. Fang. Modelling district heating and combined heat and power, 2016.
10. T. Fang and R. Lahdelma. Evaluation of a multiple linear regression model and SARIMA model in forecasting heat demand for district heating system. *Applied Energy*, 179:544–552, 2016.
11. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.
12. G. Gross and F.D. Galiana. Short-term load forecasting. *Proceedings of the IEEE*, pages 1558–1573, 1987.
13. M.T. Hagan and S.M. Behr. The time series approach to short term load forecasting. *IEEE Transactions on Power Systems*, pages 785–791, 1987.
14. Maria Jacob, Cláudia Neves, and Danica Vukadinović Greetham. *Short Term Load Forecasting*, pages 15–37. Springer International Publishing, Cham, 2020.
15. P. Mirowski, S. Chen, T.K. Ho, and C.N Yu. Demand forecasting in smart grids. *Bell Labs Technical Journal*, pages 135–158, 2014.
16. R. Ramanathan, R. Engle, C.W.J. Granger, F. Vahid-Araghi, and C. Brace. Short-run forecast of electricity loads and peaks. *International Journal of Forecasting*, pages 161–174, 1997.

17. M. Raza and A. Khosravi. A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renewable and Sustainable Energy Reviews*, 50:1352–1372, 2015.
18. Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.
19. T. Sainath, O. Vinyals, A. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE - (ICASSP)*, pages 4580–4584, 2015.