FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation:
Master of Science in Applied Data Science
Master of Science in Computer Science

The spring semester, 2022

Confidential

Author:
Aleksander Bogunovic Jakobsen
Unni Johanna Blilie Kinstad

Aleksander B Jakobsen [illegible]
(signature author)

Course coordinator: Nejm Saadallah

Supervisor(s):

Nejm Saadallah (Internal supervisor - UiS)
Tore Wennberg (External supervisor – Veni AS)
Eirik Svanes (External supervisor – Veni AS)

Thesis title:

Smart Building Data Collection and Ventilation System Energy Prediction

Credits (ECTS): 30

Keywords:
Smart buildings
Sensor data collection
Ventilation system
Indoor climate optimization
Control strategies
Energy prediction

Pages: 95

+ appendix: 10

Stavanger, 15/06/2022

ALEKSANDER BOGUNOVIC JAKOBSEN & UNNI JOHANNA BLILIE KIN-STAD

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Smart Building Data Collection and Ventilation System Energy Prediction

Master's Thesis - Applied Data Science & Computer Science - June 2022

We, **Aleksander Bogunovic Jakobsen & Unni Johanna Blilie Kinstad**, declare that this thesis titled, "Smart Building Data Collection and Ventilation System Energy Prediction" and the work presented in it is our own. We confirm that:

- This work was done wholly or mainly while in candidature for a master's degree at the University of Stavanger.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work.
- We have acknowledged all main sources of help.

“Any sufficiently advanced technology is indistinguishable from magic.”

– Arthur C. Clarke

Abstract

Data has the potential to transform our environments for the better if utilized to its full potential. A highly interesting use case of data is in relation to Smart Buildings, where IoT technology presents new possibilities. With appropriate collection and structuring of the available data, many new opportunities present themselves.

In this thesis, a data gathering system is proposed for sensors in Arkivenes Hus. To illustrate the potential in the data, one specific problem is researched, namely that of indoor climate optimization and its effects on energy usage. The problem description and the development of the data system comprises identifying governing system equations using sparse identification of nonlinear dynamics, control strategy using model predictive control and various machine learning methods to predict energy usage.

For a one day simulation, the proposed optimization strategy yields a 174.86% increase in energy usage. The conducted work indicates that the proposed model identification technique is unsuitable for the underlying data utilized in this work. The proposed model predictive control strategy and machine learning methods contain promising results.

Acknowledgement

First and foremost, we want to extend an earnest thanks to Veni AS and Knut Raeng for granting us the opportunity to be a part of such a interesting and important project. A special thanks to Eirik Svanes for all valuable insight and eager assistance during the duration of this thesis. Thanks also to Tore Wennberg and the rest of the project team for all the knowledge sharing and support along the way. We would also like to express our gratitude to Nejm Saadallah, supervisor at the Department of Electrical Engineering and Computer Science at the University of Stavanger for his guidance and supervision through this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

BREEAM Building Research Establishment Environmental Assessment Method

IoT Internet of Things

DT Digital twin

BIM Building information model

SINDy Sparse identification of nonlinear dynamical systems

MPC Model Predictive Control

LR Linear Regression

LSTM Long short-term memory netowrks

SVM Support Vector Machine

SVR Support Vector Regression

XGBoost Extreme gradient boost

MSE Mean Squared Error

MAE Mean Absolute Error

RMSE Root Mean Squared Error

DTDL Digital Twin Definition Language

JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
SQL	Structured Query Language
PaaS	Platform as a Service
SaaS	Software as a service
IaaS	Infrastructure as a service
T-SQL	Transact-SQL
API	Application Programming Interface
RBF	Radial basis function

Chapter 1

Introduction

Global energy production is currently transitioning from fossil fuels toward renewable energy sources as a result of climate changes. Renewable energy sources such as solar and wind are transitory in their nature, and cannot generate electricity on demand like fossil fuel sources such as coal plants can. In addition to the energy production aspect of adapting to changing climate, most industries and sectors are working towards minimizing emissions and energy usage [61], [68], [79]. Within the construction industry, modern buildings are planned and built to be more energy efficient as one response to the problem. Among other things, buildings are getting more airtight to achieve this, giving excellent thermal protection and minimizing the demand for energy-powered heating [9].

With rising energy prices and energy demand due to electrification, as well as fluctuating grid properties, forecasting energy requirements is becoming increasingly important. Monitoring energy consumption allows users to be more conscious of their consumption and adjust the way energy-intensive systems are utilized. For large office buildings in particular, predicting future energy consumption can provide two-sided benefits. First off, building managers can get insight into key elements affecting their buildings energy demand. Secondly, and perhaps most importantly, accurately predicting energy demand makes for a more efficient energy grid. With a more fluctuating energy production side, it is crucial to match consumption and supply. Overestimating energy demand causes excess energy production, which cannot be stored unless transformed to other forms, a process that introduces new costs and challenges. Underestimating energy demands can result in a overloaded electrical grid, and in worst case in blackouts.

Whilst raising awareness concerning energy demands is important, it is also critical to acknowledge the necessity of not sacrificing interior environment while striving to optimize a building's energy consumption. The modern population spends up to 90% of their time indoors, with an average of 8 hours spent at work [78]. Maintaining an ideal indoor climate is therefore critical for people's well-being, health, and productivity, all of which is essential for a productive work environment. Indoor climate is made up of a variety of physical, chemical, and biological factors [9], some of which include temperature, lighting and air quality.

As buildings are becoming increasingly more advanced in regards to integration of modern technology, the term *Smart building* has emerged. The term has become a large part of day-to-day conversations, both private and in business and refers to a building that has control systems and devices interconnected in addition to the features of a conventional building [101]. Most indoor climatic parameters in such structures are managed by highly sophisticated systems, e.g. automated ventilation systems. Such systems are often monitored and controlled with help of IoT (Internet of Things), an infrastructure technology enabling devices to collect and transfer data [51]. IoT devices can collect large volumes of data, but interpreting the data without prior knowledge and expertise in the system might be challenging. To ensure that the information is easily accessible for monitoring and analysis, it is crucial to facilitate for appropriate, efficient and secure collection and storage.

This study explores the possibility of predicting future energy demands based on performance of ventilation systems, offering useful insight and significant benefits to parties wishing to optimize their present energy consumption. By creating a digital twin of the building and its ventilation system, we aim to structure data in such a way that context, algorithms and additional information can be applied to increase the accessibility, value and application range of the original data. Specifically, the study explores the possibility of optimizing the indoor temperature, keeping it within the recommended limits, whilst still keeping the energy usage and cost to a minimum. In this project, the investigation is based on data collected from sensors located in the ventilation systems in Arkivenes Hus. Arkivenes Hus is a BREEAM-certified [12] building in class Excellent, which ensures high quality indoor climate and lower costs related to energy consumption. The building receives environmentally friendly energy from Ullandhaug Energi and is classified to Energy Class A. [7]

1.1 Aim of The Project

The aim of this project is to gather data from Arkivenes Hus, use it to optimize indoor climate comfort and study the effects this has on energy consumption. Thus, the primary research question is:

How can a control strategy for a ventilation system optimize indoor climate, and what effect does this have on required energy consumption?

Specific areas requiring research in order to achieve this are:

- How to collect and structure sensor and energy data.
- Identify best way of structuring available sensor data in a digital twin for closer monitoring of the system.
- Develop model describing relations in the ventilation system.

- How to implement a control strategy.
- Test, compare and identify the best machine learning models for prediction of energy consumption.

1.2 Outline

- Chapter 2 covers related works with focus on data structuring, optimization and prediction.
- Chapter 3 introduces all technologies used for the project.
- Chapter 4 provides an overview of how the solution is built, as well as how the different technologies are utilized in order to solve the problem. It also presents the process of data analysis, identification of control strategy and utilization of machine learning.
- Chapter 5 provides an experimental evaluation of all methods utilized. Presenting the results of different machine learning algorithms and identification of control strategies.
- Chapter 6 presents a discussion around the work and results.
- Chapter 7 concludes the work.
- Chapter 8 presents possibilities for future work.

Chapter 2

Background and Related Work

In this chapter a brief theoretical overview is provided to insure a better understanding of how the various steps of our solution are performed. This chapter summarizes the theory of digital twins, Sparse Identification of Nonlinear Dynamical Systems (SINDy), Model Predictive Control (MPC), and datasets, machine learning algorithms and evaluation methods related to energy prediction. This chapter also introduces any related work.

2.1 Digital Twin

It is commonly accepted that the twin concept dates back to technological innovations from NASA's Apollo program, completed between 1967 and 1972 [6], [94], [66]. One of the innovations used in this program was the creation of a physical twin aircraft. This enabled engineers and scientists to troubleshoot issues without imposing increased risk to the mission, something which ended up being crucial in bringing home the Apollo 13 mission crew [6].

The technical realm is currently in the era of digital transformation. Within the computer science and engineering literature, notions such as big data [100], industry 4.0 [93] and IoT [43] have become a common part of the terminology. Frequently mentioned along with these technologies, and advocated for having equally disruptive effects, is the concept of digital twins [94].

A digital twin (DT) can be defined as a "digital replica of a physical entity" [57]. An integral part of the potential of digital twins is the possibility to model a digital replica and combine it with data. This potential is reinforced by the emergence of and possibilities to incorporate machine learning, thereby bringing data, algorithms and framework together [57] [94]. This broad definition causes the digital twin literature to contain a vast amount of different approaches, ranging from manufacturing- [56] to business- processes [59] , and even digital twin models of entire cities [42]. With such diverse application areas, combined with the broad and unspecific definition, the digital twin domain takes a variety of shapes and contain numerous technologies.

The manufacturing and automotive industries are industries where digital twins have been widely utilized and integrated with success, while the building industry has yet to incorporate the full potential of digital twin at a similar scale [66]. One major reason for this is the diversity of aspects and life cycle phases to consider within a building. Some possible areas for digital twin implementation include building information modelling, facility management, monitoring, logistics, energy analysis and structural system analysis [66]. A consequence of this large application domain and the broad definition of digital twin is that there are multiple ways of implementing digital twins, depending on desired insights.

Some authors [47] define a building information model as a digital twin. BIM enables 3D modelling and information management. A 3D BIM model provides an accurate picture of how the physical building is constructed and how various aspects and systems are related to each other, e.g. electrical systems or ventilation. A large number of people, each with a different role, are involved in the construction of a new building. It is critical that relevant information flows amongst these people, and that the information is as accurate as possible; this is one of the primary reasons for using BIM in construction projects. Figure 2.1 displays a BIM illustration of Arkivenes Hus as an example of such a model. A model like this provides information about everything both inside and outside the building, such as construction materials, furniture, and ventilation systems.

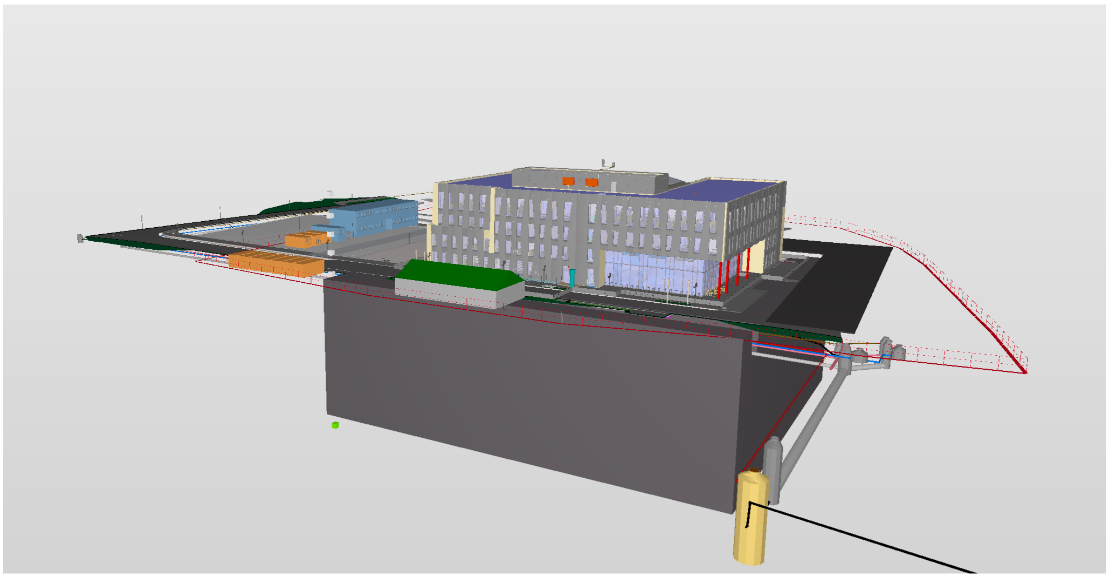


Figure 2.1: 3D BIM illustration of Arkivenes Hus created in Solibri [11], provided by Veni AS

BIM models provide useful information in the construction process. However, as already emphasized in this section, one main component of digital twins is the ability to incorporate data. BIM models lack the ability to provide an updated representation of the current state of a system based on a continuous data flow [66].

Another method of digital twin representation is knowledge graphs. This approach is defined through ontologies, which is a compilation of classes, object- and data-properties used to exhibit facts about and a semantic model of area of interest [3]. A knowledge graph is a system of connections of nodes or data points expressed as a graph. The nodes/points are representations of instances, with relevant relationships defined between the nodes. Such graphs are commonly developed around the concept of linked data [54], [3]. Linked data revolves around creating links between data (as opposed to isolated data points), so that a person or machine can explore and discover relations between entities [54]. Figure 2.2 illustrates and example of a knowledge graph. The figure depicts the fact that a room contains both a temperature sensor and a motion sensor.

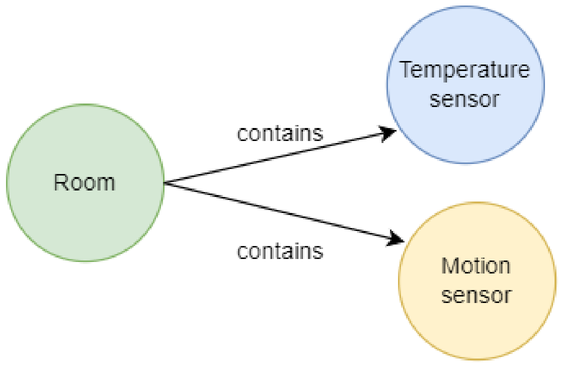


Figure 2.2: Knowledge graph example displaying the relation between a room and its sensors.

Creating such knowledge graphs and connecting sensor data to nodes has become a popular way to develop digital twins, and is now one of the main methods used by industries, something which will be covered further in Section 3.1.1.

2.2 Sparse Identification of Nonlinear Dynamical Systems

Sparse Identification of Nonlinear Dynamical Systems [13], known as SINDy, is a method used to identify governing equations for nonlinear dynamical systems. Assume a set of data-points representing some measurements $x(t) \in \mathbb{R}^n$ at various times t for a system. SINDy aims to provide a way to translate the data-points $x(t)$ into a non-linear function of $x(t)$, f, using the equation

$$\frac{d}{dt}x(t) = f(x(t)), \tag{2.1}$$

which represents a dynamical system for the measured data-points in $x(t)$. $x(t)$ is a vector on the form $x(t) = [x_1(t), x_2(t), \ldots x_n(t)]^\top$ and f describes how the system evolves with time. SINDy uses sparse regression [35], [91], [44] to identify a linear combination of basis functions that most fully describe the nature of the system. The method is built around the assumption that most dynamical systems have few relevant terms describing the system, meaning that f only has a few terms. If one imagines a nonlinear function space of high dimension, then the equations representing some physical system will be sparse in this space, given some threshold value [13].

The SIDNy method provides a way to translate data measurements into a dynamical system representing the time progression of the system. Suppose X represents measurement values of quantities $x_1, x_2, \ldots, x_n$ obtained at times $t_1, \ldots, t_n$:

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \ldots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \ldots & x_n(t_2) \\ \vdots & \vdots & & \vdots \\ x_1(t_m) & x_2(t_m) & \ldots & x_n(t_m) \end{bmatrix} \tag{2.2}$$

The derivatives of the monitored variables are represented by $\dot{X}$, either obtained by measurements or numerically approximated [74], [75]:

$$\dot{X} = \begin{bmatrix} \dot{x_1}(t_1) & \dot{x_2}(t_1) & \ldots & \dot{x_n}(t_1) \\ \dot{x_1}(t_2) & \dot{x_2}(t_2) & \ldots & \dot{x_n}(t_2) \\ \vdots & \vdots & & \vdots \\ \dot{x_1}(t_m) & \dot{x_2}(t_m) & \ldots & \dot{x_n}(t_m) \end{bmatrix} \tag{2.3}$$

A candidate matrix, denoted $\Theta(X)$, is constructed. It contains candidate non-linear functions for the columns of X:

$$\Theta(X) = \begin{bmatrix} | & | & & | \\ \theta_1(X) & \theta_2(X) & \dots & \theta_\ell(X) \\ | & | & & | \end{bmatrix} \tag{2.4}$$

SINDy aims to find a set of sparse coefficient vectors gathered in a matrix:

$$\Xi = \begin{bmatrix} | & | & & | \\ \xi_1 & \xi_2 & \dots & \xi_n \\ | & | & & | \end{bmatrix}, \tag{2.5}$$

where, as described in [5], " ξ_i provides the coefficients for a linear combination of basis functions $\theta_1(x), \theta_2(x), \dots, \theta_\ell(x)$ representing the ith component function of f: $f_i(x)$. That is to say,

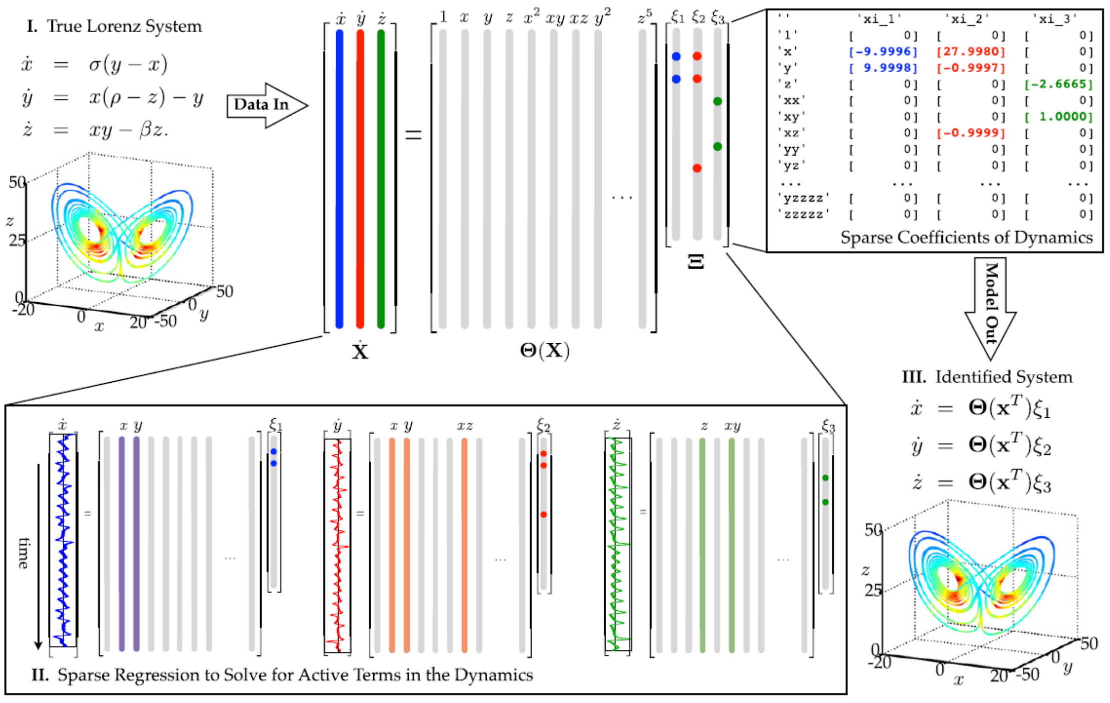
$$f_i(x) = \Theta\left(x^\top\right)\xi_i, \tag{2.6}$$

where $\Theta\left(x^\top\right)$ is understood to be a row vector consisting of symbolic functions (whereas $\Theta(X)$ is a matrix whose entries are numerical values)."

Once X, $\dot{X}$, $\Theta(X)$ and Ξ are obtained, the sparse regression problem that defines the active non-linearities is expressed by [13]:

$$\dot{X} = \Theta(X)\Xi \tag{2.7}$$

Least squares regression would yield some values from all of the columns in $\Theta(X)$. Sparse regression is therefore preferred to ensure good regression fit with as few nonlinear right hand side terms as possible in eq. 2.7. Figure 2.3 provides a schematic of the algorithm.


I. True Lorenz System
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\rho - z) - y$
$\dot{z} = xy - \beta z.$
Data In
$\dot{\mathbf{X}}$
$\boldsymbol{\Theta}(\mathbf{X})$
Ξ
Model Out
III. Identified System
$\dot{x} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_1$
$\dot{y} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_2$
$\dot{z} = \boldsymbol{\Theta}(\mathbf{x}^T)\xi_3$
time
II. Sparse Regression to Solve for Active Terms in the Dynamics

```
''          'xi_1'       'xi_2'       'xi_3'
'1'        [      0]    [      0]    [      0]
'x'        [-9.9996]    [27.9980]    [      0]
'y'        [ 9.9998]    [-0.9997]    [      0]
'z'        [      0]    [      0]    [-2.6665]
'xx'       [      0]    [      0]    [      0]
'xy'       [      0]    [      0]    [ 1.0000]
'xz'       [      0]    [-0.9999]    [      0]
'yy'       [      0]    [      0]    [      0]
'yz'       [      0]    [      0]    [      0]
...             ...          ...          ...
'yzzzz'    [      0]    [      0]    [      0]
'zzzzz'    [      0]    [      0]    [      0]
```

Sparse Coefficients of Dynamics

Figure 2.3: The SINDy algorithm applied on the Lorenz equations. Measurements of the system performance over time and it's derivative, are obtained and constructs X and $\dot{X}$, respectively. Following that, a collection of non-linear state functions, $\Theta(X)$, is built. This collection is used to determine the smallest numbers of terms required to describe $\dot{X} = \Theta(X)\Xi$. The relevant terms in the right-hand side of the dynamics are denoted by the few entries in the vectors of Ξ, which where solved for using sparse regression. From [13].

2.3 Model Predictive Control

Model Predictive Control (MPC) focuses on forecasting and optimizing the change and behavior of a dynamical system caused by changes in independent variables [77]. The objective is to minimize the error between some desired reference trajectory and predicted future output, as well as minimizing some control action required to obtain the desired output. A basic MPC structure is shown in Fig. 2.4:

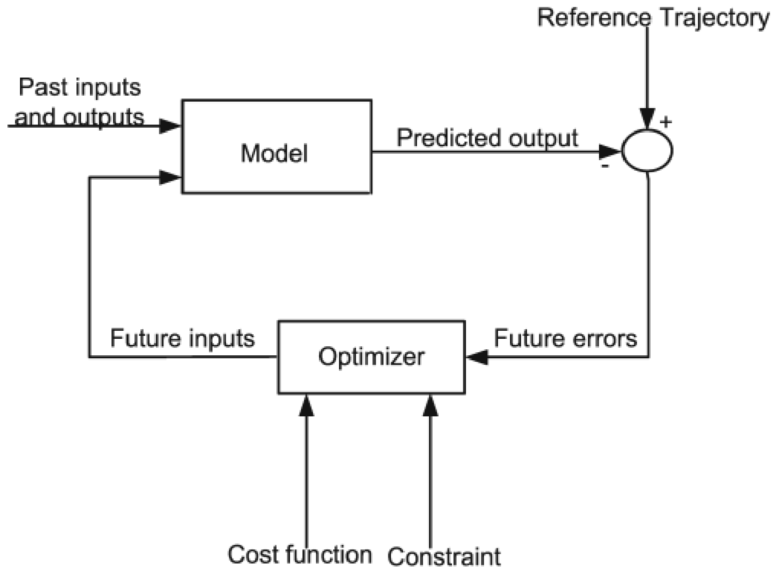


Figure 2.4: Basic structure of MPC [69].

MPC consists of three main aspects: a model used to predict the output of a process along a future time horizon, estimation of a control strategy to optimize performance and sliding time horizon strategy. There exist several types of MPCs, such as Linear MPC, Nonlinear MPC, Robust MPC, Feedback MPC, Pre-computed MPC and Decentralized MPC [69]. The main concepts for all these variations of MPC remain the same. First, the future outputs for the forecast horizon are calculated for each time, here denoted i, using a model describing the system. This step requires all previous recorded inputs and outputs of the system up to time i, as well as the current state which is treated as initial condition. Thereafter, a sequence of signals for future control is identified given the objective to optimize a specified criteria. This is typically to minimize the error between some desired trajectory of performance and predicted model performance. With the sequence of control signals identified, only the control signal for the next time step is transmitted to the system. Then, the time horizon moves one instant and the process is repeated. This means that the prediction horizon remains the same length, but slides along one time step per iteration [69], [77]. A typical optimization problem with MPC can take the form

$$\min_u f(x,u) = \sum_{i=0}^{n-1} \{(x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) + (u_i - u_{ref,i})^T P (u_i - u_{ref,i})^T\} \\ + (x_n - x_{ref,n})^T S (x_n - x_{ref,n}), \tag{2.8}$$

subject to some constraints [40]. In Eq. 2.8, differences of the state x_i compared to some reference trajectory $x_{ref,i}$ are penalized. This also holds for the control variable u_i, which is compared to some reference $u_{ref,i}$. The references are understood to be supplied to the controller by some external source [40]. Q, P and S are assumed to be symmetric.

2.4 Energy prediction

The global energy sector is currently undergoing significant changes, driven by a commonly acknowledged goal to reduce global greenhouse gas emission. This affects all sides of the energy sector. The energy production and supply side are turning their backs to fossil-driven sources such as coal and transitioning towards renewable sources like wind and solar. On the demand and consumer side, numerous improvements can be done. According to statistics, the building sector is a significant consumer of energy, accounting for 39% of the worlds overall energy consumption [87]. Building energy management and energy operations rely heavily on either physical models or data-driven models to create forecasts to improve energy efficiency. Data-driven energy models are based on data to derive a context between input (e.g. ventilation sensors) and output (e.g. energy consumption). Although physical models can incorporate a greater amount of detail about a building, data-driven models have the benefit of being easily implemented and adapted to other buildings. Due to this flexibility, data-driven energy models are gaining popularity [96]. Data-driven models for energy prediction are based on machine learning algorithms such as Linear Regression, Recurrent Neural Networks, Support Vector Machines and Extreme Gradient Boosting [20], [34]. Figure 2.5 depicts a flow chart of a common process of a machine learning process.

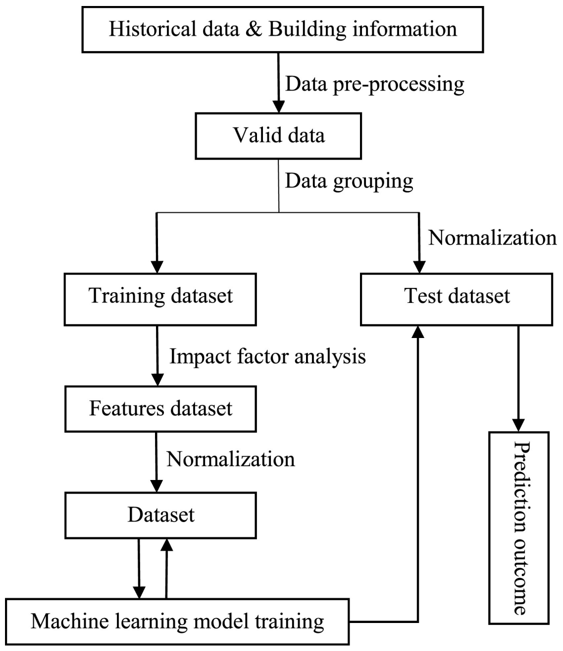


Figure 2.5: General machine learning development flow chart [20].

2.4.1 Linear Regression

Linear Regression aims to model the relation between a output response variable y given some observed input data x. The most common Linear Regression model is expressed in Eq. 2.9

$$Y = a + bX, \tag{2.9}$$

where Y represents a variable dependent of some causing variables X. b represents the regression coefficients of the causing variables X, and a the intercept [53]. Linear Regression is one of the simplest machine learning implementations as it does not require parameter tuning. The common way to fit a regression line is by the least-squares method, which yields the following equations for the parameters a and b

$$b = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n (x_i - \bar{x})^2} \tag{2.10}$$

and

$$a = \bar{y} - b\bar{x} \tag{2.11}$$

for data points $i \in [0, n]$.

2.4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are an extension of the regular feed-forward neural network and are suited to process sequential data such as time-series and use patterns to predict outputs [29], [50].

Where regular feed-forward neural networks map from a fixed-size input to a fixed-size output, Recurrent Neural Networks have the ability to handle input data of varying length and produce varying length output sequences. The network contains several successive recurrent layers that are sequentially modeled. Figure 2.6 illustrates the architecture of a Recurrent Neural Networks.

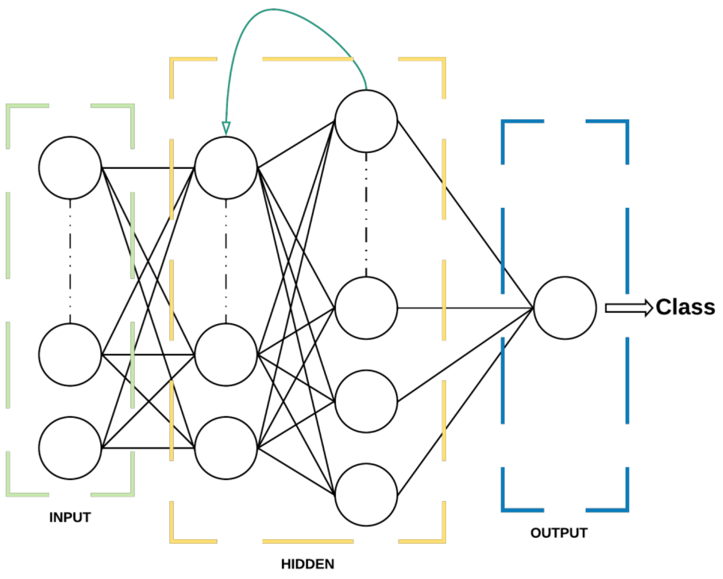


Figure 2.6: Recurrent Neural Networks.

The difference between a Recurrent Neural Networks and a regular feed-forward neural network is the introduction of loops in the hidden layers, as shown in Figure 2.6. A Recurrent Neural Networks with 3 time-steps is unpacked and illustrated in Figure 2.7.

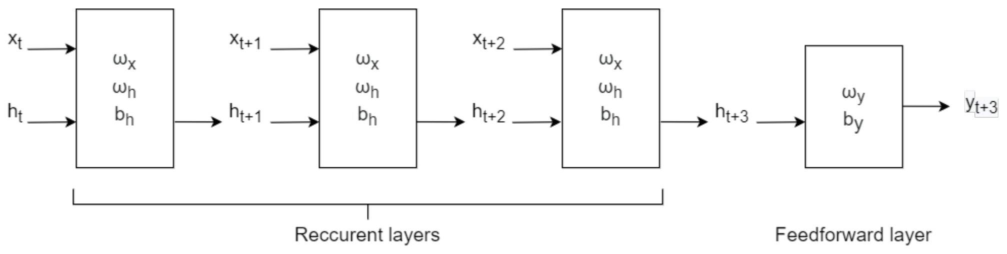


Figure 2.7: Recurrent Neural Networks, 3 time steps.

In Figure 2.7, x_t represents the input at time t. x_t can represent a d-dimensional feature vector. y_t represents the output at time t. A Recurrent Neural Networks can produce multiple outputs. In Figure 2.7, the network only generate one output at the last timestep, $t = 3$. h_t represents a m-dimensional vector storing hidden state values at time t, often called the current context. ω_x is a m-dimensional weight vector for the input in a recurrent layer. ω_h represents a $m \times m$ matrix of the weights for the hidden units. ω_y represents a m-dimensional vector of the weights related to the hidden to output units. b_h is a m-dimensional vector representing the bias at the recurrent layers and b_y is the bias related to the feed-forward layer. One rectangle in Figure 2.7 illustrates a operation taking place with a given activation function f:

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x x_t + w_h h_t + b_h) \tag{2.12}$$

At time t, the output y is found by

$$y_t = f(h_t, w_y) = f(w_y \cdot h_t + b_y), \tag{2.13}$$

where $\cdot$represents the dot-product. Common activation function choices are the Sigmoid, RELU or Tanh [4].

One shortcoming of recurrent neural networks is the vanishing gradient problem, a problem occurring when training using back-propagation related to updating the weights that are used to connect the hidden layers to themselves. If the weights are small, this will lead to the gradient vanishing. If the weights are large, it will cause the gradient to explode [15].

Long short-term memory networks, LSTMs [39], is a version of a recurrent neural network that is designed to overcome the vanishing gradient problem. A LSTM can be considered as a recurrent neural network where each hidden unit is replaced by what is known as a LSTM cell, and the addition of a connection between each cell known as the cell state.

Each LSTM cell maintains a cell state vector. At each time-step, the next LSTM cell can choose to read, write or reset the cell using a gating mechanism. A LSTM cell has three gates. A input gate controls whether the memory cell is updated. The forget gate controls if the memory cell is reset to 0. The output gate controls whether the information of the current cell state is made visible. Every gate has the Sigmoid activation function:

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \tag{2.14}$$

$$f_t = \sigma(w_f[h_{t-1}, x_t] + b_f), \tag{2.15}$$

and

$$o_t = \sigma(w_o[h_{t-1}, x_t] + b_o), \tag{2.16}$$

for the input, forget and output gate, where the w_i, w_f, w_o and b_i, b_f, b_o, represents the weights and biases, respectively.

In addition to the described gates, a vector $\bar{C}$ modifies the cell state. This vector has the Tanh activation function to ensure the distribution of gradients[65], thereby preventing the vanishing/exploding gradient problem:

$$\bar{C}_t = \tanh(w_C[h_{t-1}, x_t] + b_C) \tag{2.17}$$

A LSTM cell is shown in Figure 2.8.

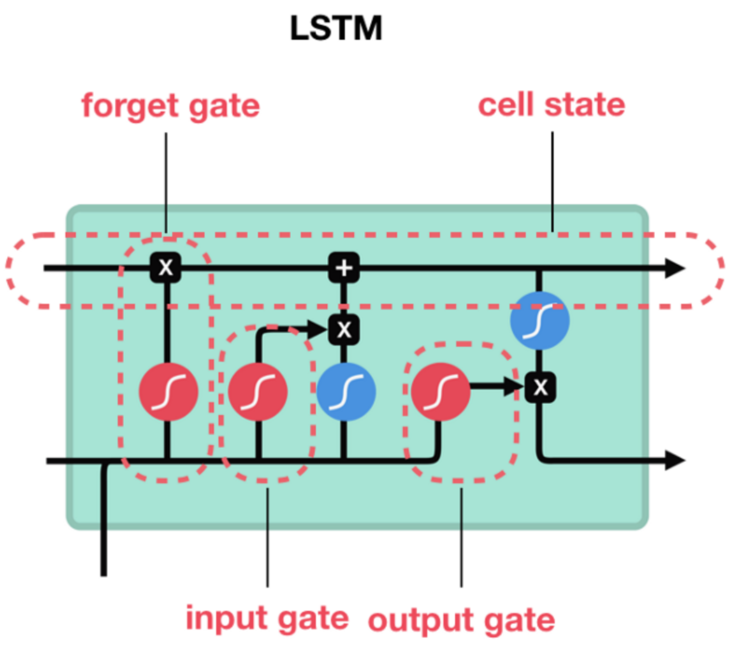


Figure 2.8: One LSTM cell [72].

LSTMs have shown to perform well when faced with the task of short-term load prediction [95].

2.4.3 Support Vector Machine

Support Vector Machine is a machine learning algorithm providing strong non-linear capabilities and support for both classification and regression. The main idea of the Support Vector Machine algorithm is to obtain a hyperplane in an N-dimensional space that distinctly classifies all data points. N is representative to the number of features present. SVM separates data in to two classes, and the data points are classified by identifying the hyperplane with the maximum margin out of multiple planes. The margin in question refers to the distance between data points of different classes. The purpose behind maximizing the distance is to provide some reinforcement in order to increase the confidence related to classification of future data points [32]. In order to maximize the margin, hinge loss is calculated. Hinge loss is a loss function defined by Eq. 2.18

$$c(x, y, f(\mathbf{x})) = \begin{cases} 0, & \text{if } y * f(x) \geq 1. \\ 1 - y * f(x), & \text{otherwise.} \end{cases} \tag{2.18}$$

Meaning that $c(x, y, f(\mathbf{x})) = (1 - y * f(x))_+$

To obtain balance between the margin maximization and the loss, a regularisation parameter is added to the loss function. The result of introducing the regularisation parameter is presented in Eq. 2.19.

$$min_w \lambda ||w||^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+ \tag{2.19}$$

The loss function is minimized by calculating gradients and updating the weights of the Eq. 2.19. The gradients are found by taking the partial derivatives with respect to the weights, the calculation is presented in Eq. 2.20.

$$\begin{aligned} \frac{\delta}{\delta w_k} \lambda ||w||^2 &= 2\lambda w_k \\ \frac{\delta}{\delta w_k} (1 - y_i \langle x_i, w \rangle)_+ &= \begin{cases} 0, & \text{if } y \langle x_i, w \rangle \geq 1. \\ -y_i x_{ik}, & \text{otherwise.} \end{cases} \end{aligned} \tag{2.20}$$

The gradients are updated prior to every classification. The way in which the gradients are updated depends on whether the data point was correctly classified or misclassified [32]. If a misclassification has occurred, the update happens according to Eq. 2.21, which include both the loss and the regularization parameter in the gradient update.

$$w = w + \alpha * (y_i * x_i - 2\lambda w) \tag{2.21}$$

When a data point has been correctly classified, the gradient update is performed according to Eq. 2.22 which only updates the gradient from the regularization parameter.

$$w = w - \alpha * (2\lambda w) \tag{2.22}$$

When it comes to logistic regression using Support Vector Regression (SVR), the main idea is to introduce a kernel function, which has the capability of mapping the input space to a hyper-dimensional space that creates a optimized hyperplane. The function is expressed in Eq. 2.23, where $f(\mathbf{x})$ denotes the prediction outputs, $\mathbf{W}$ denotes the weight factor, b is the adjustable factor and $\varphi(\mathbf{x})$ represents the mapping function used to map input space to the hyper-dimensional space [20].

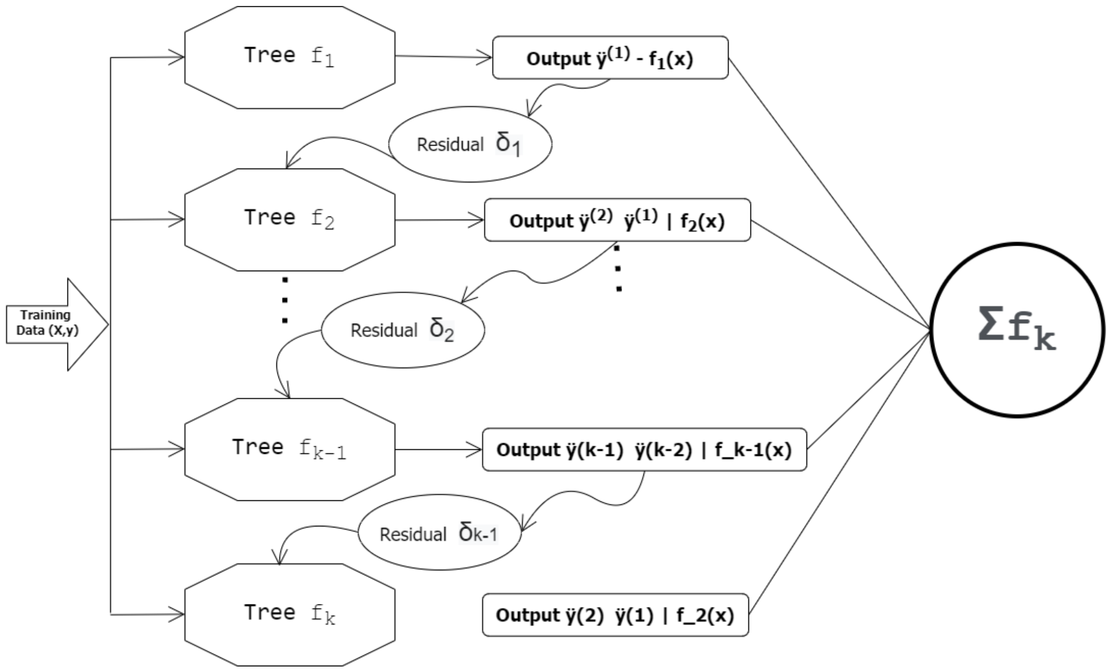
$$f(\mathbf{x}) = \mathbf{W}^T\varphi(\mathbf{x}) + b, \tag{2.23}$$

An area of research where SVM has been proved to perform at a high success rate is forecasting of energy consumption, this is due to its ability to resolve non-linear regression issues [19]. SVR has also been proved to be a promising method when it comes to forecasting aggregated loads of data from single buildings or clusters of buildings, but that the method might not be the most appropriate for forecasting electricity usage for an individual household. SVR has been reported to perform well in short-term load forecasting due to its prediction accuracy and speed in the field [21].

2.4.4 Extreme gradient boosting

Extreme gradient boosting is a type of machine learning referred to as an ensemble learning algorithm. It is a model constructed from a collection of decision tree models. The trees are added sequentially to the ensemble and fit to correct prediction errors made by prior trees. The structure of the model provides it with the capability to solve vast amounts of data-mining issues in a short period of time, with high accuracy.

The extreme gradient boost (XGBoost) algorithm is based on a gradient boosting algorithm and is one of the most reached for methods in the field of machine learning [20]. The model produces higher level prediction outcomes by managing the complexity and reducing the level of overfitting with help of the integrated regularization.



Training Data (X,y)
Tree f_1
Output $\ddot{y}^{(1)}$ - $f_1(x)$
Residual δ_1
Tree f_2
Output $\ddot{y}^{(2)}$ $\ddot{y}^{(1)}$ | $f_2(x)$
Residual δ_2
Tree f_{k-1}
Output ÿ(k-1) ÿ(k-2) | f_k-1(x)
Residual δ_{k-1}
Tree f_k
Output ÿ(2) ÿ(1) | f_2(x)
Σf_k

Figure 2.9: Extreme Gradient Boosting Algorithm

.

Extreme gradient boosting has been recognized for its performance levels and has won a large variety of machine learning competitions, making it one of the go-to algorithms for many data scientists [60]. Some of its known weaknesses is its capability to predict the future as well as making prediction for values outside the range of the training data. These weaknesses limit the XGBoost model when applied to regression problems that involve predicting continuous output. Despite the mentioned weaknesses, Extreme Gradient Boosting algorithms are know to work well for classification problems, situations with many categorical variables and prediction issues where the range of the target values present in the training set can be expected.

2.4.5 Model Evaluation

Model evaluation is the process of utilizing various evaluation metrics to get insight into strengths and weaknesses related to a model, as well as get a better understanding of the performance of the machine learning model in question. Model evaluation is an important step in machine learning, as it allows one to assess the efficiency and performance of a model from an early point in the research process. Evaluation can also be extremely helpful when choosing what model to use for the specific data or issue. When experimenting with different models, the level of accuracy/error of which the models perform at is highly interesting. In most cases, a model with as high/low accuracy/error as possible is desired. There are many evaluation methods available, reaching from statistical measurements such as confidence intervals to visual plots and confusion matrices. For the case of our research and predictive models, we will introduce Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and R^2-squared, as these are the most commonly reported measures of predictive accuracy [52].

2.4.5.1 Mean Squared Error (MSE)

The Mean Squared Error (MSE) is a evaluation method often used with regression models. It describes the distance from a regression line to a set of observed data points. The distance from a specific point to the regression line is computed and then squared. Calculating the square root of the distances handles the occurrence of negative values, leaving us with only positive measures. In addition, computation of the square root also distributed more weight to points further from the line. In order to obtain the MSE of the model in its entirety, the mean of all the distances are computed [82]. The lower the value of the MSE, the better the model is for the data in question. The mathematical equation used to compute the Mean Square Error is listed in Eq. 2.24,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.24}$$

where y_i represents the real observed value, $\hat{y}_i$ is the predicted value on the regression line and n is the number of point pairs present or residuals.

2.4.5.2 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) is an evaluation method with many similarities to MSE. It is a model often used in relation to regression problems, and utilizes the distance between the actual measurements and the predicted values in its calculations. The MAE of a model with respect to a specific test dataset is calculated according to Eq. 2.25, where y_i represents the real observed value, $\hat{y}_i$ is the predicted value on the regression line and n is the number of residuals.

$$MAE = \frac{\sum_{i=1}^{n} abs(y_i - \hat{y}_i)}{n} \tag{2.25}$$

The absolute value of the distance between the true measured value and the respected predicted value is calculated for all values, producing n prediction errors. The mean of all calculated prediction errors are added, before being divided by the number of calculated errors, leaving us with a measure of the MAE of the model [81].

2.4.5.3 Root Mean Square Error (RMSE)

Root Mean Square Error (RMSE) is an evaluation method built on MSE (2.4.5.1). RMSE is computed by taking the square root of the mean of the square of all prediction errors computed for a data set. The mathematical equation representing RMSE is presented in Eq.2.26. y_i represents the real observed value, $\hat{y}_i$ is the predicted value on the regression line and n is the number of distances computed. This function gives us a measurement of how spread out the computed residuals are, also representing how concentrated the data is around the regression line [63].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2} \tag{2.26}$$

RMSE is very commonly utilized for evaluation of prediction models , it is often used in climatology, forecasting, and regression analysis to verify experimental results and is considered an excellent general-purpose error metric for numerical predictions. RMSE is a good measure of accuracy, but only to compare prediction errors of different models or model configurations for a particular variable and not between variables, as it is scale-dependent.

2.4.5.4 R-squared

R-squared (R^2) is a statistical measure that represents the percentage of variance of a dependent variable that can be explained by an independent variable/variables in a regression model. The measure produces decimal values ranging from 0 to 1, however it is most usually expressed as the equivalent percentage ranging from 0% to 100%. R^2 gives information about how close the specific data is to the fitted regression line, generally the higher the R^2 the better the fit of the model is. R-squared of a model is measured according to the following equation;

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{2.27}$$

where y_i represents the actual value, $\hat{y}_i$ is the corresponding predicted value and $\bar{y}$ is the mean of the y values.

For instance if R^2 of a model is computed according to Eq.2.27 and result in 100%, then all observed variation can be explained by movement in the input values. Such a case is visualised graphically in Figure 2.10

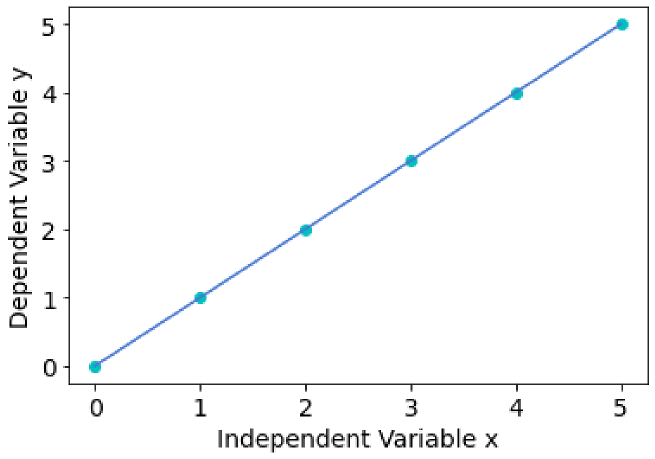


Figure 2.10: $R^2 = 1$. All variation in y can be explained by variation in x

2.5 Datasets

This section gives an overview of the historical dataset utilized for training our machine learning models.

2.5.1 Training Data

In order to train various models for prediction of energy consumption and ventilation system sensor values, a large selection of historical data is combined in a single set of training data. All historical data has been collected from Arkivenes Hus and stored by Veni AS.

The original historical dataset contains data related to a large variety of sensors stationed in the ventilation system. The values are representative for a 10-day period, from April 17th 2022 to April 27th 2022. For the present study, some of the original values are not of interest and will be removed prior to further work. This will be discussed in further detail in section 4.3. In addition to a variation of sensor values, the dataset contains a feature representing the energy provided to and consumed by the ventilation system. Both the historical sensor data and the historical energy data is represented by real-time values on a 10 minute basis, allowing for efficient preprocessing and analysis.

Chapter 3

Tools and libraries

This chapter explores all relevant technologies related to this thesis.

3.1 Azure Services

This section gives an introduction to the Azure platform and some of the many services accessible. It will give an overview of the Azure services utilized for this research, along with some of their key features.

Azure is a public digital cloud computing platform created by Microsoft, meaning that in addition to traditional cloud services which provides data storage, the platform enables users to engage in agile cloud computing [97]. The aim of the platform is to assist businesses manage what can be view as large challenges and reach any organizational goals.
This area of technology is a growing industry commonly referred to as PaaS (Platform as a Service), SaaS (Software as a service) and IaaS (Infrastructure as a service). Azure can be considered to be a provider of all these services. The Azure portal provides a wide variety of cloud services, including computation, analytics, storage and networking. The services are helpful tools when developing new technology and applications,

as well as running existing applications in the cloud.

Many of the tools used during this research are provided by Azure and they represent only a few of the countless services available in the Azure platform.

3.1.1 Azure Digital Twin

Azure Digital Twin is a platform for developing and interacting with digital models of real-world physical systems [27]. When using Azure Digital Twin, one first creates a digital twin model representing the desired system, and then one can connect real world sensor data to the model. In order to create a model in Azure Digital Twin, one needs to use what is called the Digital Twin Definition Language.

3.1.1.1 Digital Twin Definition Language

This subsection provides a brief summary of the Digital Twin Definition Language (DTDL), mainly extracted from the official documentation of the language, which is found in [28].

The Digital Twin Definition Language (DTDL) is based on JSON-LD, JSON for Linked Data, which is a variant of JSON used for large scale integration and reasoning of data [25]. A model created with DTDL must have a top level interface that provides a description of the contents of the system which is modelled. The top level interface should encapsulate the entire model. The model interface is built around four classes defined in the DTDL language: Telemetry, Property, Relationship and Component.

- Telemetry describes measurements or events, e.g. a continuous temperature measurement from a thermostat. The data stream from a telemetry field is not stored in digital twins, but has to be managed in real time as it appear in the model.

- Property represents data fields which contain information of about the state of an entity. Unlike telemetry, properties can viewed at any time as they contain storage abilities.
- Relationships enables one to define how entities in a digital twin are related to each other. Relationships can be defined with distinct semantic definitions, e.g. contains ("Building contains floor) or ("System contains sensor"). By defining relationships between entities, the digital twin model provides a graph of connected entities.
- Components enables the construction of a model interface as an aggregation of other models, if desired. E.g., one model component can be *temperatureSensor*, another can be *motionSensor*. These can be used in defining a model *room*. This requires first defining the two components as separate models, and then referencing them in the *room* model.

Several fields are used to define a model:

- `@id` is an identifier that must contain information about the model on the formal `dtmi:<domain>:<unique-model-identifier>;<model-version-number>`.
- `@type` describes the type of information.
- `@context`, used to provide context information about the JSON-LD format. This is a property of JSON-LD ensuring that the JSON document uses local meaningful names while still providing meaning elsewhere, despite beeing merged with other data [46].
- `displayName` is an optional feature which defines the name that will be shown in the model.
- `contents`, which contains the information about the data. The information is stored in an array of attributes, where each attribute is one of the four classes

(Telemetry, Property, Component, Relationship) described above. Each attribute must provide a set of features that describe the attribute, such as a `name` or a `schema`.

Since data is a significant aspect of digital twin technologies, DTDL also implements semantic type annotations, so that analytics, machine learning and other computations can reason about the semantics of data, and not just the schema of the data.

3.1.2 Azure IoT Hub

This section introduces the concepts of Azure IoT Hub along with some of its key features, providing an understanding of how the service is highly useful when working with a variation of devices and services that desire to be connected.

IoT refers to 'The Internet of Things' and is a network of different connected physical devices that exchange data with both each other and other services across the internet. Azure IoT Hub is a cloud hosted service which serves as a central message hub for communication between IoT applications and all of their connected devices. As a central message hub, Azure IoT Hub performs a variation of tasks, including message processing, triggering actions and collecting information about the device system before delivering it to an appropriate system or application. The hub is highly scalable and is capable of handling millions of connected devices along with their back end solutions in a reliable and secure way [49].

The service provides support for integration with a variation of other Azure services, allowing user to build end-to-end solutions. For instance Azure IoT Hub integrate well with services such as Azure Machine Learning and Azure Stream Analytics, which are both highly interesting services with regards to analysis and prediction issues like in this research. The simplicity related to integrating Azure IoT Hub with other Azure services allows for organizations and developers to efficiently and seamlessly extend their platform to what it needs to be, without compromising on the platforms scalability or reliability.

3.1.2.1 Devices

All smart devices has the capability to connect to an Azure IoT Hub, and for large IoT systems the number of devices can become quite vast. Azure IoT Hub has an identity registry which stores device information and modules with permission to connect to the device. This allows for quick and easy identification of the different devices. All devices and modules must have a registered entry in the identity registry before being able to connect, and authenticates itself with the IoT Hub by providing credentials stored in the registry. Devices can be authenticated through two different methods, SAS token-based authentication or X.509 certificate authentication [49].

Post authentication, the connection is secured utilizing the Transport Layer Security (TLS) standard. With a secure connection established, the devices can communicate in a variation of ways. Typically IoT devices send telemetry from their sensors to backend services in the cloud, but it is also possible for the backend services to send commands to a device [49]. Examples of telemetry received can be temperature or pressure, registered by device sensors. Besides telemetry, device properties can also be read or set from the IoT Hub. Properties can be used to send prompt notifications when an action is considered complete. Temperature is also an example of a device property as it can be updated and set on the device or read from a temperature sensor located on the device.

3.1.2.2 Data Collection

By default all data received in Azure IoT Hub is collected at a built-in endpoint, but it can also be routed on to different services for additional processing. As soon as a message route has been created, the data stops flowing to the default endpoint. In order to spread the data out to multiple subscribers, Event Grid can be utilized. Event Grid is well integrated with Azure IoT Hub and is a fully managed event service [49]. It allows the user to manage events across multiple Azure services in an easy and efficient way, and it works together with Azure IoT Hub to integrate IoT Hub Events into both Azure and non-Azure services, with low latency [48].

3.1.3 Azure Functions

Azure functions is a cloud service hosted by Microsoft, which allows users to develop and debug solutions more efficiently. It is available on-demand and provides all the continually updated resources and infrastructure necessary to run an application [33]. Azure functions runs event-triggered code-blocks in a scalable way without the user having to provide and manage infrastructure. The code only runs as a response to a collection of different events, meaning that it does not have to run continuously, but instead, only when needed. Theses features enables the user to write server-less code for handling events, with minimal overhead and cost.

3.1.4 Data Storage

This section provides a short introduction to data storage and highlights some of the key features provided by Azure SQL Database.

Storage of historical data can be of large interest and highly beneficial for a variation of industries. The method of storage utilized plays a major role in how simple the process of accessing, using and securing the data is. There is a vast amount of options when it comes to data storage, ranging from data lakes to SQL databases, which is utilized for this project.

3.1.4.1 Azure SQL Database

This subsection provides a brief summary of concepts and functionality provided by Azure SQL Database, mainly retrieved from Azure's official documentation [8].

Azure SQL Database, managed by Microsoft Azure, is one of the dominant relational database cloud service on the market today. The service is a fully managed platform as a service (PaaS) database engine, meaning it handles most of the database-management functions such as upgrading, patching, backup and monitoring the database without any user involvements required. This feature significantly reduces administrative overhead and allows for applications to reach the desired market in less time[55]. Azure SQL Database service is often chosen by developers and clients for just this reason alone, but many also opt for the service for one or more of the following reasons [62]:

- They have a desire to build a multi-tenanted software as a service type of application
- They are in possession of an existing database, and do not wish to compromise on either performance or cost.
- They wish to have the opportunity to scale their compute and storage independently.
- They wish to start using Azure as a part of their business and want to integrate azure services with their current applications.

When it comes to deploying the database, Azure offers multiple options [55];

Azure SQL Database - *Single database*, which creates a single database that relies only on its own resources and is tiered based on performance and size.

Azure SQL Database - *Elastic Pool*, which allows for resource sharing, meaning that databases share their resources with the other databases assigned to the same pool. Resource sharing allows for processing power to be allocated to only active databases and not idle ones. Although resources are shared, users are not allowed to query across databases. In fact, databases assigned to the same pool are not aware of each others existence, and can therefore not communicate and are dependent on individual security.

Azure SQL managed instance, which is highly compatible with SQL Server. It maximized its compatibility with SQL Server Databases on premise, making it highly relevant for instances looking to migrate their existing system to the cloud. Similarly to *elastic pools*, *Azure SQL managed instance* supports resource sharing among databases, but in this case the databases are not isolated in the same matter. For the user this means that they have the opportunity to transparently query across databases.

In order to query an Azure SQL Database, regardless of deployment method, queries needs to be written according to Transact-SQL (T-SQL) language [55]. T-SQL is an extension to the ANSI(American National Standards Institute) SQL standard and utilizes well known keywords such as *SELECT, FROM, WHERE, ORDER BY.* The extension adds improvements and capabilities, making T-SQL an efficient, robust, and secure language for both data access and manipulation.

3.2 PySINDy

PySINDy [26] is a Python package implementing the SINDy algorithm described in Section 2.2. The package consists of three main components, one for each term in Eq. 2.7. The first, `differentiation_method` is a component used to obtain $\dot{X}$ from X if needed. `feature_library` enables one to specify the candidate basis functions to be used in constructing $\Theta(X)$. Lastly, `optimizer` defines which method to use for solving the sparse regression problem when identifying Ξ. A SINDy object needs all three of these components. After creating an object, it needs to be fit to measurement data.

3.3 GEKKO

GEKKO [10] is a optimization library for Python, specializing in dynamic optimization and differential algebraic equation problems. It implements functionality for developing tools such as moving horizon estimation, real-time optimization and model predictive control. For MPC, it allows for specification of upper and lower boundaries, soft variables, control variables and manipulated variables, as well as great freedom in defining dynamics describing the desired system, be it linear or nonlinear.

3.4 Machine Learning

This section provides all necessary information regarding various technologies utilized in implementing, running and evaluating the different machine learning algorithms introduced in 2.4. The main resources include Scikit-learn [84] and Tensorflow [90].

3.4.1 Scikit-learn

For implementation of various regression models we can utilize Scikit-learn, a free software machine learning library for the Python programming language. It is mainly written in Python and uses NumPy [64] extensively for linear algebra and array operations. The library was first introduced in January 2010, and has been updated throughout the years. The latest release *scikit-learn 1.1.0* was released in May this year [84].

Scikit-learn is the said to be one of the most useful and robust library for working with machine learning in Python [71]. It provides its users with a variety of efficient methods which are highly useful when working with both classification issues and real value prediction issues such as ours. A library such as Scikit-learn simplifies the area of machine learning vastly and streamlines the process of both preprocessing and prediction.

3.4.2 Tensorflow

For implementation of neural networks, Tensorflow can be utilized. Tensorflow is an end-to-end platform for machine learning and provides an extensive and flexible collection of tools, libraries and community resources [90]. The platform allows for researchers and developers to create state-of-the-art machine learning models and build and deploy applications powered by machine learning.

Tensorflow facilitate for fast and easy end-to-end model building with its simple and flexible architecture, and with use high-level APIs such as *Keras* they enable immediate model iteration and simple debugging. It promises high quality production regardless of the users choice of platform, allowing for easy training and deployment in the cloud, on premise, in the browser or on-device independent of the language used[90].

3.4.2.1 Keras

Keras is a Python written API, running on top of Tensorflow, providing a Python interface for artificial neural networks. The library was developed with the intention of enabling fast experimentation. Keras, according to the library documentation [88], is all of the following:

- **Simple** – but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- **Flexible** – Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows

should be possible via a clear path that builds upon what you’ve already learned.

- **Powerful** – Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

Chapter 4

Method and Approach

This chapter provides all necessary insight into the approach taken and methods utilized in order to perform our research. It presents a step-by-step implementation of an Azure digital twin representing Arkivenes Hus, as well as the steps taken to populate the model with real-life data from sensors located in the building. It provides information regarding methods utilized in preprocessing sensor and energy data, in order to prepare it for any further work. Tools used for analysis, system identification and optimization are explained along with all the different machine learning algorithms used to predict the upcoming energy consumption.

4.1 Architecture

This section provides an overview of the system's architecture. A more detailed presentation of the proposed architecture of the subsystem that performs optimization and prediction is also given.

4.1.1 Overall System

To conduct the study in question, multiple steps are required, all of which are interconnected in some way. Figure 4.1 presents the end-to-end solution, with all actions taken.

Initially data from ventilation system sensors placed in Arkivenes Hus is collected and accessed through an Azure IoT Hub before being ingested into an Azure Digital Twin and stored in an Azure SQL Database via an Azure Function. Simultaneously, historical sensor and energy data is collected from a separate source. This data is then preprocessed and analysed before optimization is performed. Machine learning is then applied to the optimized data and prediction of energy consumption is performed.

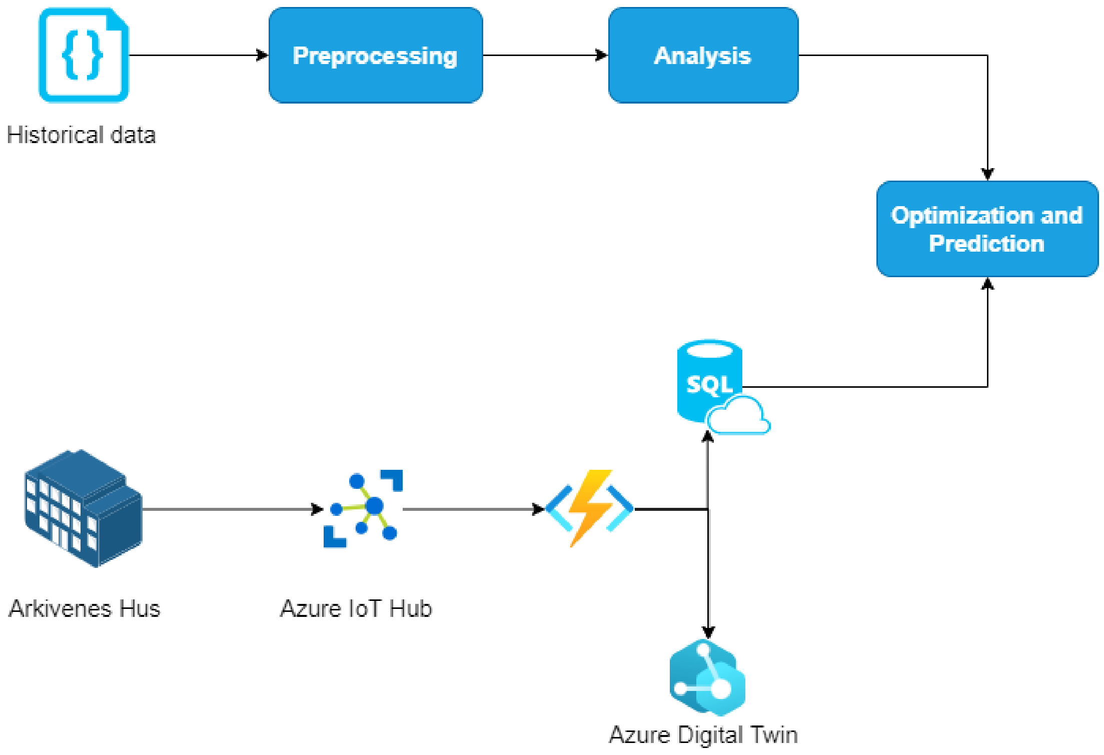


Figure 4.1: Proposed System Architecture

4.1.2 Optimization and Prediction

In this project, we propose a data-driven architecture to optimize and predict the control of one of the ventilation systems in Arkivenes hus. The architecture is illustrated in Fig. 4.2.

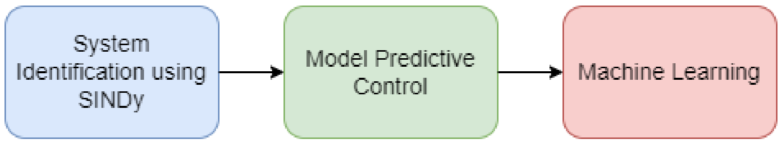


Figure 4.2: Proposed architecture for optimization and prediction
.

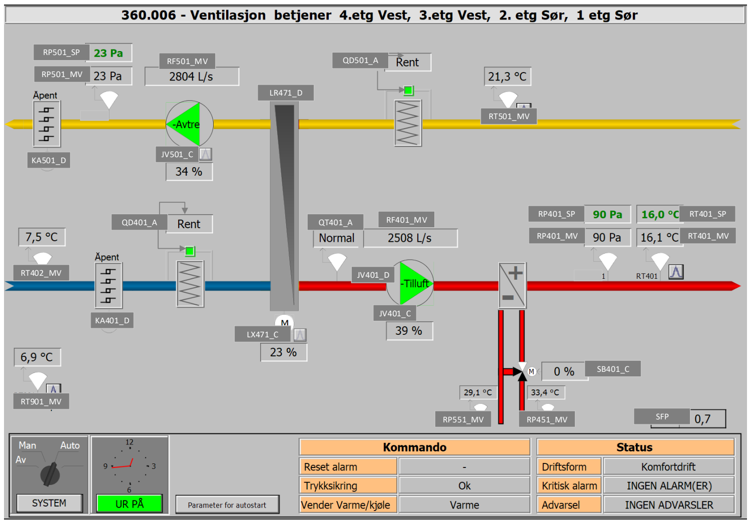
The first step of this proposed architecture is system identification using SINDy. This step obtains governing equations of relations on the data. Using this, model predictive control will be applied to determine the optimal control strategy given some objective. The output values of the control strategy will be utilized to predict the energy consumption of the ventilation system. The work related to this architecture is presented in Sections 4.4, 4.5 and 4.6.

4.2 Data Model

This section gives insight into all aspects related to the implementation of an Azure Digital Twin representing Arkivenes Hus.

4.2.1 Identification of Entities

In order to implement a realistic model, a substantial overview of all entities present in the building is required. Access to floor plans, system drawings and an extensive 3D model of the building makes the process of identifying the different entities and mapping their relations, highly effective. The floor plans provide an initial overview of how many spaces there are, what types of spaces there are, and where the different spaces are situated in the building. As for most larger buildings, Arkivenes Hus contains multiple systems, both electrical and ventilation. For the purposes of this project, we are only interested in the ventilation systems.



Kommando	
Reset alarm	-
Trykksikring	Ok
Vender Varme/kjøle	Varme

Status	
Driftsform	Komfortdrift
Kritisk alarm	INGEN ALARM(ER)
Advarsel	INGEN ADVARSLER

Figure 4.3: System drawing of ventilation systems in Arkivenes Hus. Made available by Veni AS.

Figure 4.3 presents a system drawing of a ventilation system in Arkivenes Hus, ventilation system 360.006 to be specific. As can be seen in Figure 4.3, the system drawings gives an overview of what elements and sensors are present in each individual ventilation system, which is of high interest when it comes to mapping additional information to the various sensor values currently available. Based on the information provided by these documents, we are able to identify the entities needed to produce a realistic model, these entities are presented in Appendix A.

In addition to an overview of internal entities, the system drawings also provide a brief explanation of where the system is located, in form of floor and cardinal direction. Overall the floor plans and system drawing provide little specific information on the location of the various ventilation systems in relation to the rooms of the building. In order to produce an accurate and detailed digital twin, in terms of relationships between entities, we would greatly benefit from knowing which rooms or zones each ventilation system serves. The 3D models provides us with the exact desired information. As seen is Figure 4.4 the digital 3D model visualizes the exact positioning of the ventilation systems, as well as a clear separation between the systems serving the floor.

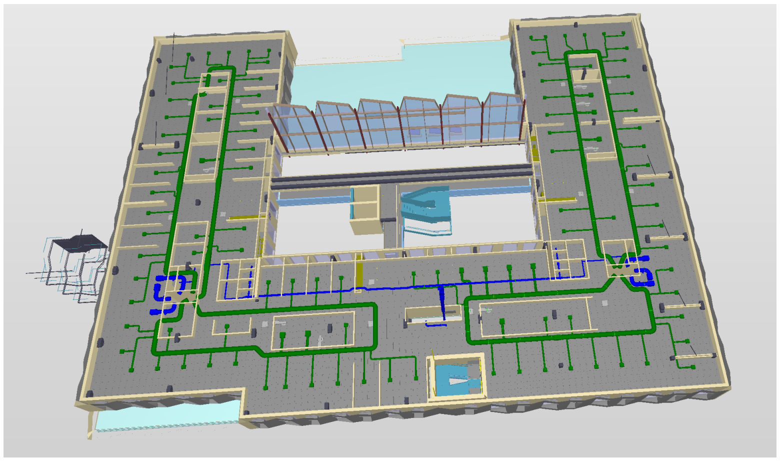


Figure 4.4: 3D model of Floor 3 with corresponding ventilation systems in Arkivenes Hus

Entities are mapped together according to the relationships stated in 4.1.

Entity	**Relationship**	**Target**
Building	contains_floor	Floor
Floor	contains_room	Room
Floor	contains_room	Archive
Floor	contains_room	Technical Room
Floor	contains	Shaft
Ventilation system	serves	Room
Ventilation system	contains_sensor	Temperature Sensor
Ventilation system	contains_sensor	SFP Sensor
Ventilation system	contains_sensor	Pressure Sensor
Ventilation system	contains_sensor	Relative Performance Sensor
Ventilation system	contains_sensor	Humidity Sensor
Ventilation system	contains_sensor	Filter Sensor
Ventilation system	contains_sensor	Damper Sensor
Ventilation system	contains_sensor	Battery Sensor
Ventilation system	contains_sensor	Airflow Sensor

Table 4.1: Mapping of relationships between entities

4.2.2 Implementation of Entity Models

For each of the entities defined in A.1 a entity model is implemented, containing their respected properties and relationships. The models are implemented according to the Digital Twin Definition Language (DTDL), and stored as separate JSON files.

Listing 4.1: Example of model implementation - Room entity

```
{
    "@id": "dtmi:com:ah:room;1",
    "@context": "dtmi:dtdl:context;2",
    "@type": "Interface",
    "displayName": "room",
    "contents": [
        {
            "@type": "Relationship",
            "name": "contains_sensor",
            "displayName": "contains sensor"
        },
        {
            "@type": "Telemetry",
            "name": "Temperature",
            "schema": "double"
        },
        {
            "@type": "Property",
            "name": "Area",
            "schema": "double"
        }
    ]
}
```

Listing 4.1 presents an example of how a model is implemented for use in our digital twin. The model is identified by a model ID, *"@id"*, which follows a "dtmi:com:ah:{*entity*}:{*specification*};1" naming convention. All relationships and properties defined in A.1 are implemented according to the example. All the entity models combined is what defines the structure of the digital twin, and all instances added to the twin has to be represented in the models.

4.2.3 Implementation of Graph

In order to produce an actual digital twin, a graph containing all entities and existing relations has to be created. This implantation is performed in Excel, before the graph is uploaded to Azure Digital Twin where the graph is read and visualized. It is crucial that the Excel file is structured correctly in order for it to be imported and read correctly by Azure's Digital Twin service. As seen in Table 4.2 the Excel document needs to be structured in 5 columns; *ModelID*, *ID* - unique instance ID, *Relationship(From)* - owner of the relationship, *Relationship Name* and *Init Data*. In addition to containing these 5 fields, all instances must have its own unique ID and must be defined prior to being listed as a Relationship owner.

The structure is well displayed in the table, as instance ***"ah"*** is the owner of the ***"contains_floor"*** relationship and must therefore be defined prior to being used in the *Relationship(From)* column for the ***ah_floor_02*** instance. For instances which occur in multiple relationships, these are inserted below the entry where the instance is first defined with its unique ID.

ModelID	ID(must be unique)	Relationship(From)	Relationship Name	Init Data
dtmi:com:ah:building;1	ah			
dtmi:com:ah:ventilation_system;1	ah_ventilation_360.002			
dtmi:com:ah:floor;1	ah_floor_02	ah	contains_floor	
dtmi:com:ah:room:archive;1	ah_room_archive_U3003	ah_floor_U3	contains_room	{"Area": 681.3}
	ah_room_archive_U3003	ah_ventilation_360.002	serves	

Table 4.2: Digital Twin Graph - Excel structure

4.2.4 Populating the Model

The primary objective of the Digital Twin is to improve access to live sensor data while also offering useful insight into the relationships between the various entities located in the building. The model is populated with live data from the physical sensors in the building, this data is sent to and accessed through an Azure IoT Hub before being passed to the Digital Twin through an Azure Function.

4.2.4.1 Azure IoT Hub

Real-life data is accessed through our Azure IoT Hub. In Azure IoT Hub several unique devices are created, representing each of the ventilation systems serving the building. The real-life data from the sensors in the building is sent as one package per unique ventilation system, containing the data from all the internal sensors. These packages are directed to their respected IoT Device, which is present in the device registry.

The IoT Hub receives data stored as objects containing information related to identifying the sensor, a sample object is provided in listing 4.2. The object provides information related to what system in which the sensor is located, as well as the name/id of the sensor. In addition to identification properties, the object also contains the most current sensor value, along with the unit of the value. The data is sent to the hub with a 1 minute frequency, allowing the digital twin to be updated at the same frequency.

Listing 4.2: Sensor object

```
{
"point": "RT501",
"value": 22.7,
"unit": "°C",
"system": "A360_006"
}
```

4.2.4.2 Azure Function

For distributing the data to our Azure Digital Twin an Azure Function is created. The main purpose of the function is to update the digital twin with the newest data collected from the sensors. In order to perform this task, an *EventGridTrigger* that runs each time a new event is registered, is created. For each event, the data received is converted to a JSON object in order to retrieve the data needed to update the digital twin. The data is received as a list of sensor objects, consisting of a sensor id (”point”), the most current sensor value (”value”), the unit of the value (”unit”) and a system id which identifies what system the sensor is located in (”system”).

By retrieving the system id we can control that all updates occur on the correct system in the Digital Twin. The digital twins of the individual sensors are updated by accessing and matching the sensor identifier with the one stated in the digital twin. If a match is found, then the previous twin value is updated with the current real-life value. The update is performed by initializing an instance of a *DigitalTwinClient* from the *Azure.DigitalTwin.Core library.* We can then call the *UpdateDigitalTwinAsync* function on the client, and passing it the device ID along with the updated values.

4.2.5 Data Storage

The sensor data received in Azure IoT Hub is valuable for future work related to analysis and machine learning. Thus, in addition to populating the model, the data is also ingested into and stored in an Azure SQL Database. The data is injected into the database through the Azure function used to populate our Digital Twin. After an event is received and distributed to the accurate instances in the digital twin, the data is ingested in the database using the following code:

Listing 4.3: Azure function executed for ingesting data into Azure SQL Database

```
1   var str = Environment.GetEnvironmentVariable("sqldb_connection");
2   using SqlConnection conn = new SqlConnection(str);
3   conn.Open();
4   var text = $"INSERT INTO [dbo].[ah_sensors] ([System_id], [Sensor_id], [Sensor_val], [Unit]) VALUES ({device["system"]}, {device["point"]},{device["value"]},{device["unit"]});";
5   using SqlCommand cmd = new SqlCommand(text, conn);
6   var rows = cmd.ExecuteNonQueryAsync();
```

Listing 4.3 presents how a connection with the database is established from the Azure function by retrieving the environment variable configured in Azure Portal[45][17]. This variable is then used to establish a connection with the SQL database through a *SqlConnector()* instance. We then define a insertion query, shown on line 4 in the listing. We then query the database using a *SqlCommand()* instance, which is executed with our defined insertion query. The query is executed by calling the *ExecuteNonQueryAsync()* function on the initiated *SqlCommand()* instance.

4.3 Data Preprocessing and Analysis

This section describes the step-by-step preprocessing of the historical data conducted in Python using Pandas [70]. We go through how the data is initially retrieved and stored is such a way that we have a clean and informative dataset for use in further work. It also includes a basic analysis of the data, which was done alongside the preprocessing to facilitate preprocessing decisions and further work.

4.3.1 Retrieving and Creating Dataset

All data used in this research has been collected and stored by Veni AS, through sensors located in Arkivenes Hus. As mentioned in section 2.5, the data present in the dataset was collected at 10 minute rate during the 10-day period from April 17th 2022 til April 27th 2022. Data for each of the different sensors, as well as the energy consumption measurements, are stored in separate files. Prior to utilizing this data in any further work, all collected data is combined in one common data set and stored in a Pandas *DataFrame*.

4.3.1.1 Timestamp Formatting

In order to structure the data set in the most efficient and informative way possible, all data points are matched by their respected timestamps. Matching the timestamps require a common format for all timestamps across all the different sensor data sets. This is done in Python using a variation of string operators. Originally the timestamps are represented on four different formats; '*dd-MM-yy hh:mm:ss AM/PM CEST*', '*dd-MM-yy hh:mm:ss AM/PM CET*','*dd-MM-yy hh:mm:ss CEST*', '*dd-MM-yy hh:mm:ss CET*'. After formatting all timestamps, they are are all presented by a common format; '*dd-MM-yy hh:mm AM/PM*'.

Post formatting, the timestamp is of type string. In order to prepare and improve the data set for use in analysis and visualisation, the string is converted to a datetime object. This is done using the *strptime()* function from the *Datetime* module in Python.

4.3.2 Correlation Analysis

In order to determine what features are important and valuable in relation to energy consumption, we conduct a correlation analysis. A correlation analysis is a statistical method utilized to obtain a measurement of the strength of the linear relationship between two variables as well as their directional association [23]. If a correlation analysis results in a high correlation, a strong relationship between the values can be assumed, whilst a low correlation points to a weak relation. A value of correlation can be both positive and negative. A positive correlation is the result of two variables moving in the same direction , meaning that an increase in one of the variables leads to an increase in the other. The opposite occurs when a negative correlation is observed, meaning that a decrease in one variable leads to an increase in the other, making the variables move in different directions [22].

Our analysis was performed using tools from the Pandas library, specifically the *corr()* function. The function calculates the correlation between each of the variables, and returns the result as a matrix of size [*number of features x number of features*]. The result of the correlation analysis performed on the original data set is presented in its entirety in Appendix C. The analysis shows a large variation in strength of relationship between the different variables, making the process of feature selection much more efficient and valuable. Table 4.3 presents an excerpt of the analysis results, representing the correlation between the final features selected.

	Value(kW)	JV401	RT501
Value(kW)	1.00	0.98	0.71
JV401	0.98	1.00	0.71
RT501	0.71	0.71	1.00

Table 4.3: Result of correlation analysis between final features

4.3.3 Feature Selection

As previously mentioned, some of the original features are to be considered irrelevant for the purpose of our research and will therefore be removed. Based on the correlation analysis, it becomes clear that some specific features have a substantially greater impact on the systems energy demand (*Value (kW)*) that others. From an overall system perspective, the features *JV401* and *RT501* are the most interesting to study further. These features are highly relevant when speaking of indoor climate, as they directly affect or represent the temperature of the building.

Table 4.4, provides a brief explanation of what the final features represent, as well as their intended use in future machine learning models.

Feature	**Description**	
JV401	Relative performance of ventilation fan	Input
Value(kW)	Energy submitted to/consumed by the system in kW	Output
RT501	Measured indoor temperature(C) in Arkivenes Hus	Output

Table 4.4: Description of selected features

4.3.4 Trends and Patterns

Following preprocessing, we conduct an analysis to explore the selected data for any trends or patterns. Figure 4.5 depicts the temperature change inside Arkivenes Hus on a random weekday. The fact that the day under consideration is a weekday is significant since factors such as occupancy can be anticipated to influence indoor temperature. As can be seen, the temperature drops at night before rising in the morning around the time of sunrise. This observation is to be expected, given outdoor temperature and sun position is known to have a significant impact on indoor temperature and climate. This insinuates that the temperature drops at night as a result of lower outdoor temperatures and an occupancy which can be assumed to be close to 0, before rising at sunrise when temperatures tend to increase. During the active work hours of the day, the temperature rises steadily until lowering towards the end of the day. The temperature rise is expected as the building's occupancy level, as well as the outdoor temperature, are at their highest during those hours.

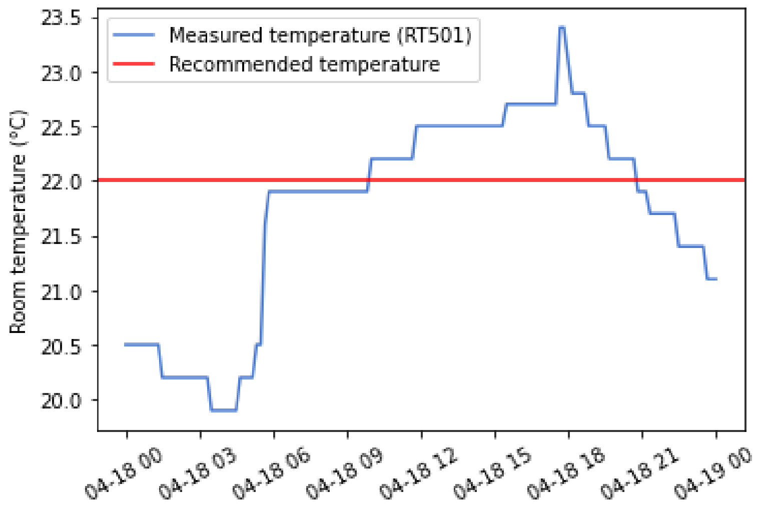


Figure 4.5: Indoor temperature in Arkivenes Hus.

Arbeidstilsynet recommends that when conducting light indoor labor, the interior temperature at a workplace not exceed 22 degrees Celsius or fall below 19 degrees Celsius. This recommendation is based on the significance of maintaining a balanced interior climate, as working in overly hot or too cold temperatures can be uncomfortable and make one less productive [89]. Observing the y-axis reveals that the measured temperature of the building is fairly high during work hours, exceeding the recommended temperature of 22 degrees for the majority of the day.

As previously stated, Arkivenes Hus is served by several ventilation systems, each of which serves a distinct portion of the structure. The ventilation systems are in charge of regulating and maintaining the ideal temperature in the building. The effect the system has is mainly controlled by the internal fan, which controls the amount of air supplied to the building. Sensor JV401 measures the relative performance of the fan, the data collected by the sensor over a 24-hour period is visualised in Figure. 4.6

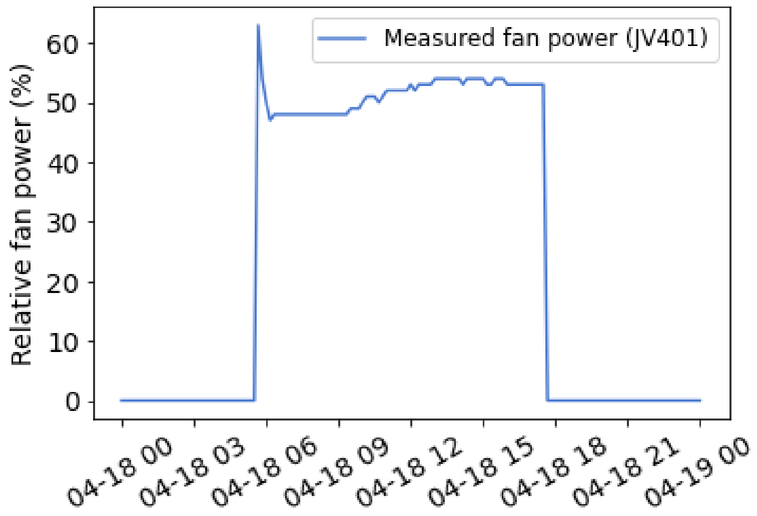


Figure 4.6: Relative performance of ventilation system serving Arkivenes Hus.

Based on the data presented, it becomes clear that the ventilation system is shut down during the night, resulting in a relative fan power of 0 %. The system is then turned back on right around sunrise, before people return to work. The system is set to perform at about 60% of possible power, causing a spike in the graph. It is then kept at consistent level though out the day of around 50% of maximum capacity, slightly increasing during the busiest hours of the day. The system is shut down again at the end of the day, around 18:00.

The ventilation system serving a large building, such as Arkivenes Hus, requires energy to operate. By visualizing the energy consumed by the ventilation system, we receive significant insight into how system activity influences energy consumption. Figure 4.7 displays the system's reported kW consumption every 10 minutes. The observed pattern corresponds to the activity recorded in the ventilation system and is nearly equivalent to the pattern found in the relative performance of the ventilation fan. This implies a close association between the two events, which is also corroborated by the prior correlation analysis.

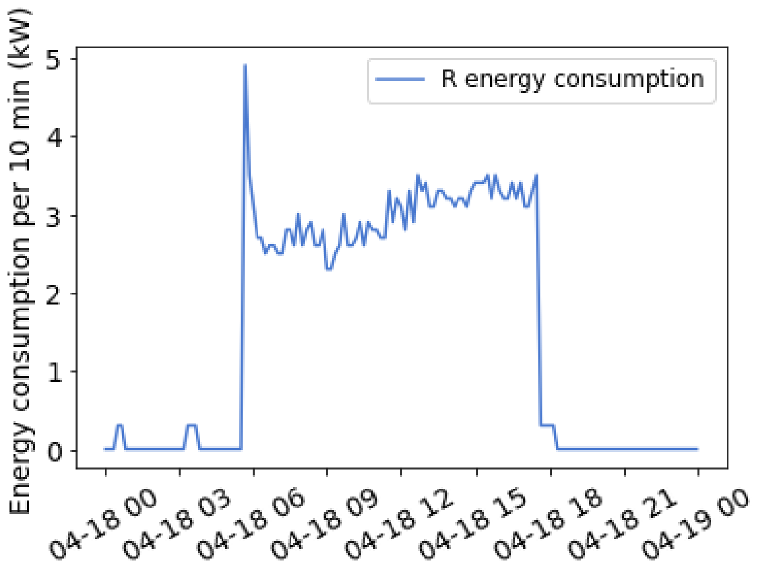


Figure 4.7: Energy consumption for ventilation system serving Arkivenes Hus

Aside from the strong association between energy consumption and ventilation system performance, the patterns seen in ventilation activity also match temperature development rather well, demonstrating that the ventilation system has a considerable impact on indoor temperature as well. This leads us to assume that with the ideal control system, interior climatic requirements can be met, resulting in a much more optimal work environment, while possibly still keeping energy consumption to a minimum. This is the overarching motivation for the research carried out throughout this study.

Appendix B contains illustrations similar to the ones presented in this section, for a larger selection of days. The visualizations provided support the analysis performed and present a similar outcome.

4.4 Sparse Identification of Nonlinear Dynamical Systems

4.4.1 Initial Model Identification

For the data utilized in this project, no information about derivatives are known. Therefore, we initialize a PySINDy model with ability to calculate the finite difference derivatives [31], thereby obtaining $\dot{X}$. When specifying the model's feature library, meaning the library of candidate terms $\Theta(X)$, we implement a polynomial library, which is the standard feature library to implement when working with SINDy [67]. We implement the library such that it contains up to 4th degree polynomials. We use the Sequentially Thresholded Least Squares algorithm (STLSQ) [76] as optimizer for the regression. The threshold is set to 0.01, representing the minimum value for a coefficient in the weigh vector [76]. Any values below this threshold will be set to zero. We also implement the algorithm with normalization of the X columns by dividing with the L2 norm. We set the max iteration parameter to 30. The resulting mathematical models and simulation results obtained using this initial model from the work described in this Section is described in Section 5.2.1.

4.4.2 Smoothed Model

As described in Section 5.2.1, the initial model obtained from SINDy does not deliver desired results. This is further discussed in Secton 6.1.1. As an adjustment to this issue, we try to smooth the data used to fit the model. We therefore implement a Savitzky-Golay filter [83], which has been described to ”reduce noise while maintain the shape and height of waveform peaks” [83]. Fig 4.8 below illustrates the initial (blue), and smoothed data (green) used to fit the SINDy model.

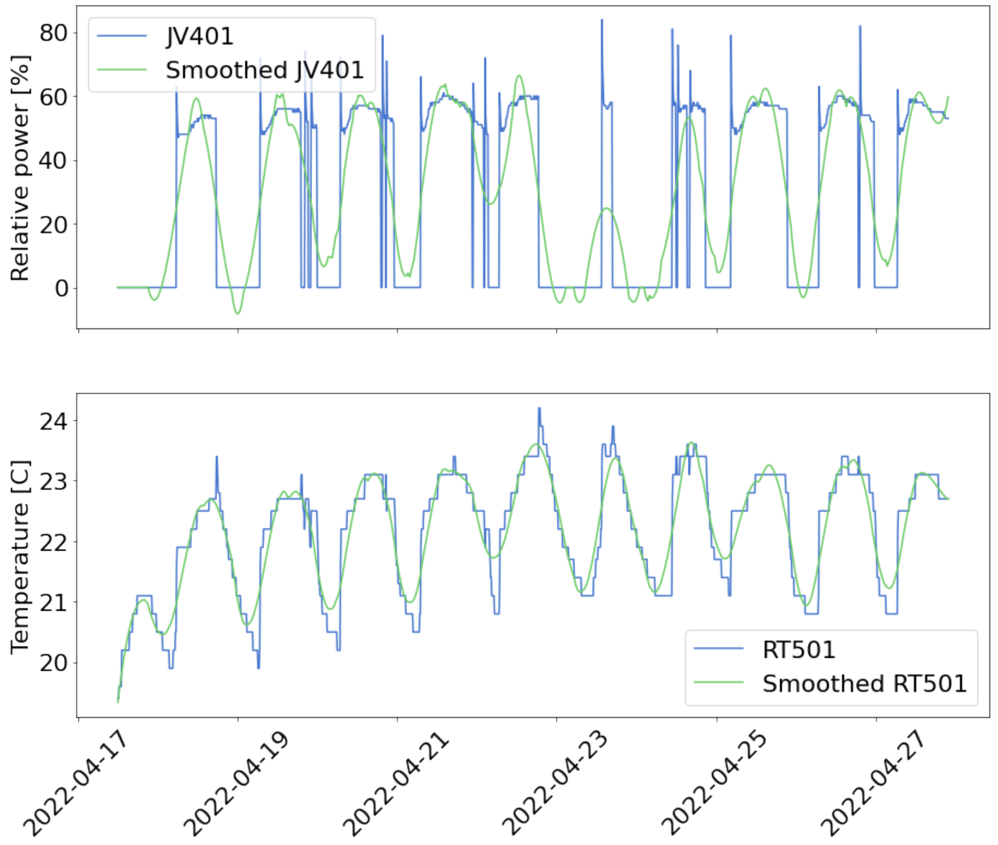


Figure 4.8: Original and smoothed data used to fit SINDy model.

4.5 Model Predictive Control

As model predictive control has been proved suitable for ventilation systems [2], [80], [14], we choose to implement this as control strategy. We formulate a nonlinear model predictive control problem using the obtained system equation for registered room temperature given activation in the ventilation fan as obtained by SINDy. We define a manipulated variable u to be the fan power, with a lower boundary of 0 and a upper boundary of 100, representing the relative fan power, and specify that this variable is to be included in the optimization. This is done by specifying `GEKKO().MV().STATUS() = 1` [92]. The room temperature measured by RT501 is defined as a state variable T, as it is changing over the time horizon. We define upper and lower constraints for temperature, where the temperature must be within 17.5-18.5 °C during the night and afternoon, and between 21-22 °C during the day. This means that the controlled variable reference seeks to follow this trajectory, and that errors are calculated using the squared error from this trajectory. Specifically, the problem is formulated as:

$$\min_{u} \quad \frac{\partial}{\partial t}T = 2.05 + 0.029u - 0.003Tu - 0.001T^2 \tag{4.1}$$

subject to the constraints

$$T_L \leq T_i \leq T_U \tag{4.2}$$

and

$$0 \leq u \leq 100. \tag{4.3}$$

T_L and T_H represents arrays of lower and upper temperature limits throughout the day, respectively, and T_i represent the current time step. Note that the term on the right hand side of Eq. 4.1 is obtained from the work described in Section 5.2.

4.6 Machine Learning Algorithms

In order to properly predict the systems energy consumption and desired sensor values, we implement a variation of machine learning algorithms fitted on the preprocessed historical dataset introduced and discussed in sections 2.5 and 4.3. The algorithms implemented are Linear Regression, Support Vector Regression, Extreme Gradient Boosting and LSTM. Implementation is conducted using tools from Python's *Scikit-learn* library and *XGBoost* [98], as well as modules from the *Tensorflow* and *Keras* libraries.

4.6.1 Linear Regression

Linear Regression is perhaps one of the simplest or well understood machine learning algorithms on the market. In addition to being quite straight forward in terms of implementation, the model also manages multiple output values, such as ours, without any additional configuration. For the implementation of the linear regressor, an instance of the *LinearRegression()* class from *Scikit-learn* is initialized with only its default parameters. The class fits a linear model with the purpose of minimizing the total sum of squares between observed targets present in the dataset and the targets predicted through linear approximation. The instance is fitted on training data representing our input values, X_train, and output values, y_train. Eqs 4.4 and 4.5 display an example of the structure of the input and output values in question.

$$Input \longrightarrow [48] \tag{4.4}$$

$$Output \longrightarrow [3.5, 23.1] \tag{4.5}$$

The training data is provided to the *fit(X_train,y_train)* method from the *LinearRegression()* class. Finally, a set of testing input data, X_test, is fed to a *predict(X_test)* method from the same class and a corresponding set of output vectors are predicted and returned.

4.6.2 Support Vector Regression

Unlike Linear Regression, Support Vector Regressors are originally intended for single value prediction. This introduces the need for an additional step in the implementation process. The implementation starts of with initializing an instance of the *SVR()* class from the *Scikit-learn* library. The *SVR()* class is a supervised learning algorithm that is used to predict discrete values. The basic idea behind the Support Vector Regression

(SVR) is to find the best fit line. The best fit line is the hyperplane which contains the highest number of points. Initially an instance of the class is initialized with most of its default values, with the exception of $gamma$ and C which is assigned according to the following values; $gamma = 0.1$ and $C = 1e3$. After an SVR is created, it is passed to an instance of the *MultiOutputRegressor()* class from *Scikit-learn*. The class allows us to fit one regressor per target, which in our case equals two. This is a quite simple strategy for extending regressors which originally only support single-output regression. After extending the original regressor to a multi-output regressor, the model is fitted on the available training data, illustrated in Eqs. 4.4 and 4.5. As for the Linear Regression model, the final step consist of passing a set of testing input data to a *predict(X_test)* method which then return a set of predicted vectors.

In addition to the initial implementation, we implement multiple other SVR models with variation in kernels used. This is done in an attempt to optimize and find the most appropriate model. The different kernels utilized are gaussian RBF, polynomial and linear Kernel. The implementation is identical to the initial approach, with the exception of a specification of the *kernel* parameter.

4.6.3 Extreme Gradient Boosting

Extreme Gradient Boosting is the last regressor implemented for modeling our data. Similarly to the two other regressors, tools from existing libraries are utilized. For the implementation of the Extreme Gradient Boosting regressor an instance of the XGBRegressor() class from the *XGBoost* library [98] has been created. Similarly to Linear Regression, XGB is able to handle multi-output regression, thus, implementation of the MultiOutputRegressor() from Scikit-learn is not necessary. The model is then fitted on our training data in the same matter as the other regression algorithms, before a prediction is preformed on the test data leaving us with a set of predicted output values.

4.6.4 LSTM

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Networks and is the only neural network implemented and tested during this research. The implementation of LSTM utilizes classes and methods from Python's *Keras* library. Initially an instance of the *Sequential()* class from Keras is created. Instances of this class groups together a linear stack of different layers into what is known as a *tf.keras.Model* object. Then a variation of layers are added to the Sequential model, starting with an LSTM layer. The layer is created using the *layers.LSTM* class, and it accepts a variation of parameters upon creation. Parameters that have been specified for this layer are presented in table 4.5.

Layer	Parameters
LSTM	Units: 64 input_shape: (1, 1) activation: Relu

Table 4.5: Specified parameters for LSTM layer

After specifying all necessary layers the model is compiled with *Mean Squared Error (MSE)* as loss function and an *"Adam"* optimizer. Finally, the model is fitted and trained on our training data. The model is configured to train according to the following parameters; $Epochs = 100$, $batchsize = 10$.

Chapter 5

Experimental Evaluation

This chapter provides the results obtained by applying the different methods, described above, in our study case. It presents the estimated equations obtained from our SINDy implementations, along with the suggested control strategy produced by MPC. In addition, an overview of the results achieved by the different machine learning algorithms is provided. Lastly, the final results achieved by the overall system is presented.

5.1 Digital Twin

Figure 5.1 depicts the final Digital Twin of Arkivenes Hus, presenting the structure of the building along with all systems and sensors present. Every node shown in the Figure represents an instance of an entity model, all of which are defined according to the structure explained in section 4.2.2. Nodes visualised with the same color are instances of the same entity model. The lines connecting the nodes are generated by the graph, which was implemented with relations found in Arkivenes Hus, following the excel structure presented in section 4.2.3

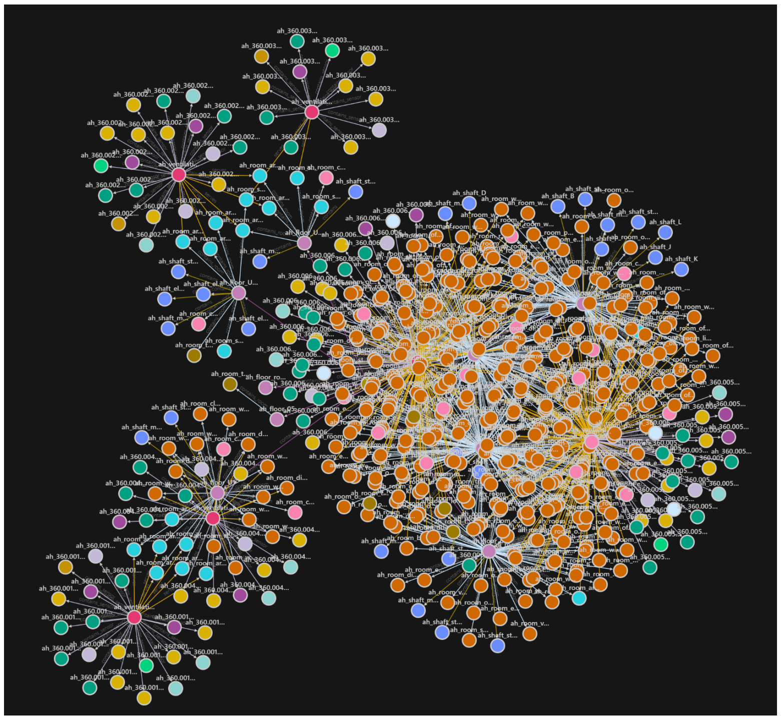
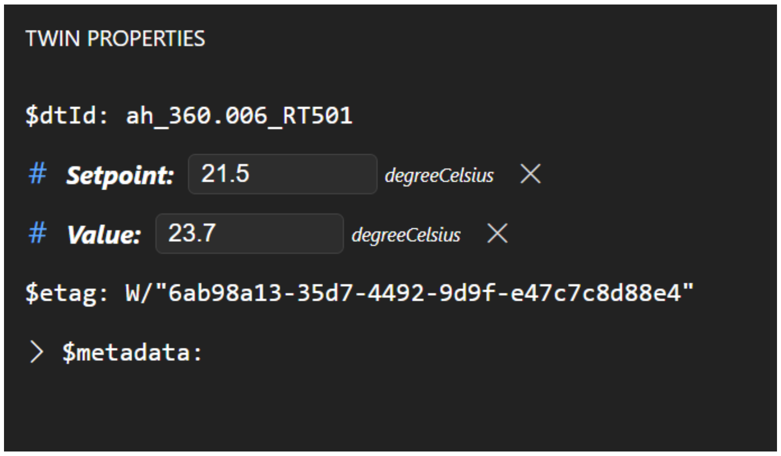


Figure 5.1: Digital Twin of Arkivenes Hus.

Real-life data is as mentioned distributed to the model, providing the nodes with the most current information, as seen in Figure 5.2. Continuously updating the nodes, enabling close monitoring. The relationships simplifies the process of identifying location of sensors and potential issues in the system.



TWIN PROPERTIES

$dtId: ah_360.006_RT501

Setpoint: 21.5 degreeCelsius

Value: 23.7 degreeCelsius

$etag: W/"6ab98a13-35d7-4492-9d9f-e47c7c8d88e4"

> $metadata:

Figure 5.2: Twin node of sensor RT501 updated with real-life data.

5.2 Sparse Identification of Nonlinear Dynamical Systems

5.2.1 Initial Results

The resulting formulas obtained from the SINDy algorithm describing the dynamics of the system are:

$$\begin{aligned}(u)' = &-4136.040 + 38.583u + 771.596T - 0.377u^2 - 4.040uT - 53.935T^2 \\ &- 0.002u^3 + 0.043u^2T + 0.117uT^2 + 1.674T^3 - 0.001u^2T^2 - 0.019T^4\end{aligned} \tag{5.1}$$

$$\begin{aligned}(T)' = &\, 14.356 + -0.505u + -3.033T + 0.009u^2 + 0.049uT \\ &+ 0.238T^2 + -0.001u^2T + -0.002uT^2 + -0.008T^3\end{aligned} \tag{5.2}$$

where u and T are the relative fan power represented by JV401 and the temperature measured in the room by RT501, respectively. The relations described above in Eqs. 5.1 and 5.2 where identified while fitting the model to the entire data period using sparse regression as described in section 2.2. After the fitting procedure, we conduct a one day simulation based on the model. Initial values of 0% fan power and a measured temperature of 20.5 by RT501 where provided to the model. These values are taken from the actual measured data on the 18th of April 2022, at 00:00. We simulate the system for 24hrs, and compare it to the actual measured data, as displayed in Fig. 5.3.

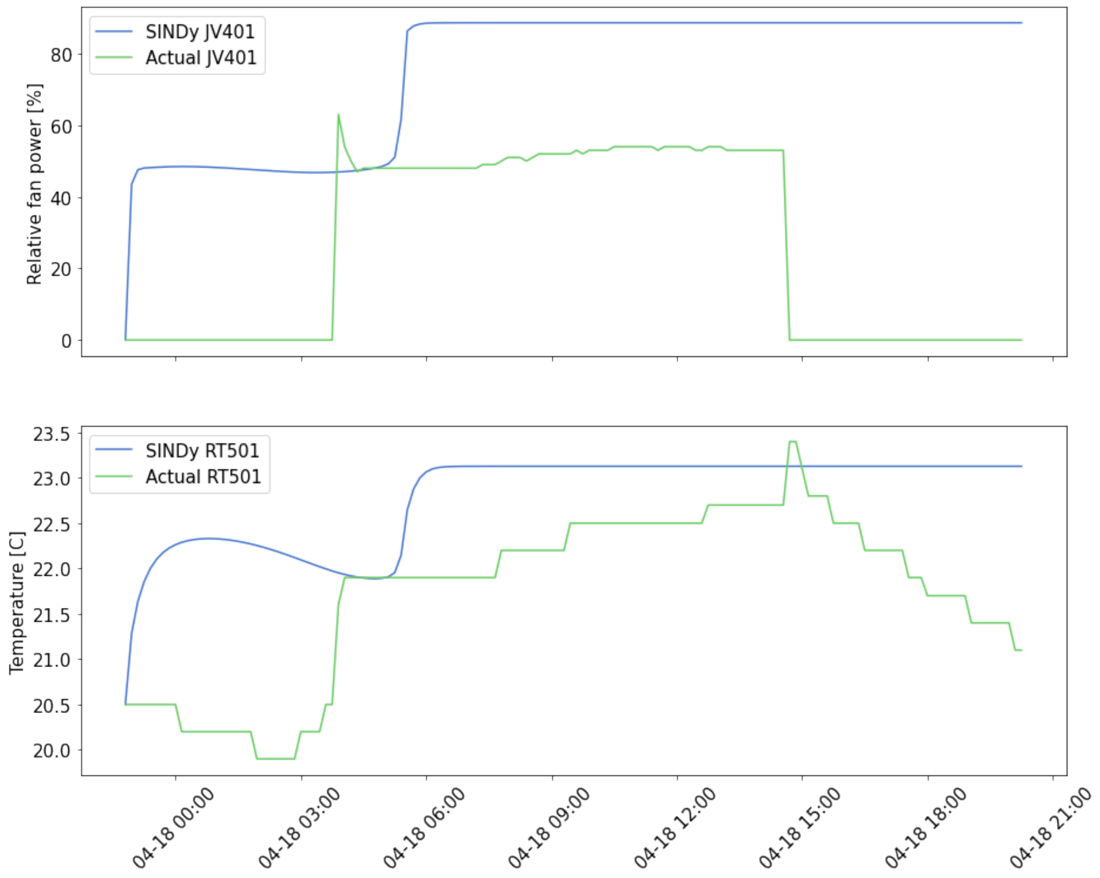


Figure 5.3: SINDy simulation and actual data from 18th of April 2022.

As seen in Fig. 5.3, the governing equations obtained from the model do not accurately describe the system. This is further discussed in Section 6.1.1.

5.2.2 Smoothed Model Results

Results using smoothed data to fit the SINDy model, as presented in Section 4.4.2 are presented here. The new Equations describing the system are given as:

$$(u)' = -174.655 + 0.195u + 24.504T - 0.013uT - 1.143T^2 + 0.018T^3 \tag{5.3}$$

$$(T)' = 2.05 + 0.029u - 0.003uT - 0.001T^2 \tag{5.4}$$

We simulate the system using this new model, hereby referred to as the smoothed model, with initial conditions equal to that of the simulation presented in 5.2.1. However, as identified by the initial model result, it is clear that the model does not capture dynamics late in the simulation (after half way point). This also holds for the smoothed model. We therefore choose to only observe how the model fits the first half of the day. The simulation results are shown in Fig. 5.4 below:

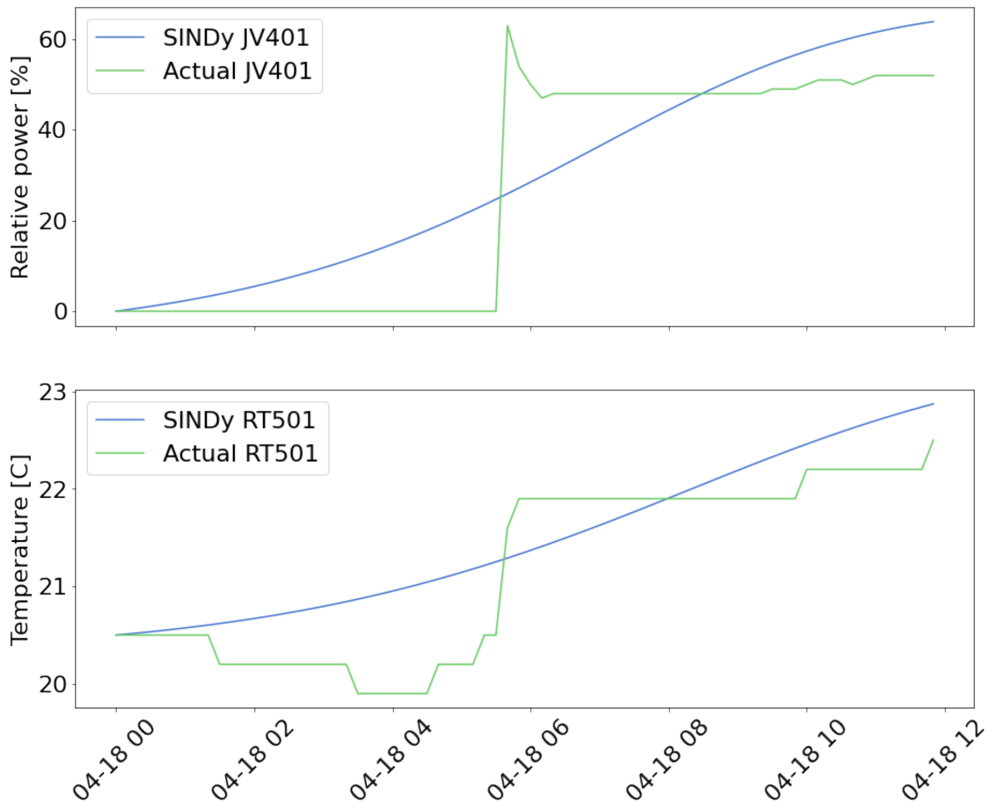


Figure 5.4: SINDy simulation and actual data from first half of 18th of April 2022.

Comparing the smoothed model simulation results in Fig. 5.4 to that of the initial model simulation in Fig. 5.3, it becomes clear that the initial offsets, as discussed in Section 6.1.1, are removed. However, further comparing Fig. 5.3 and 5.4, one can see that the smoothed model has weaker dynamics than that of the initial. The errors of the models are quantified using the mean square error, and are presented in Table 5.1:

Table 5.1: Mean square error evaluation on SINDy models compared to actual data.

	Initial model	Smoothed model
MSE JV401 (Fan power)	4347.89	148.85
MSE RT501 (Temperature)	3.53	0.29

Notice that the smoothed model perform better than the initial model on both metrics. The scores presented in Table 5.1 where calculated using values from midnight up to noon. Based on the significant lower errors in the smoothed model, we decide to use the smoothed model for further implementation with model predictive control.

5.3 Model Predictive Control

Fig. 5.5 depicts the identified control strategy for the fan power, JV401, given the identified smoothed model and respective temperature behaviour and constraints.

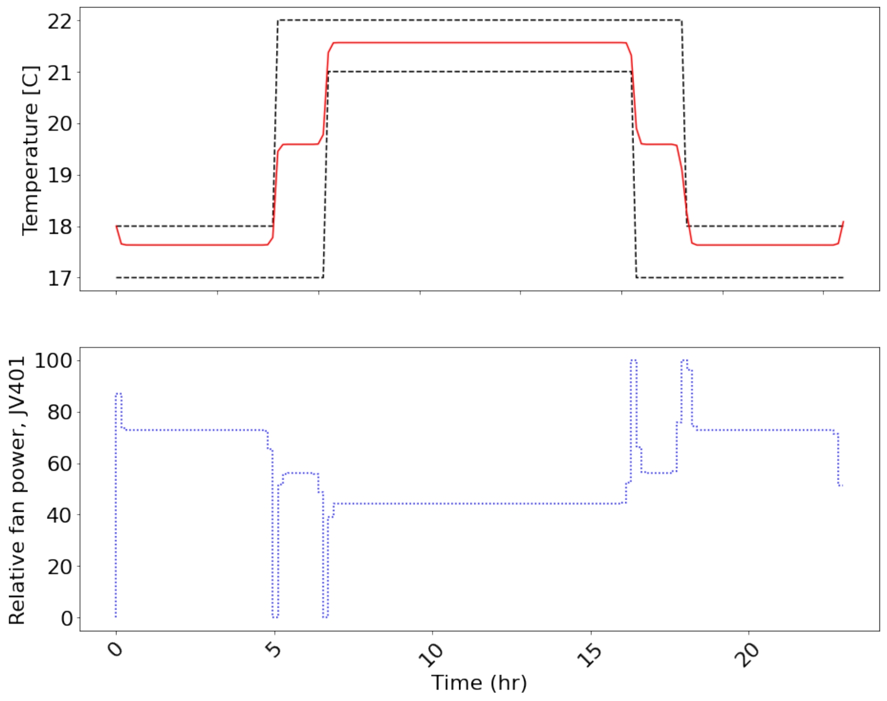


Figure 5.5: MPC strategy

As can be seen in Fig. 5.5, in order to keep the temperature (red, upper plot) within the constraints (black, upper plot), the required fan power is relatively high. Throughout the day, as the temperature is set to meet higher constraints, the required fan power is decreased. When the constraints are lowered, the required power once again increases.

5.4 Machine Learning Algorithms

Throughout this research a variation of machine learning models has been implemented, fitted and tested on our data; Linear Regression, Support Vector Regression (SVR), Extreme Gradient Boosting and Long Short-Term Memory (LSTM). This section highlights all results attained by the various models. All models have been trained on and applied to the same datasets in order to ensure an as accurate comparison as possible. In order to properly determine the best model for our data, we will evaluate the results based on calculated scores and errors.

5.4.1 Performance Evaluation

In order to evaluate the performance of the different models, a variation of error measurements and the model's R^2-score is calculated. Methods of error measurement used are as follows; Mean Squared Error (MSE), Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). The evaluation of all models is presented in table 5.2.

Table 5.2: Evaluation of implemented machine learning models

Model	MSE	MAE	RMSE	R-squared
Linear Regression	0.29	0.35	0.54	0.71
SVM Regression (RBF)	0.27	0.31	0.52	0.72
SVM Regression (Linear)	0.31	0.36	0.56	0.69
SVM Regression (Poly)	0.29	0.32	0.54	0.70
Extreme Gradient Boosting	0.25	0.30	0.50	0.74
LSTM	0.27	0.31	0.52	0.72

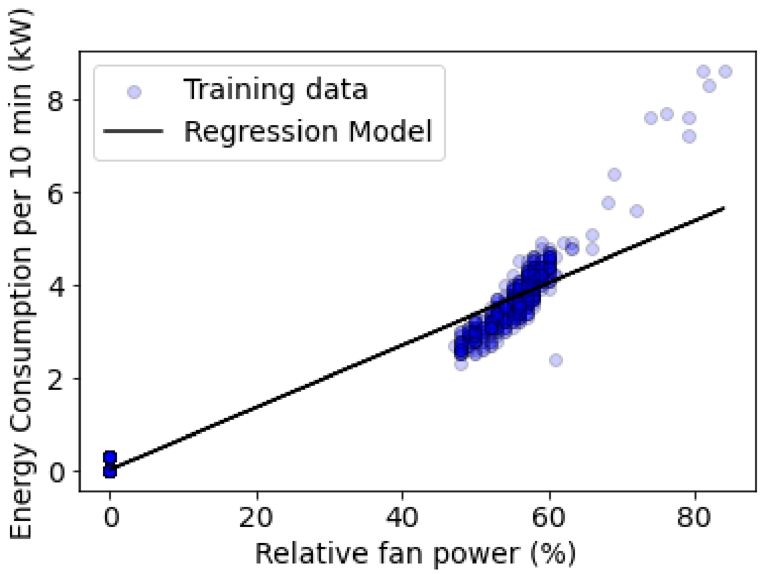
As can be interpreted form Table 5.2, all models perform at a similar level, with R^2-

scores around the high 60s and low 70s. Evaluating the models using the other methods, also results in a similar performance levels for all models. Our implementation of the Extreme gradient boosting model performs slightly better than the other models, achieving better results independent of evaluation method used.

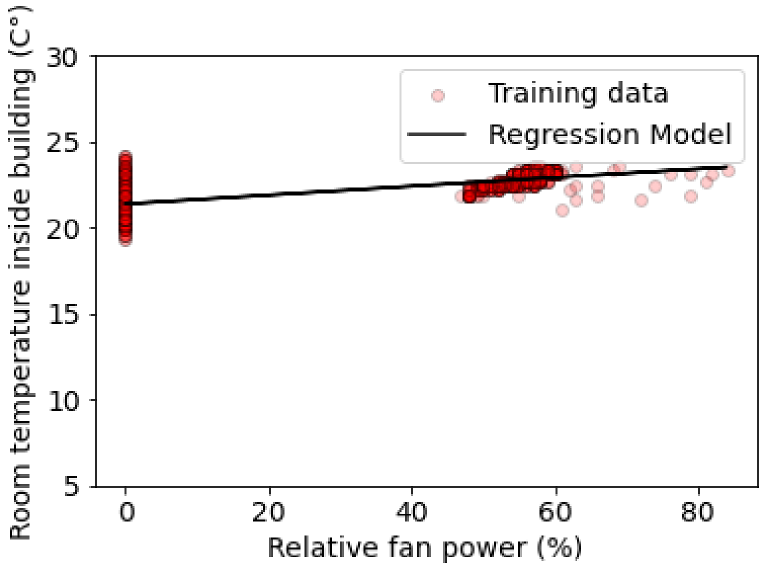
5.4.1.1 Linear Regression

Based on the results presented in Table 5.2, it becomes clear that Linear Regression is an acceptable model for predicting our target values, but not the most optimal of the ones tested. The model reaches an R^2-score of 0.71, meaning that 71% of the variation observed in the predicted data points can be accounted for by the x values, which represent the relative fan power of the ventilation system.

Figure 5.6 shows how well of a fit the linear regressor is for each of our target values. This is represented by a visualisation of our training data and the predictive Linear Regression model. It is clear that the model is not a perfect fit, however the model appears to fit some of the target values better than others. Figure 5.6a visualise the relation between the ventilation system's relative fan power and the systems energy consumption. As can be interpreted from the Figure, the relation is not perfectly linear, resulting in the model not being able to accurately predict values related to higher relative fan power. As for the relation between our input value and last target value, it is clear from Figure 5.6b that a Linear Regression might not be the optimal approach.



(a) Fit of model on target value = energy consumption



(b) Fit of model on target value = room temperature

Figure 5.6: Linear Regression model for energy consumption and room temperature as functions of the relative fan power.

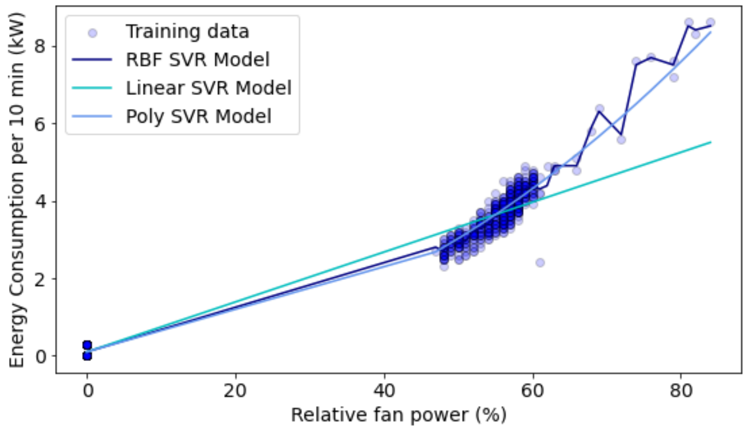
5.4.1.2 SVR

Table 5.3 shows a range of scores achieved by the different variations of the SVR model. The score is in this case affected by the change in kernel. The results clearly state that the SVR model running with a Gaussian RBF kernel performs the best on our data, reaching a R^2-score of 72%. In addition to the R^2 evaluation, the RBF kernel produces the lowest errors for all evaluation methods compared to the other kernels evaluated.

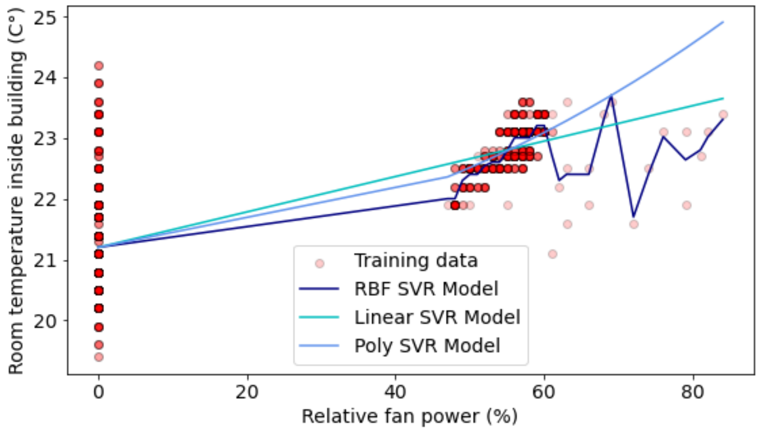
Table 5.3: Evaluation of SVM Regression Model

Kernel	Evaluation	Value
RBF	Mean Squared Error (MSE)	0.27
	Mean Absolute Error (MAE)	0.31
	Root Mean Squared Error (RMSE)	0.52
	R-squared score	0.72
Linear	Mean Squared Error (MSE)	0.31
	Mean Absolute Error (MAE)	0.36
	Root Mean Squared Error (RMSE)	0.56
	R-squared score	0.69
Polynomial	Mean Squared Error (MSE)	0.29
	Mean Absolute Error (MAE)	0.32
	Root Mean Squared Error (RMSE)	0.54
	R-squared score	0.70

Figures 5.7a and 5.7b indicates that the evaluation performed is correct, indicating that the RBF kernel is performing better than the polynomial and linear kernels. Looking at the Figures mentioned, we observe a much stronger fit in the model implemented with a RBF kernel. It is able to interpret the data at a higher level and that way captures a larger amount of the outliers compared to the other models running on different configurations.



(a) Kernel evaluation for target value = Energy consumption

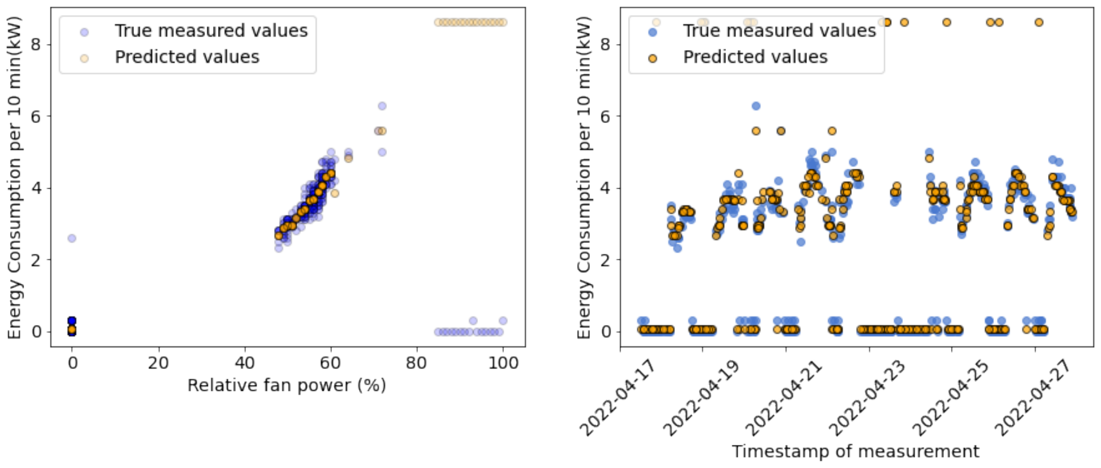


(b) Kernel evaluation for target value = Room temperature

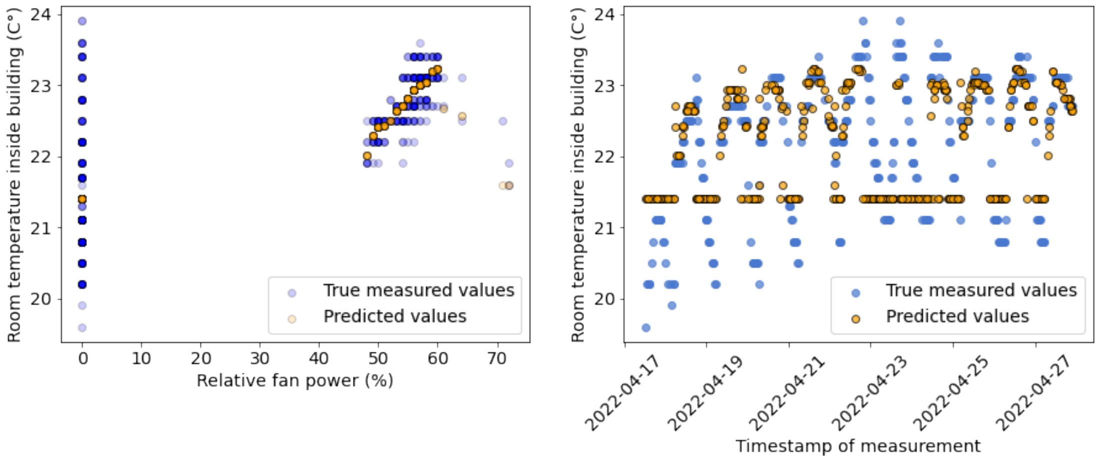
Figure 5.7: Evaluation of SVM Regression kernels.

5.4.1.3 Extreme Gradient Boosting

According to our evaluation, the implementation of the extreme gradient boosting algorithm achieves the best results out of all models tested. It has been fitted on our training dataset, just like the other models, and manages to achieve a satisfying R^2-score of 74%. When visualising the achieved results, Figure 5.8, we see a clear accuracy pattern across all target values. The model interprets our data sufficiently enough to be able to predict and place the target values with an acceptable accuracy. This is strongly represented in the following visualisations of the training data and predicted data. When plotting this data against the model input value, *Relative fan power (%)*, and it becomes quite clear that the model has managed to detect the patterns in the data to a varying extent for the different target values.



(a) Evaluation of XGB model for target - Energy consumption



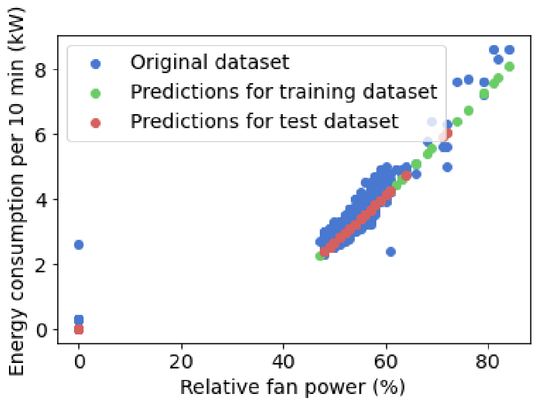
(b) Evaluation of XGB model for target - Room temperature

Figure 5.8: Extreme Gradient Boosting

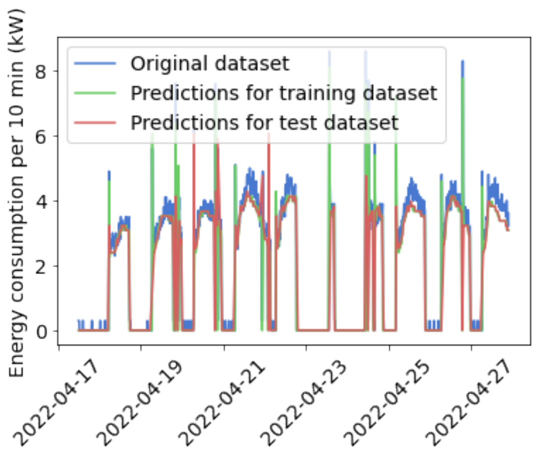
For the first target value, represented in Figure 5.8a, we see a clear similarity in pattern for the measured data and the predicted data. The model manages to place close to all values in the correct areas, making for great evaluation results. Looking at Figure 5.8b, we can still observe grate similarities in the pattern of both predicted and real values, but we observe a larger amount of deviation than for the first target. Looking at the Figure, we see that the model struggles to identify the real value of the temperature when the relative fan power is 0. This is due to temperature being strongly impacted by other non-linear parameters, resulting in a wide range of potential outputs. This is not the case for energy consumption, which is mostly impacted by the ventilation system's fan activity.

5.4.1.4 LSTM

The results presented in 5.2 indicates that the LSTM performs at reasonably high level, and is among our top three models with a R^2-score of 72%. To gain additional insight into the results achieved, the original dataset is plotted along with predictions made for both training data and test data. This is done for all target values. The results are provided in Figure 5.9 and 5.10. It is clear from plots 5.9a and 5.9b, that the model understands the nature of the original data to a great extent, interpreting the relation between the systems relative fan power and the measured energy consumption at an acceptable accuracy rate.



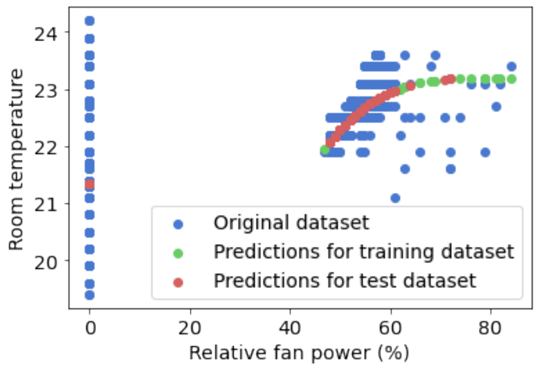
(a) Evaluation of predicted Energy consumption



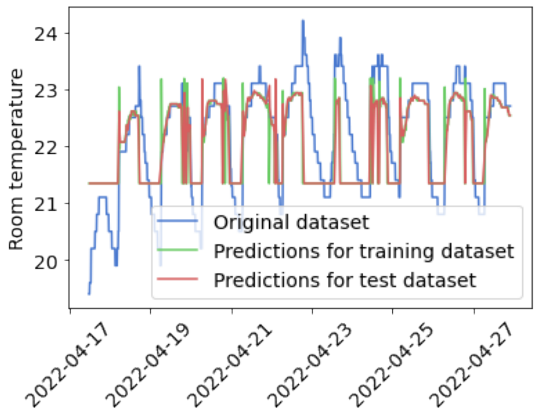
(b) Evaluation of predicted Energy consumption over time

Figure 5.9: Evaluation of LSTM trained on Regression Formulation of Energy consumption prediction problem

Figure 5.10 shows predictions made for indoor room temperature. We can clearly see that the model performance is different for the different target values based on how well the pattern of the predicted values present in Figure 5.10a and 5.10b match the pattern observed in the original data. The level of resemblance in the patterns observed in Figure 5.10 is substantially smaller than the similarity observed in Figure 5.9, implying a much lower accuracy for prediction of room temperature.



(a) Evaluation of predicted Energy consumption



(b) Evaluation of predicted Energy consumption over time

Figure 5.10: Evaluation of LSTM trained on Regression Formulation of Indoor Temperature prediction problem

5.5 System Results

The final step of our proposed architecture is to apply the best performing machine learning algorithm to the results found with MPC during optimization. Based on the results presented earlier in this chapter, XGBoost is the model of choice. By conducting this step we get an indication of how our suggested control strategy affects the energy consumption, allowing us to determine whether or not this is an optimization worth performing. Figure 5.11 illustrates the change in energy consumption after optimizing, and as can clearly be seen, the energy consumption is predicted to lay on a much higher level throughout the entire day if our control strategy is implemented.

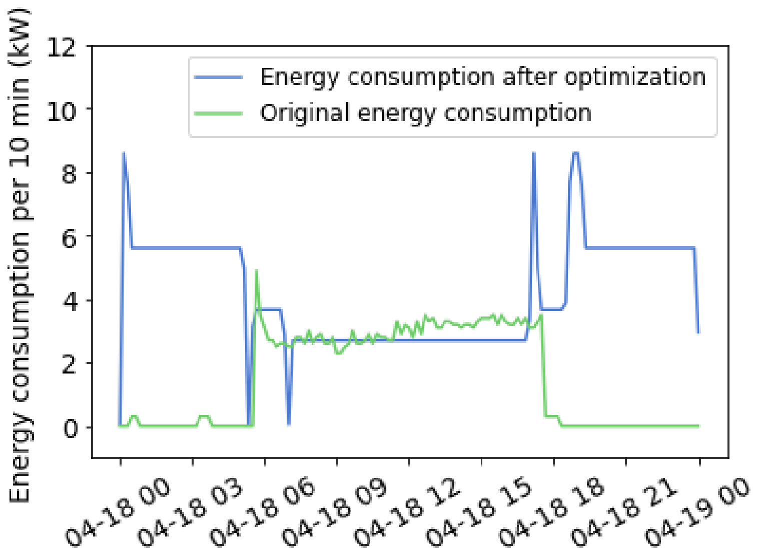


Figure 5.11: Energy consumption post optimization

Compared to today's energy consumption, the suggested control strategy appears to require a rather large increase in energy during the evening and night, whilst requiring a similar amount during the day. This, along with the numbers presented in table 5.4, suggests that the daily energy requirement will rise significantly, rising from roughly 220 kW to around 600 kW per day. Specifically, the numbers in 5.4 represent a 174.86% increase using the proposed control strategy compared to that of the current system.

Table 5.4: Daily energy consumption with and without temperature optimization

	Daily Energy consumption (kW)
Optimized for temperature	603.86792
Not optimized	219.70000

Chapter 6

Discussion

This chapter addresses all of the results reported in Chapter 5, arguing why they are as they are and whether the results are as expected or not. The chapter suggests modifications and alternative methodologies which we believe to be more appropriate for the studied problem.

6.1 Sparse Identification of Nonlinear Dynamics

6.1.1 Initial Model

As presented and shown in Fig. 5.3 in Section 5.2.1, the governing equations of the system obtained using SINDy do not accurately describe the system. Specifically, we notice large errors occurring in the beginning of the simulation, as highlighted by the red circles in Fig. 6.1 below.

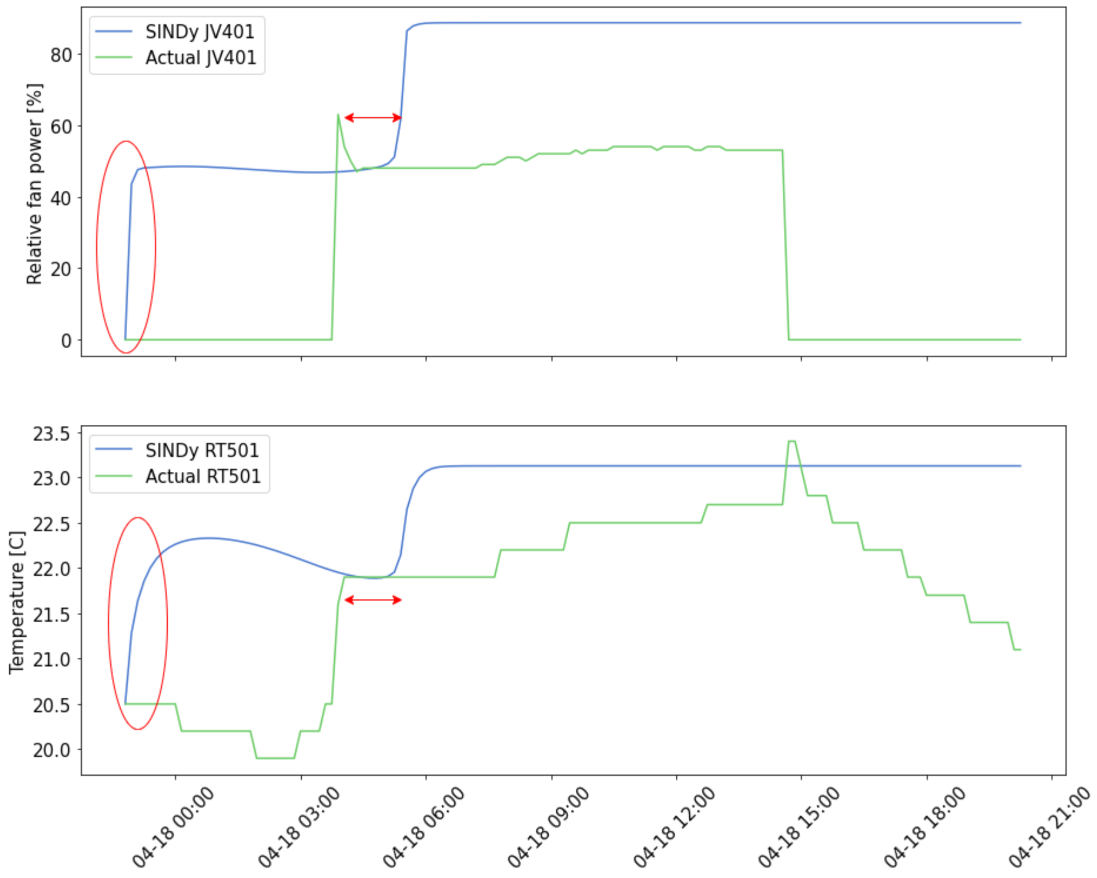


Figure 6.1: SINDy simulation and actual data from 18th of April 2022, with remarks.

These errors cause the model to offset compared to the original data. An effect of this offset is that even though the model can understand dynamics later in the simulation period, the predicted values themselves are affected by this initial offset. Further, the model seems to be affected by some lag in time when compared to the data. This is highlighted by the red arrows in Fig. 6.1. If we remove the original offset highlighted in the red circles, and also adjust the time lag highlighted by the red arrows, then the model would produce results as shown in Fig. 6.2. Note that this Figure does not depict real results, but shows the results from the simulation presented in Fig. 5.3 with edits.

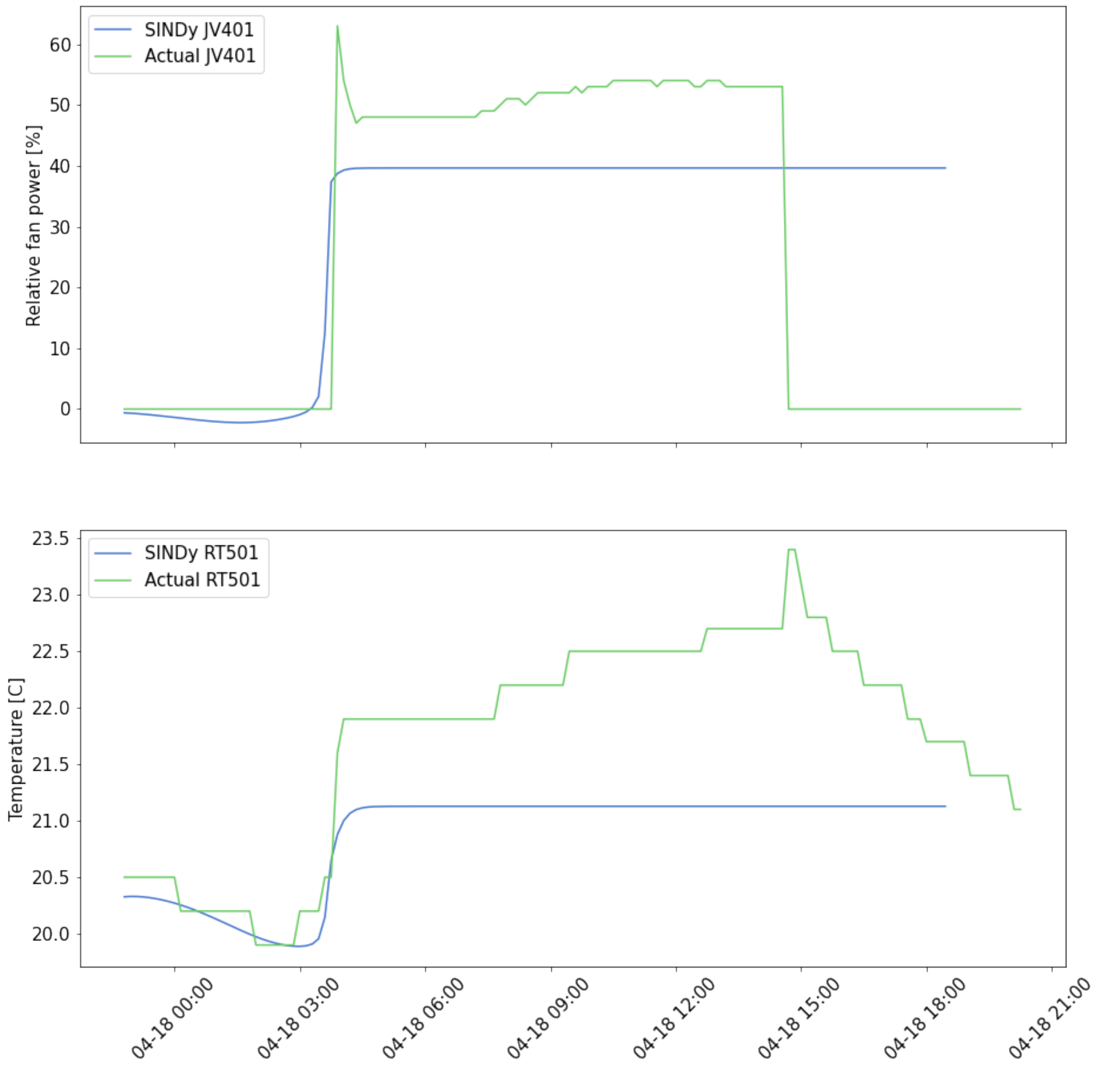


Figure 6.2: Edited SINDy simulation together with actual data from 18th of April 2022.

The edits made in Fig. 6.2 was to first account for the initial offset that occurred in the red circles in Fig. 6.1. This was done by simple approximation, where 40 was subtracted from all values noted in the upper JV401 graph and a subtraction of 2.0 was applied to all values in the lower RT501 graph. Secondly, to account for the lag in the response and to also remove the initial offset, we remove the first 2 hours of the simulation, thereby aligning the dynamics of the simulation with the dynamics shown in the real data. With the described adjustments, Fig 6.2 illustrates significantly better results. With the adjustments, we can see that the model in question actually manages to capture the dynamics of the system to some degree, as seen around the timestamp 04-18 03:00 in Fig. 6.2 for all graphs. The model does not, however, manage to capture later dynamics in the simulation. Given the poor results obtained from the simulation, it was decided not to carry on with implementing model predictive control (MPC) with SINDy, as was done in [30]. The main foundation for implementing SINDy with MPC is to have accurate equations governing the system, which is not the case here. However, had the result been more aligned with the edited results, it would have been of interest to implement MPC as a next step.

When selecting parameters from the model, we followed the "Practical tips" guide in the PySINDy documentation, which contains insights into selecting numerical differentiation schemes, library selection, optimization and regularization [73]. As discussed in the guide, numerical differentiation is one of the core components of SINDy as it provides the target $\dot{X}$ on the left hand side of Eq. 2.7. Due to the importance of $\dot{X}$, the model results are likely to suffer if the derivatives are calculated wrongly.

One possible explanation for the poor model performance may be caused by the noise in the data. The noise we refer to here is highlighted by the red circles in Fig. 6.3.

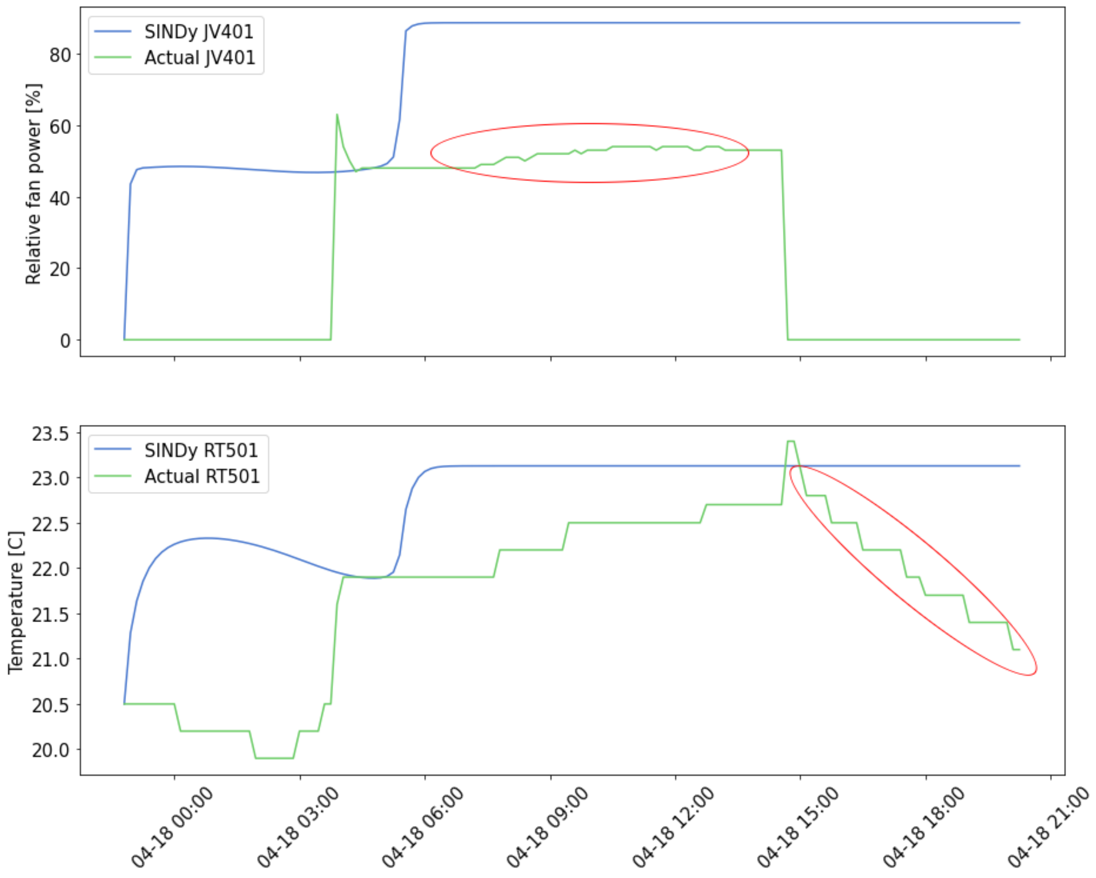


Figure 6.3: SINDy simulation and actual data from 18th of April 2022, with potential noise highlighted.

We did not initially consider this to be noise when fitting the model, as it represents the pure dynamics of the system. However, we here argue that these "noise-like" dynamics may cause poor model performance. The reason for this is that finite difference methods tend to intensify noise in the data [18]. As we see in Fig. 6.3, the data is not smooth. This can cause the finite difference method to yield derivatives with a larger amount of noise compared to that of the original data. A potential solution to this problem is to implement a SINDy model with smoothed finite difference as differentiation method. We consider this a interesting problem for future work.

Another potential solution to this problem is to smooth the data beforehand, and thereafter apply the finite difference method. We implement this method in Section 4.4.2, with accompanying results presented in Section 5.2.2.

6.1.2 Smoothed Model

The smoothed model removes the initial offsets illustrated in Fig. 6.1, and ensures that initial conditions remain similar for the model and that of the actual data. As can be seen in Fig. 6.1, the smoothed model contains weaker dynamics compared to that of the initial model. There seems to be no time lag compared to that of Fig. 6.1, but instead a smooth and steady increase. One drawback of this is that the characteristic dynamics of the system seem to vanish. This may be explained by the data used to train the smoothed model compared to that used to train the initial model, as illustrated in Fig. 4.8. Although the smoothed curves well captures the overall trend, they omit information about the step-like activation in e.g. the relative power plot. This is a weakness with smoothing the data. However, since the quantified errors from Tab. 5.1 in Section 5.2.2 show that the error is significantly reduced for the smoothed model, we decide to use this model for implementation with model predictive control.

6.1.3 Ensemble Model

Another possible cause for the poor model performance is the fact that the data combines slow variations (when the fan operates at a certain level), with sudden variations (when the fan changes operating mode). This can cause the derivatives to become quite large over time, as the system behaviour can almost be described by a step function, which the model is trying to smooth. It would therefore be very interesting to investigate the possibility of implementing a ensemble model, combining one SINDy model for the “slow” dynamics and one for the “fast” dynamics. We consider this relevant for future work.

6.2 Model Predictive Control

As stated in [24], successful deployment of MPC requires multiple aspects. Such aspects includes model development, test design and control design. The control design of the MPC presented in this project enables any feedforward setpoint profile to follow the desired strategy, as has been confirmed by the main author of works such as [38], [37] and [36], and the co-founder of the GEKKO optimization suite [10]. Therefore, we consider the main challenges of the MPC method deployed in this work to be related to that of the underlying model, which is discussed in Section 6.1, and test design. Testing the MPC is crucial as it can reveal potential run time stability- or numerical issues. It also allows for the comparison of response plots and that of desired trajectories. Further, implementing a estimator [41], which has the purpose to align the predictive model with current measurement is a natural extension after testing. We consider this relevant for future work.

6.3 Machine Learning Algorithms

A variation of machine learning algorithms has been tested throughout this research, all being popular for use with energy prediction and with a history of performing well for use cases such as ours. They are all quite fast and simple to implement and test with help of tools provided by Skicit-learn and TensorFlow. As presented in chapter 5, all methods reach a similar R^2-score, located in the high 60s and low 70s. The high similarity in performance is rather unexpected as the models operate in such different ways.

Despite being popular for energy prediction and holder of the characteristics mentioned, regression models such as Linear Regression models tend to struggle to meet the desired precision when utilized in prediction issues. This is said to be especially true for ventilation systems, as they are constantly affected by non-linear factors such as weather and occupancy [20]. As presented and shown in section 5.4 and Figure 5.6, the function found with the linear regression model, does not accurately describe the system. Non-linear factors influencing the system result in outliers and ”unexpected” values. These values are difficult for linear regression to capture and they affect the regression line in various ways. SVM for regression is often reached for when dealing with these expected external non-linear factors, and has been widely reported in recent years [19]. The result of replacing the classic linear regression with an implementation of an SVR model is presented in section 5.4, where an increase in R^2-score and decrease in error can be observed. Figure 5.7 clearly displays how the SVR model, with the correct kernel configuration, is able to capture the different variations/outliers occurring in the data and that way establish a non-linear relationship between the features with higher R^2-score.

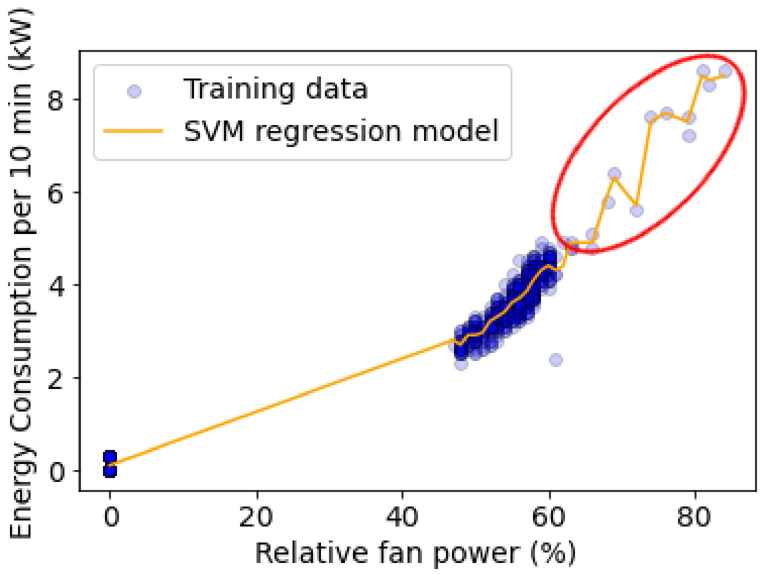


Figure 6.4: SVR capturing variation in data

As for the other models implemented, Extreme Gradient Boosting is the model with the highest level of performance, this is presented in table 5.2. There are other studies out there supporting this result, and Lu and Meng [99] state that XGBoost is the best model for predicting AC energy consumption in residential buildings in relation to a varying amount of input values. All target values for Energy consumption present in the training set are located within a specified range (0.0,8.8) which correspond to input values ranging from 0 to 84, covering the majority of the expected test data but not all. The target value in question is the energy consumed by the ventilation system per 10 minutes, we estimate that we can expect values ranging from 0 to somewhere around 10 kW. With a model like XGBoost, that means that in a case where the fan is set to its maximum power, resulting in an input value of 100, then the model will be restricted by the upper limit of the energy consumption present in the training data to some extent, possibly underestimating the increase in energy consumption as the fan power is increased. By extending the range of the training data to match the range of the expected real values, we could prevent this from happening and allow XGBoost to perform to the best of its ability as this is an area that Extreme Gradient Boosting is known for performing well in, due to its ability to control the model complexity and reduce overfitting [20].

LSTM was the only neural network implemented and tested, achieving an R^2-score of 72%. Neural networks, such as LSTM, is know for being able to approximate any continuous function to any desired level of precision, meaning it should be able to predict our target values even if a test value outside the range of the training data occurs. LSTM has previously been reported to score better than the other models when applied to short-term prediction despite being known for its ability to solve complex and long-time-lag tasks [20]. One issue with applying LSTM to our current data set can be related to neural networks' weakness related to overfitting the training data. A limited set of training data is prone to overfitting as a model, such as our LSTM model, might discover patterns and trends that in not in fact general for the values occurring in the test set.

Limitations related to our dataset will be further discussed.

6.3.0.1 Clusters in the Original Data

Common for the models discussed, is the effect clusters in the data have on the final result. As can be observed in Figure 6.6, clusters occur in the areas related to *Relative fan power* ≈ 0 and $45 \geq$ *Relative fan power* ≤ 65. Such clusters are caused by large collections of data points observed for similar input values. For this specific case, it is clear that our dataset consist of larger areas without any data points between these clusters, providing them with an even bigger impact on the models' interpretation of the data.

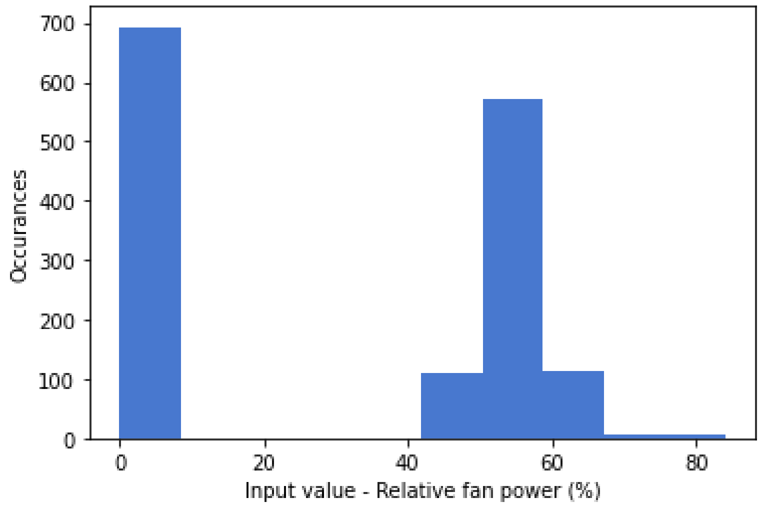


Figure 6.5: Histogram of input values

For the regression models in particular, the heavy weight of the clusters causes the an intersection between the regression line and the clusters, as this is where most of the interpreted relationships lay. The close proximity from the regression line to the clustered data points is a possible explanation as to why, for instance, the linear regression model is able to achieve a competitive R^2-score.

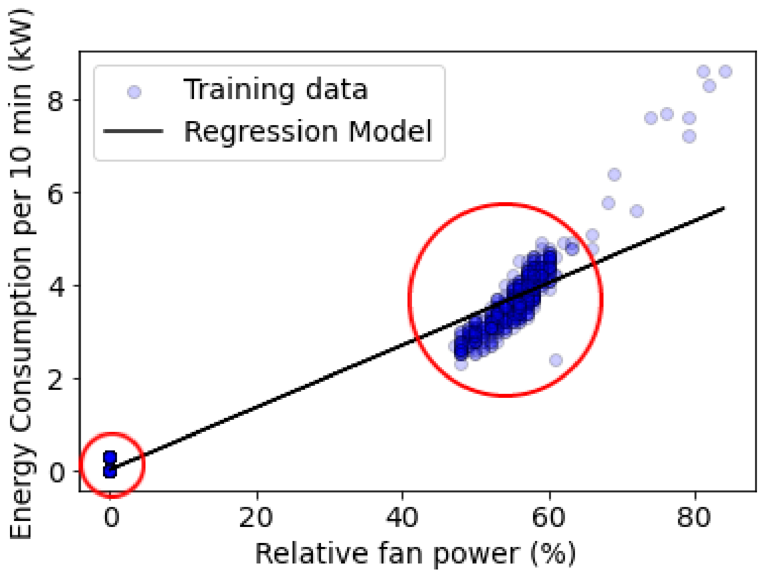


Figure 6.6: Clusters in data.

By accumulating additional data, consisting of points spanning a larger portion of the expected input range, the evaluation would have likely resulted different R^2-scores. Regression models, which utilize distance in their computation, might have been negatively affected as the generated regression lines might have shifted in position and the accumulated distance would be affected by a larger range of points and distances, not just the clusters. With a higher amount of data points representing the entirety of the input range, we might also be able to produce models with even higher scores than currently achieved. A wider range of data, would provide the model with more information and insight into how the input and targets relate to each other, allowing for the models to interpret and identify potential patterns, resulting in a higher accuracy score.

6.3.1 Model evaluation per target value

When analysing the results achieved it becomes clear that all off our models perform at a similar level, regardless of the type of model. The linear regression achieves an overall score in the same range as the other models, despite the other models' capability to introduce non-linear relationships and capture the nature of the data to a greater extent. In order to investigate this further we perform an evaluation of the model for each separate target value, by comparing the predicted value with the real measurement. Level of performance is determined with use of MSE and R^2-score, similar to the approach taken when evaluating the models as a whole. The results of the separate evaluations are presented in table 6.1.

	Energy consumption (kW)		**Room temperature (C°)**	
Algorithm	MSE	R^2	MSE	R^2
LR	0.1000	97%	0.4800	46%
SVM	0.0500	98%	0.4900	45%
XGB	0.0500	98%	0.4500	49%
LSTM	0.0600	98%	0.5500	37%

Table 6.1: Model Evaluation Per Target Value

The results clearly state that all the models tested are capable of interpreting the data related to Energy consumption with a very high R^2-score, whilst the performance achieved in the attempt to predict the respected Room temperature is much lower. For room temperature, no model is able achieve an R^2-score above 50%, which is much lower than the scores achieved for energy consumption. When evaluated for performance related to energy consumption, all models reach R^2-scores above 97%. The difference in R^2-score is most likely caused by the level off correlation, presented in Table 4.3, between the relative fan power (input) and the room temperature (output), which is quite a bit lower than for Energy consumption (output). Just based on these results alone it becomes clear that the weak performance lowers the total evaluation of the models, leaving them all with a similar overall score.

A potential solution to this issue could be to split the prediction, using separate and specified models for each of the target values, instead of implementing *Multi-output Regressors* which are currently utilized. Such an approach could enhance prediction accuracy, which is especially relevant for our Room temperature prediction problem. By researching the possibility of increasing the accuracy of predicted room temperature with use of different machine learning models, we could achieve an increase in the total performance across both targets. We consider this an interesting and valuable research area for future work.

6.4 Overall System

With regards to the final results presented in Section 5.5, it is clear that the approach taken and methods utilized is not optimal for the problem in question. We experience that the optimization performed is not as desired and that substituting the current control strategy used for ventilation systems in Arkivenes Hus with the one proposed by our system would only increases the buildings estimated energy consumption.

We argue here that a main driver for this significantly increased energy consumption is the underlying model used to describe the system. Specifically, in Fig. 5.11, the energy consumption is noticeably larger than the previous control strategy in periods when the previous control strategy did not require any energy, e.g. between 04-18 00 and 04-18 05 as well as 04-18 18 as 04-19 00. This increased energy requirement is a consequence of the high fan power in the same time periods, as shown in the lower plot in Fig. 5.5. This high fan power period occurs as the temperature is to be within the lower constraint range, as shown by the upper plot in 5.5. Fig. 6.7 shows the smoothed SINDy model simulation, as also presented in Section 5.2.2, with highlights.

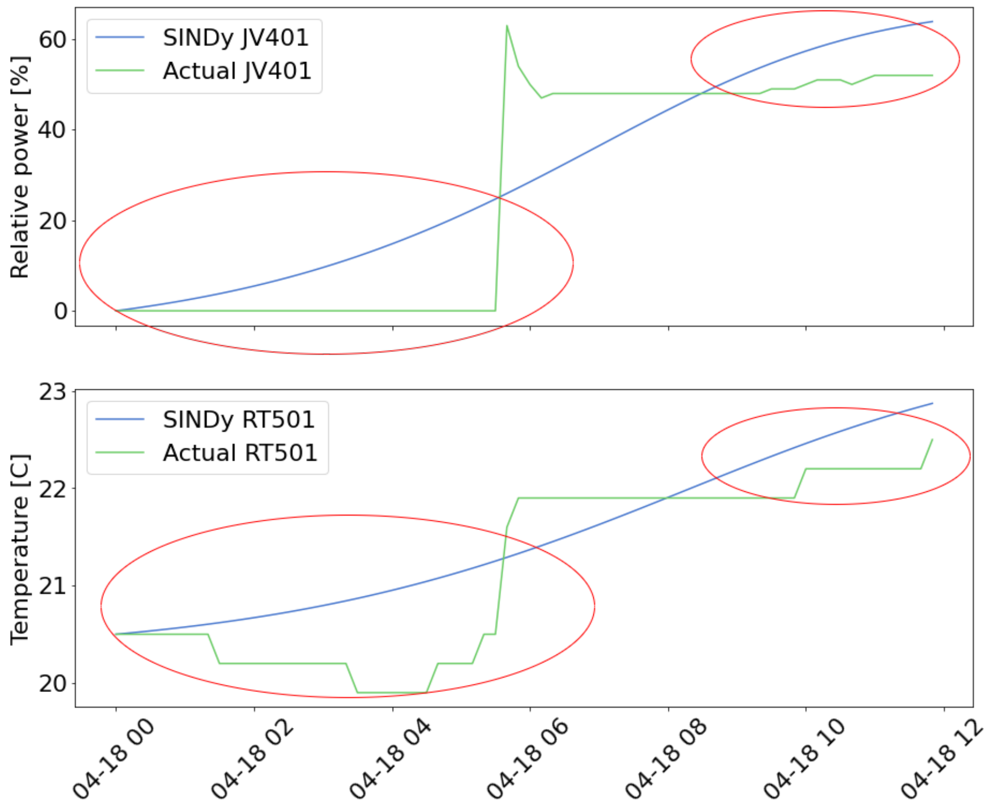


Figure 6.7: Smoothed SINDy model simulation with highlights.

In Fig. 6.7, the red circles (in both plots) highlight a interesting fact; that the smoothed SINDy model tend to produce values above that of the original data. We therefore believe that values obtained using this model are higher than what other potential models would produce.

It is important to consider the fact that Arkivenes Hus is a complex smart building with underlying algorithms controlling the behaviour of the different systems serving the building, such as ventilation systems. It is reasonable to believe that these algorithms influence the relation between our features to such an extent that identifying the actual relationships using our proposed methods is not the best approach. With additional information and knowledge related to these control algorithms, one might bee able to optimize the existing methods or introduce new methods, making the overall system more appropriate for solving the problem in question.

6.4.1 Other Approaches

Within the literature, one can classify the identification of ventilation system models into two categories, black box models and grey box models [1], [80], [58]. Black box models are based on data-driven approaches where one has no prior information. Grey box models are based on physical knowledge about the system. Due to difficulties with thermodynamics, developing a black box model using SINDy was chosen for this project. However, several other black box models have previously been implemented with success to model ventilation system. These include ARX [14], ARMAX [85], Box-Jenkins [16] and Output Error model identification [85]. It is of great interest to consider how these model would perform in the proposed architecture of this work instead of the SINDy model. This remains for future work.

Another possible solution to the temperature control problem would be to simply use one of the machine learning models to predict the fan power based on desired temperature. In such an approach, one would input a desired trajectory of temperature, and at each point of the trajectory let the model predict the corresponding fan power. The benefit of this approach is that the relation between fan power and temperature is learned directly from the data. No mathematical model is developed. As XGBoost provided the best results for the temperature to fan power relation, we reverse train a XGBoost model, meaning that we develop a model to predict fan power based on temperature. We then feed the model with a desired temperature trajectory. Fig. 6.8 below displays the predicted relative fan power (lower plot) based on the temperature trajectory (upper plot).

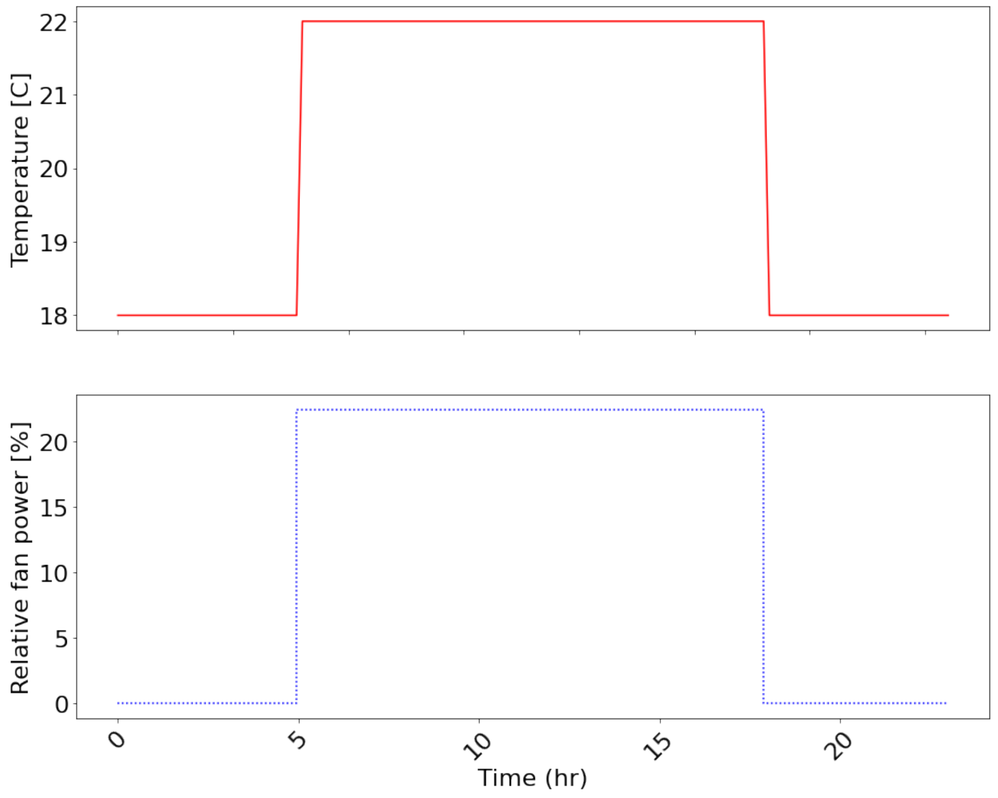


Figure 6.8: Predicted fan power based on temperature trajectory, XGBoost.

Since the trained model learns relations directly from the data of the original system, we notice that the required fan power to keep 18 °C reflects that of original data display in e.g. Fig. 5.4. This strategy illustrates significantly lower required fan power compared to that of Fig. 5.5. However, the strength of this model (not requiring a mathematical model) is also it's main weakness. Since we do not obtain a mathematical model, we cannot implement such a solution with model predictive control, which requires a model describing the dynamics. We thus loose the flexibility that comes with implementing MPC. Such include the ability of handling multi-input multi-output systems with interactions between inputs and outputs, changes in constraints and preview capabilities.

Chapter 7

Conclusions

In this work, we have constructed a data integration system for sensors within a building to liberate data. This data has been utilized on one use case, optimizing indoor temperature comfort and studying the effects on energy usage by one of the ventilation systems in the building.

We propose an architecture for this optimization consisting of three steps. The first step is to identify a mathematical model describing a relation between indoor temperature and the relative fan power of the ventilation system. This is done using sparse identification of nonlinear dynamics. The second step is to implement a control strategy for the fan based on changing constraints in temperature throughout the day. This is achieved using model predictive control. Lastly, we use machine learning to predict the energy usage of the ventilation system based on the proposed control strategy for the ventilation fan as obtained by model predictive control. We studied several machine learning models for energy prediction, and found that XGBoost was the best for this step. The development and the description of the methodology involved in the optimization problem at hand constitutes a core part of the thesis.

With respect to a one day simulation, the proposed architecture yield a 174.86% increase in energy consumption compared to that of the current control system.

For the proposed solution, the main weakness and cause of error lies in the identified SINDy model describing the relation between temperature and fan power. Alternative models which may provide an improved result are briefly outlined in the next chapter.

Chapter 8

Future Work

Some points about future work have already been mentioned throughout the Thesis. We here summarize these as well as point to other thoughts about future directions:

- Implement a SINDy model with smoothed finite difference method.
- Implement a ensemble SINDy model combining “slow” and “fast”.
- Testing the MPC scheme and implementing a estimator.
- It would be of great interest to compare the MPC scheme to some other data-driven method such as e.g. reinforcement learning.
- An urgent need exists to identify and implement a better mathematical model describing the relation between temperature and fan power in the ventilation system.
- Although the current proposed solution results in higher energy usage, it is still of interest to test the system physically, as physical operation will verify the simulation results.

- With time, more data is gathered by the data acquisition system. The data foundation thus grows. It is therefore of interest to implement a system to automatically re-train the models based on the increased volume of data, and study the effects of this.

Appendix A

Defined entities for digital twin

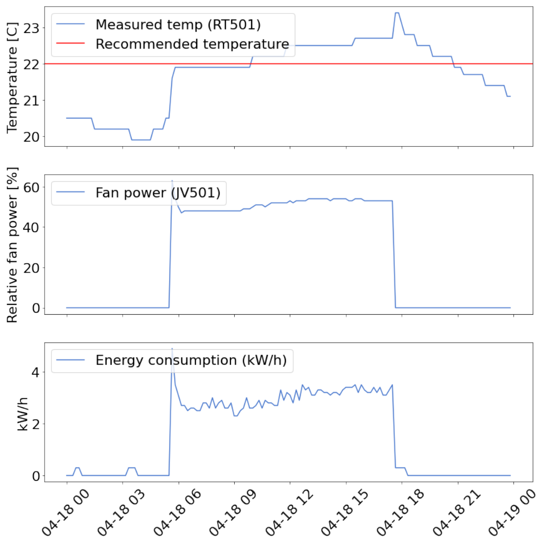
Table A.1: Entities defined for digital twin

Entity	**ID**	**Properties**	**Relationships**
Building	ah		contains contains_floor
Floor	ah_floor_{floor number}		contains contains_room
Room	ah_room_{room type}_{room number}	*'Area'*:double	contains_sensor
Archive	ah_room_archive _{room number}	*'Area'*:double	contains_sensor
Technical Room	ah_room_technical_{room number}	*'Area'*:double	contains_sensor
Shaft	ah_shaft_{shaft type}_{room number}	*'Area'*:double	contains_sensor

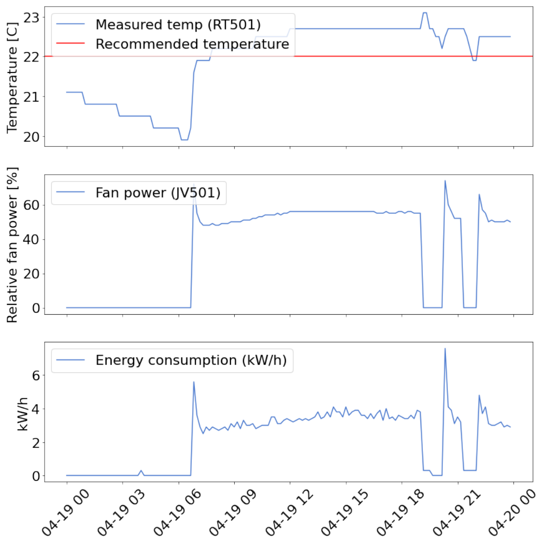
Ventilation System	ah_ventilation_{systemID}		serves contains_sensor
Temperature Sensor	ah_{systemID}_{sensorID}	*'Temperature'*:double *'Setpoint'*: double	
SFP Sensor	ah_{systemID}_{sensorID}	*'SFP'*: double	
Pressure Sensor	ah_{systemID}_{sensorID}	*'Pressure'*: double *'Setpoint'*: double	
Relative Performance Sensor	ah_{systemID}_{sensorID}	*'Relative_performance'*: double *'MaxValue'*: double *'MinValue'*: double	
Humidity Sensor	ah_{systemID}_{sensorID}	*'relative_humidity'*: double *'Setpoint'*: double	
Filter Sensor	ah_{systemID}_{sensorID}	*'Clean'*: boolean	
Damper Sensor	ah_{systemID}_{sensorID}	*'Open'*: boolean	
Battery Sensor	ah_{systemID}_{sensorID}	*'Warm'*: boolean	
Airflow Sensor	ah_{systemID}_{sensorID}	*'Air_per_second'*: double	

Appendix B

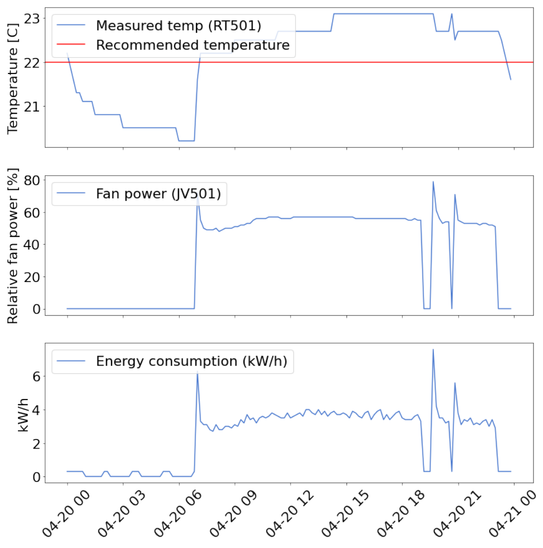
Daily sensor and energy measurements



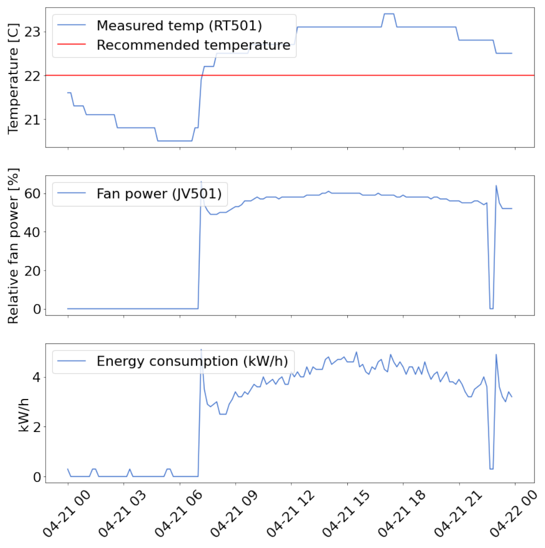
(a) Sensor measurements and energy consumption April 18th 2022



(b) Sensor measurements and energy consumption April 19th 2022

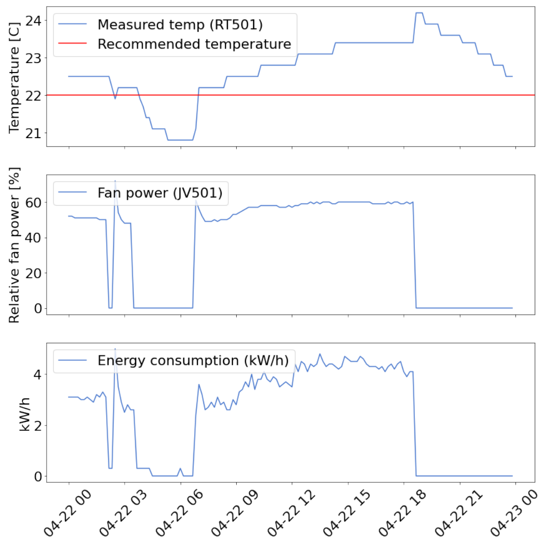


(c) Sensor measurements and energy consumption April 20th 2022

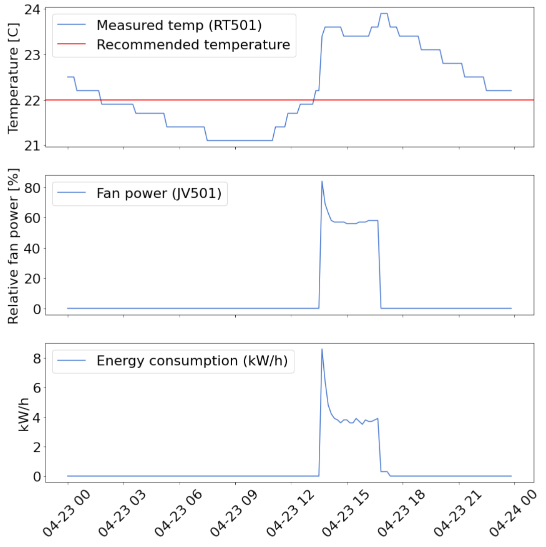


(d) Sensor measurements and energy consumption April 21th 2022

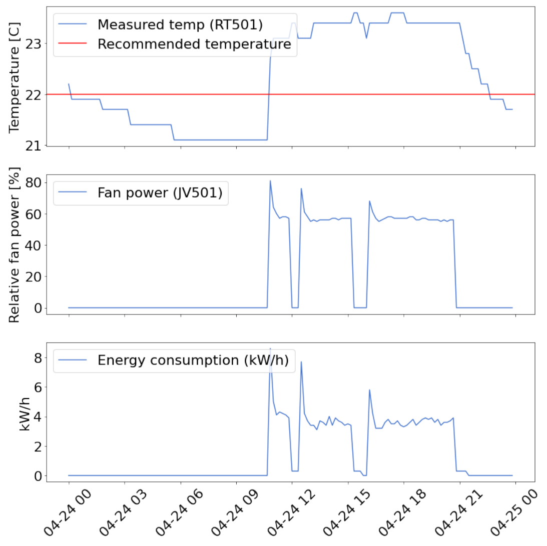
Figure B.1: Daily sensor measurements and energy consumption - Part 1



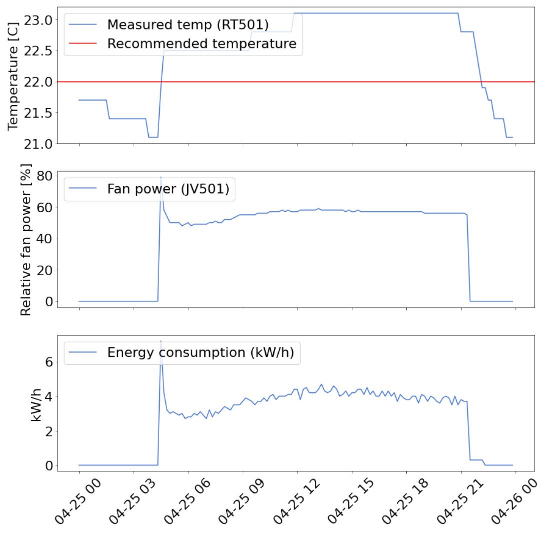
(a) Sensor measurements and energy consumption April 22th 2022



(b) Sensor measurements and energy consumption April 23rd 2022



(c) Sensor measurements and energy consumption April 24th 2022



(d) Sensor measurements and energy consumption April 25th 2022

Figure B.2: Daily sensor measurements and energy consumption - Part 2

Appendix C

Correlation analysis result

	Value (kW)	Value Heat	JP451_C	JP461_C	JV401_C	JV401_D	JV501_C	JV501_D	LX471_C	Operat-ing_Mode
Value (kW)	1	-0.38	-0.38	0.33	0.98	0.96	0.96	0.96	0.72	0.96
Value Heat	-0.38	1	0.37	-0.23	-0.41	-0.41	-0.43	-0.41	-0.21	-0.41
JP451_C	-0.38	0.37	1	-0.11	-0.43	-0.44	-0.43	-0.44	-0.31	-0.44
JP461_C	0.33	-0.23	-0.11	1	0.29	0.26	0.29	0.26	-0.23	0.26
JV401_C	0.98	-0.41	-0.43	0.29	1	0.99	0.99	0.99	0.75	0.99
JV401_D	0.96	-0.41	-0.44	0.26	0.99	1	0.99	1	0.77	1
JV501_C	0.96	-0.43	-0.43	0.29	0.99	0.99	1	0.99	0.74	0.99
JV501_D	0.96	-0.41	-0.44	0.26	0.99	1	0.99	1	0.77	1
LX471_C	0.72	-0.21	-0.31	-0.23	0.75	0.77	0.74	0.77	1	0.77
Operat-ing_Mode	0.96	-0.41	-0.44	0.26	0.99	1	0.99	1	0.77	1
RF401_MV	0.99	-0.4	-0.43	0.3	1	0.99	0.99	0.99	0.74	0.99
RF501_MV	0.97	-0.43	-0.43	0.3	0.99	0.99	1	0.99	0.74	0.99
RP401_MV	0.97	-0.38	-0.42	0.24	0.98	0.98	0.96	0.98	0.79	0.98
RP401_SP	-0.88	0.37	0.31	-0.26	-0.92	-0.94	-0.94	-0.94	-0.68	-0.94
RP501_MV	0.9	-0.41	-0.42	0.26	0.95	0.97	0.97	0.97	0.71	0.97
RT401_MV	-0.89	0.23	0.26	-0.07	-0.91	-0.92	-0.91	-0.92	-0.79	-0.92
RT402_MV	-0.54	0.03	0.13	0.29	-0.58	-0.62	-0.58	-0.62	-0.86	-0.62
RT451_MV	-0.31	0.87	0.34	-0.23	-0.32	-0.31	-0.34	-0.31	-0.07	-0.31
RT501_MV	0.71	-0.29	-0.32	0.28	0.71	0.69	0.71	0.69	0.47	0.69
RT551_MV	0.38	-0.97	-0.34	0.21	0.41	0.43	0.44	0.43	0.26	0.43

Table C.1: Correlation analysis performed on original historical dataset - Part 1

	RF401_MV	RF501_MV	RP401_MV	RP401_SP	RP501_MV	RT401_MV	RT402_MV	RT451_MV	RT501_MV	RT551_MV
Value (kW)	0.99	0.97	0.97	-0.88	0.9	-0.89	-0.54	-0.31	0.71	0.38
Value Heat	-0.4	-0.43	-0.38	0.37	-0.41	0.23	0.03	0.87	-0.29	-0.97
JP451_C	-0.43	-0.43	-0.42	0.31	-0.42	0.26	0.13	0.34	-0.32	-0.34
JP461_C	0.3	0.3	0.24	-0.26	0.26	-0.07	0.29	-0.23	0.28	0.21
JV401_C	1	0.99	0.98	-0.92	0.95	-0.91	-0.58	-0.32	0.71	0.41
JV401_D	0.99	0.99	0.98	-0.94	0.97	-0.92	-0.62	-0.31	0.69	0.43
JV501_C	0.99	1	0.96	-0.94	0.97	-0.91	-0.58	-0.34	0.71	0.44
JV501_D	0.99	0.99	0.98	-0.94	0.97	-0.92	-0.62	-0.31	0.69	0.43
LX471_C	0.74	0.74	0.79	-0.68	0.71	-0.79	-0.86	-0.07	0.47	0.26
Operat-ing_Mode	0.99	0.99	0.98	-0.94	0.97	-0.92	-0.62	-0.31	0.69	0.43
RF401_MV	1	0.99	0.98	-0.91	0.94	-0.91	-0.56	-0.32	0.71	0.41
RF501_MV	0.99	1	0.96	-0.94	0.96	-0.91	-0.58	-0.34	0.71	0.44
RP401_MV	0.98	0.96	1	-0.87	0.92	-0.91	-0.62	-0.27	0.67	0.4
RP401_SP	-0.91	-0.94	-0.87	1	-0.94	0.91	0.57	0.29	-0.72	-0.38
RP501_MV	0.94	0.96	0.92	-0.94	1	-0.88	-0.58	-0.31	0.68	0.42
RT401_MV	-0.91	-0.91	-0.91	0.91	-0.88	1	0.71	0.15	-0.69	-0.26
RT402_MV	-0.56	-0.58	-0.62	0.57	-0.58	0.71	1	-0.15	-0.26	-0.12
RT451_MV	-0.32	-0.34	-0.27	0.29	-0.31	0.15	-0.15	1	-0.32	-0.72
RT501_MV	0.71	0.71	0.67	-0.72	0.68	-0.69	-0.26	-0.32	1	0.25
RT551_MV	0.41	0.44	0.4	-0.38	0.42	-0.26	-0.12	-0.72	0.25	1

Table C.2: Correlation analysis performed on original historical dataset - Part 2

Appendix D

Poster

Smart Building Data Collection and Ventilation System Energy Prediction

Aleksander Bogunovic Jakobsen, Unni Johanna Blilie Kinstad

Introduction

Data has the potential to transform our environments for the better if utilized to its full potential. A highly interesting use case of data is in relation to Smart Buildings, where IoT technology presents new possibilities. With appropriate collection and structuring of the available data, many new opportunities present themselves.

In this thesis, a data gathering system is proposed for sensors in Arkivenes Hus. To illustrate the potential in the data, one specific problem is researched, namely that of indoor climate optimization and studying the effects on energy usage.

Main objectives

1. Collect and structure sensor and energy data.
2. Identify governing equations describing relationships in ventilation data.
3. Identify optimal control strategy.
4. Predict energy consumption based on control strategy.

Proposed solution

We propose a data acquisition system as illustrated in Fig. 1

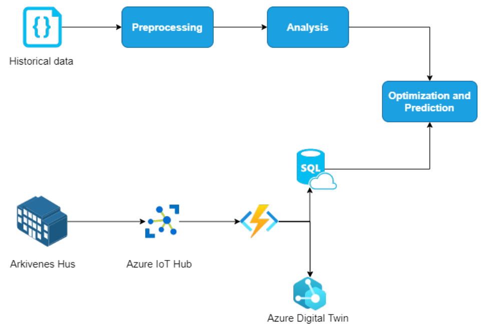


Figure 1: Overall system architecture.

with the optimization and prediction scheme as shown in Fig. 2

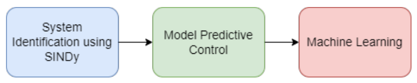


Figure 2: Proposed solution for optimization and prediction.

Experiments/Results

Using Sparse Identification of Nonlinear Dynamics (SINDy) [1], we obtain a model as shown in Fig. 3.

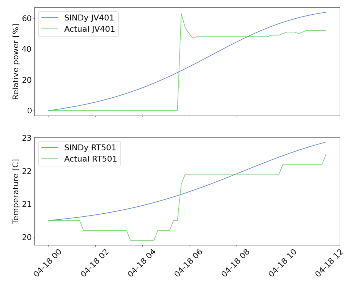


Figure 3: SINDy model result.

We design a MPC, using the acquired relation, to minimize fan power while keeping temperature within constraints, as seen in Fig. 4

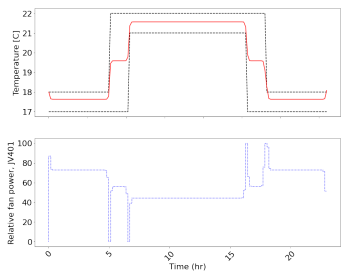


Figure 4: Proposed solution for optimization and prediction.

We implement several ML methods, and find that extreme gradient boosting [2] provide the best results when predicting energy consumption based on fan power. Using this, we achieve a 174.86% increase in required energy consumption compared to the current control strategy.

Discussion

With regards to the final results presented, it is clear that the approach taken and methods utilized is not optimal for the problem in question. We experience that the optimization performed is not as desired and that substituting the current control strategy used for ventilation systems in Arkivenes Hus with the one proposed by our system would only increases the buildings estimated energy consumption.

Conclusion

- Standalone MPC and machine learning deliver desired results.
- SINDy model is not accurate enough, need to investigate other options.
- Current solution yields 174.86% energy increase.

References

[1] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016.

[2] Rangjian et al. Qiu. Generalized extreme gradient boosting model for predicting daily global solar radiation for locations without historical data.

title=References]refs

Bibliography

[1] Abdul Afram and Farrokh Janabi-Sharifi. “Black-Box Modeling of Residential HVAC System and Comparison of Grey-Box and Black-Box Modeling Methods”. In: *Energy and Buildings* 94 (Feb. 1, 2015), pp. 121–149. DOI: 10.1016/j.enbuild.2015.02.045.

[2] Abdul Afram and Farrokh Janabi-Sharifi. “Theory and Applications of HVAC Control Systems – A Review of Model Predictive Control (MPC)”. In: *Building and Environment* 72 (Feb. 1, 2014), pp. 343–355. ISSN: 0360-1323. DOI: 10.1016/j.buildenv.2013.11.016. URL: https://www.sciencedirect.com/science/article/pii/S0360132313003363 (visited on 05/23/2022).

[3] Jethro Akroyd et al. “Universal Digital Twin - A Dynamic Knowledge Graph”. In: *Data-Centric Engineering* 2 (2021/ed). ISSN: 2632-6736. DOI: 10.1017/dce.2021.10. URL: http://www.cambridge.org/core/journals/data-centric-engineering/article/universal-digital-twin-a-dynamic-knowledge-graph/FD25CDFF886CD2ED33D1FDFC13F6BEAB (visited on 02/25/2022).

[4] Shervine Amidi and Afshine Amidi. *CS 230 - Recurrent Neural Networks Cheatsheet*. URL: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks (visited on 05/24/2022).

[5] *An Introduction to Sparse Identification of Nonlinear Dynamical Systems (SINDy) — Pysindy 1.8.Dev1+g288cf6e Documentation*. URL: https://pysindy.readthedocs.

io / en / latest / examples / 2 _ introduction _ to _ sindy . html (visited on 05/04/2022).

[6] Pushkar P. Apte and Costas J. Spanos. "The Digital Twin Opportunity". In: *MIT Sloan Management Review* 63.1 (Aut. 2021), pp. 15–17. ISSN: 15329194. URL: http://www.proquest.com/docview/2573028817/abstract/7CFF5AED73E34755PQ/1 (visited on 02/24/2022).

[7] *Arkivenes Hus*. Smedvig. URL: https : / / www . smedvig . com / no / property / arkivenes-hus/ (visited on 06/13/2022).

[8] *Azure SQL Database – Managed Cloud Database Service | Microsoft Azure*. URL: https : / / azure . microsoft . com / en - us / products / azure - sql / database/ (visited on 05/13/2022).

[9] Baumit. *Importance of Indoor Climate | Baumit.Com*. int.baumit.com. URL: https : / / int . baumit . com / guide / healthy - living - 1 / importance - of - indoor-climate (visited on 05/23/2022).

[10] Logan D. R. Beal et al. "GEKKO Optimization Suite". In: *Processes* 6.8 (8 Aug. 2018), p. 106. ISSN: 2227-9717. DOI: 10 . 3390 / pr6080106. URL: https : //www.mdpi.com/2227-9717/6/8/106 (visited on 05/20/2022).

[11] *BIM Software for Architects, Engineers and Construction Industry*. Solibri. URL: https://www.solibri.com/ (visited on 06/13/2022).

[12] *BREEAM - BRE Group*. Feb. 23, 2022. URL: https://bregroup.com/products/breeam/ (visited on 06/13/2022).

[13] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems". In: *Proceedings of the National Academy of Sciences* 113.15 (Apr. 12, 2016), pp. 3932–3937. DOI: 10 . 1073 / pnas . 1517384113. URL: https : / / www . pnas.org/doi/10.1073/pnas.1517384113 (visited on 05/04/2022).

[14] Eric M. Burger and Scott J. Moura. “ARX Model of a Residential Heating System With Backpropagation Parameter Estimation Algorithm”. In: ASME 2017 Dynamic Systems and Control Conference. American Society of Mechanical Engineers Digital Collection, Nov. 14, 2017. DOI: 10.1115/DSCC2017-5315. URL: https://pressurevesseltech.asmedigitalcollection.asme.org/DSCC/proceedings/DSCC2017/58295/V003T42A003/454972 (visited on 05/23/2022).

[15] Ovidiu Calin. “Introductory Problems”. In: *Deep Learning Architectures: A Mathematical Approach*. Ed. by Ovidiu Calin. Springer Series in the Data Sciences. Cham: Springer International Publishing, 2020, pp. 3–19. ISBN: 978-3-030-36721-3. DOI: 10.1007/978-3-030-36721-3_1. URL: https://doi.org/10.1007/978-3-030-36721-3_1 (visited on 03/07/2022).

[16] Gulben Calis et al. “Forecasting Occupancy for Demand Driven HVAC Operations Using Time Series Analysis”. In: *Journal of Asian Architecture and Building Engineering* 16.3 (Sept. 2017), pp. 655–660. ISSN: 1346-7581, 1347-2852. DOI: 10.3130/jaabe.16.655. URL: https://www.tandfonline.com/doi/full/10.3130/jaabe.16.655 (visited on 05/23/2022).

[17] cephalin. *Environment Variables and App Settings Reference - Azure App Service*. URL: https://docs.microsoft.com/en-us/azure/app-service/reference-app-settings (visited on 05/25/2022).

[18] Rick Chartrand. “Numerical Differentiation of Noisy, Nonsmooth Data”. In: *ISRN Applied Mathematics* 2011 (May 11, 2011), e164564. DOI: 10.5402/2011/164564. URL: https://www.hindawi.com/journals/isrn/2011/164564/ (visited on 05/12/2022).

[19] Yibo Chen and Hongwei Tan. “Short-Term Prediction of Electric Demand in Building Sector via Hybrid Support Vector Regression”. In: *Applied Energy* 204 (Oct. 15, 2017), pp. 1363–1374. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.

2017.03.070. URL: https://www.sciencedirect.com/science/article/pii/S0306261917303082 (visited on 03/03/2022).

[20] Yongbao Chen et al. “Physical Energy and Data-Driven Models in Building Energy Prediction: A Review”. In: *Energy Reports* 8 (Nov. 1, 2022), pp. 2656–2671. ISSN: 2352-4847. DOI: 10.1016/j.egyr.2022.01.162. URL: https://www.sciencedirect.com/science/article/pii/S2352484722001615 (visited on 02/25/2022).

[21] Yongbao Chen et al. “Short-Term Electrical Load Forecasting Using the Support Vector Regression (SVR) Model to Calculate the Demand Response Baseline for Office Buildings”. In: *Applied Energy* 195 (June 1, 2017), pp. 659–670. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2017.03.034. URL: https://www.sciencedirect.com/science/article/pii/S0306261917302581 (visited on 03/03/2022).

[22] *Correlation Analysis*. URL: https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_correlation-regression/bs704_correlation-regression2.html (visited on 06/13/2022).

[23] *Correlation Analysis | Analyze Correlation between Two Variables | QuestionPro*. URL: https://www.questionpro.com/features/correlation-analysis.html (visited on 06/13/2022).

[24] Mark Darby. “MPC: Current Practice and Challenges”. In: *Control Engineering Practice* 20.4 (Apr. 1, 2012), pp. 328–342. ISSN: 0967-0661. DOI: 10.1016/j.conengprac.2011.12.004. URL: http://www.sciencedirect.com/science/article/pii/S0967066111002528 (visited on 05/22/2022).

[25] *Data - W3C*. URL: https://www.w3.org/standards/semanticweb/data (visited on 02/21/2022).

[26] Brian de Silva et al. "PySINDy: A Python Package for the Sparse Identification of Nonlinear Dynamical Systems from Data". In: *Journal of Open Source Software* 5.49 (May 18, 2020), p. 2104. ISSN: 2475-9066. DOI: `10.21105/joss.02104`. URL: `https://joss.theoj.org/papers/10.21105/joss.02104` (visited on 05/05/2022).

[27] *Digital Twins – Modeling and Simulations | Microsoft Azure*. URL: `https://azure.microsoft.com/en-us/services/digital-twins/` (visited on 02/21/2022).

[28] *Digital Twins Definition Language*. Microsoft Azure, Feb. 20, 2022. URL: `https://github.com/Azure/opendigitaltwins-dtdl/blob/2b52b68b9646ce97f7ce3626dbb2ee3b4ebbcf` `DTDL/v2/dtdlv2.md` (visited on 02/21/2022).

[29] Robert DiPietro and Gregory D. Hager. "Chapter 21 - Deep Learning: RNNs and LSTM". In: *Handbook of Medical Image Computing and Computer Assisted Intervention*. Ed. by S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger. The Elsevier and MICCAI Society Book Series. Academic Press, Jan. 1, 2020, pp. 503–519. ISBN: 978-0-12-816176-0. DOI: `10.1016/B978-0-12-816176-0.00026-0`. URL: `https://www.sciencedirect.com/science/article/pii/B9780128161760000260` (visited on 03/02/2022).

[30] Urban Fasel et al. "SINDy with Control: A Tutorial". Aug. 30, 2021. arXiv: `2108.13404 [math]`. URL: `http://arxiv.org/abs/2108.13404` (visited on 05/12/2022).

[31] *Finite Difference Approximating Derivatives — Python Numerical Methods*. URL: `https://pythonnumericalmethods.berkeley.edu/notebooks/chapter20.02-Finite-Difference-Approximating-Derivatives.html` (visited on 05/11/2022).

[32] Rohith Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. Medium. July 5, 2018. URL: `https://towardsdatascience.com/`

support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (visited on 03/03/2022).

[33] ggailey777. *Azure Functions Documentation*. URL: https://docs.microsoft.com/en-us/azure/azure-functions/ (visited on 05/25/2022).

[34] Yabin Guo et al. "Machine Learning-Based Thermal Response Time Ahead Energy Demand Prediction for Building Heating Systems". In: *Applied Energy* 221 (July 1, 2018), pp. 16–27. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2018.03.125. URL: https://www.sciencedirect.com/science/article/pii/S030626191830463X (visited on 02/28/2022).

[35] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer, 2009. ISBN: 978-0-387-84857-0 978-0-387-84858-7. DOI: 10.1007/978-0-387-84858-7. URL: http://link.springer.com/10.1007/978-0-387-84858-7 (visited on 05/04/2022).

[36] J.D. Hedengren, K.V. Allsford, and J. Ramlal. "Moving Horizon Estimation and Control for an Industrial Gas Phase Polymerization Reactor". In: Proceedings of the American Control Conference. 2007, pp. 1353–1358. ISBN: 978-1-4244-0988-4. DOI: 10.1109/ACC.2007.4282820.

[37] J.D. Hedengren and A.N. Eaton. "Overview of Estimation Methods for Industrial Dynamic Systems". In: *Optimization and Engineering* 18.1 (2017), pp. 155–178. ISSN: 1389-4420. DOI: 10.1007/s11081-015-9295-9.

[38] John D. Hedengren et al. "Nonlinear Modeling, Estimation and Predictive Control in APMonitor". In: *Computers & Chemical Engineering*. Manfred Morari Special Issue 70 (Nov. 5, 2014), pp. 133–148. ISSN: 0098-1354. DOI: 10.1016/j.compchemeng.2014.04.013. URL: https://www.sciencedirect.com/science/article/pii/S0098135414001306 (visited on 05/22/2022).

[39] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 15, 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `https://doi.org/10.1162/neco.1997.9.8.1735` (visited on 03/07/2022).

[40] Morten Hovd. “A Brief Introduction to Model Predictive Control”. In: (), p. 28.

[41] Morten Hovd and Robert R. Bitmead. “Interaction between Control and State Estimation in Nonlinear MPC”. In: *IFAC Proceedings Volumes* 37.9 (July 2004), pp. 119–124. ISSN: 14746670. DOI: `10.1016/S1474-6670(17)31803-7`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S1474667017318037` (visited on 05/23/2022).

[42] Sergey Ivanov et al. “Digital Twin of City: Concept Overview”. In: *2020 Global Smart Industry Conference (GloSIC)*. 2020 Global Smart Industry Conference (GloSIC). Nov. 2020, pp. 178–186. DOI: `10.1109/GloSIC50886.2020.9267879`.

[43] Sweta Jain, Pruthviraj Choudhari, and Ayushi Srivastava. “Chapter 1 - The Fundamentals of Internet of Things: Architectures, Enabling Technologies, and Applications”. In: *Healthcare Paradigms in the Internet of Things Ecosystem*. Ed. by Valentina E. Balas and Souvik Pal. Academic Press, Jan. 1, 2021, pp. 1–20. ISBN: 978-0-12-819664-9. DOI: `10.1016/B978-0-12-819664-9.00001-6`. URL: `https://www.sciencedirect.com/science/article/pii/B9780128196649000016` (visited on 02/24/2022).

[44] Gareth James et al. *An Introduction to Statistical Learning*. Vol. 103. Springer Texts in Statistics. New York, NY: Springer, 2013. ISBN: 978-1-4614-7137-0 978-1-4614-7138-7. DOI: `10.1007/978-1-4614-7138-7`. URL: `http://link.springer.com/10.1007/978-1-4614-7138-7` (visited on 05/04/2022).

[45] JnHs. *Azure Portal Documentation - Azure Portal*. URL: `https://docs.microsoft.com/en-us/azure/azure-portal/` (visited on 05/25/2022).

[46] *JSON-LD Contexts | NIEM GitHub.* URL: https://niem.github.io/json/reference/json-ld/context/ (visited on 02/24/2022).

[47] Sakdirat Kaewunruen and Ningfang Xu. "Digital Twin for Sustainability Evaluation of Railway Station Buildings". In: *Frontiers in Built Environment* 4 (2018). ISSN: 2297-3362. URL: https://www.frontiersin.org/article/10.3389/fbuil.2018.00077 (visited on 02/25/2022).

[48] kgremban. *Compare Event Grid, Routing for IoT Hub.* URL: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-event-grid-routing-comparison (visited on 03/30/2022).

[49] kgremban. *IoT Concepts and Azure IoT Hub.* URL: https://docs.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub (visited on 03/30/2022).

[50] Johanna Kinstad. *Detecting Gender of the Users Based on Thei Twitter Profile.* Sept. 2021.

[51] D. R. Kiran. "Chapter 35 - Internet of Things". In: *Production Planning and Control.* Ed. by D. R. Kiran. Butterworth-Heinemann, Jan. 1, 2019, pp. 495–513. ISBN: 978-0-12-818364-9. DOI: 10.1016/B978-0-12-818364-9.00035-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128183649000354 (visited on 02/07/2022).

[52] Jin Li. "Assessing the Accuracy of Predictive Models for Numerical Data: Not r nor R2, Why Not? Then What?" In: *PLoS ONE* 12.8 (Aug. 24, 2017), e0183250. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0183250. pmid: 28837692. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5570302/ (visited on 06/13/2022).

[53] *Linear Regression.* URL: http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm (visited on 02/28/2022).

[54] *Linked Data - Design Issues.* URL: https://www.w3.org/DesignIssues/LinkedData.html (visited on 02/25/2022).

[55] LitKnd. *What Is the Azure SQL Database Service? - Azure SQL Database.* URL: https://docs.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview (visited on 02/24/2022).

[56] Yuqian Lu et al. "Digital Twin-driven Smart Manufacturing: Connotation, Reference Model, Applications and Research Issues". In: *Robotics and Computer-Integrated Manufacturing* 61 (Feb. 1, 2020), p. 101837. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2019.101837. URL: https://www.sciencedirect.com/science/article/pii/S0736584519302480 (visited on 02/24/2022).

[57] Mateev M. "INDUSTRY 4.0 AND THE DIGITAL TWIN FOR BUILDING INDUSTRY". In: *Industry 4.0* 5.1 (2020), pp. 29–32. URL: https://stumejournals.com/journals/i4/2020/1/29 (visited on 02/09/2022).

[58] Marcel Macarulla et al. "Use of Grey-Box Modeling to Determine the Air Ventilation Flows in a Room". In: Jan. 1, 2021, pp. 449–461. ISBN: 978-3-030-54409-6. DOI: 10.1007/978-3-030-54410-2_32.

[59] Veronica Martinez Hernandez et al. "Service Business Model Innovation: The Digital Twin Technology". In: (Jan. 18, 2019). DOI: 10.17863/CAM.35482. URL: https://www.repository.cam.ac.uk/handle/1810/288166 (visited on 02/24/2022).

[60] Amol Mavuduru. *Why XGBoost Can't Solve All Your Problems.* Medium. Nov. 10, 2020. URL: https://towardsdatascience.com/why-xgboost-cant-solve-all-your-problems-b5003a62d12a (visited on 05/24/2022).

[61] Benoît Mercereau et al. "Fighting Climate Change as a Global Equity Investor". In: *Journal of Asset Management* 21.1 (Feb. 1, 2020), pp. 70–83. ISSN: 1479-179X.

DOI: 10.1057/s41260-020-00150-9. URL: https://doi.org/10.1057/s41260-020-00150-9 (visited on 05/25/2022).

[62] *Microsoft Azure - Introduction to Azure SQL Database*. GeeksforGeeks. Aug. 20, 2021. URL: https://www.geeksforgeeks.org/microsoft-azure-introduction-to-azure-sql-database/ (visited on 03/03/2022).

[63] Simon P. Neill and M. Reza Hashemi. "Chapter 8 - Ocean Modelling for Resource Characterization". In: *Fundamentals of Ocean Renewable Energy*. Ed. by Simon P. Neill and M. Reza Hashemi. E-Business Solutions. Academic Press, Jan. 1, 2018, pp. 193–235. ISBN: 978-0-12-810448-4. DOI: 10.1016/B978-0-12-810448-4.00008-2. URL: https://www.sciencedirect.com/science/article/pii/B9780128104484000082 (visited on 05/12/2022).

[64] *NumPy*. URL: https://numpy.org/ (visited on 05/15/2022).

[65] Chigozie Nwankpa et al. "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning". Nov. 8, 2018. arXiv: 1811.03378 [cs]. URL: http://arxiv.org/abs/1811.03378 (visited on 05/24/2022).

[66] De-Graft Opoku et al. "Digital Twin Application in the Construction Industry: A Literature Review". In: *Journal of Building Engineering* 40 (May 1, 2021), p. 102726. DOI: 10.1016/j.jobe.2021.102726.

[67] *Original Paper: Sparse Identification of Nonlinear Dynamical Systems (SINDy) — Pysindy 1.8.Dev1+g288cf6e Documentation*. URL: https://pysindy.readthedocs.io/en/latest/examples/3_original_paper.html (visited on 05/11/2022).

[68] Oscar Ortiz, Francesc Castells, and Guido Sonnemann. "Sustainability in the Construction Industry: A Review of Recent Developments Based on LCA". In: *Construction and Building Materials* 23.1 (Jan. 1, 2009), pp. 28–39. ISSN: 0950-0618. DOI: 10.1016/j.conbuildmat.2007.11.012. URL: https://www.

sciencedirect.com/science/article/pii/S0950061807003005 (visited on 05/25/2022).

[69] P. E. Orukpe. *Basics of Model Predictive Control.* Apr. 14, 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.177.4756&rep=rep1&type=pdf.

[70] *Pandas Documentation — Pandas 1.4.2 Documentation.* URL: https://pandas.pydata.org/docs/# (visited on 05/21/2022).

[71] Fabian Pedregosa et al. "Scikit-Learn: Machine Learning in Python". In: *MACHINE LEARNING IN PYTHON* (), p. 6.

[72] Michael Phi. *Illustrated Guide to LSTM's and GRU's: A Step by Step Explanation.* Medium. June 28, 2020. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21 (visited on 03/07/2022).

[73] *Practical Tips — Pysindy 1.7 Documentation.* URL: https://pysindy.readthedocs.io/en/stable/tips.html (visited on 05/12/2022).

[74] *PySINDy.* dynamicslab, May 3, 2022. URL: https://github.com/dynamicslab/pysindy/blob/288cf6e895b1534509d70d3f979261e9384e4564/examples/2_introduction_to_sindy.ipynb (visited on 05/04/2022).

[75] *Pysindy.Differentiation Package — Pysindy 1.8.Dev1+g288cf6e Documentation.* URL: https://pysindy.readthedocs.io/en/latest/api/pysindy.differentiation.html (visited on 05/11/2022).

[76] *Pysindy.Optimizers.Stlsq — Pysindy 1.8.Dev1+g288cf6e Documentation.* URL: https://pysindy.readthedocs.io/en/latest/_modules/pysindy/optimizers/stlsq.html (visited on 05/11/2022).

[77] James Blake Rawlings, David Q. Mayne, and Moritz Diehl. *Model Predictive Control: Theory, Computation, and Design.* 2nd edition. Madison, Wisconsin: Nob Hill Publishing, 2017. 623 pp. ISBN: 978-0-9759377-3-0.

[78] *Report to Congress on Indoor Air Quality. Volume 2. Assessment and Control of Indoor Air Pollution. Final Report.* PB-90-167396/XAB; EPA-400/1-89/001C. Environmental Protection Agency, Washington, DC (USA), Aug. 1, 1989. URL: `https://www.osti.gov/biblio/6958939-report-congress-indoor-air-quality-volume-assessment-control-indoor-air-pollution-final-report` (visited on 05/24/2022).

[79] Luca Riboldi et al. "An Integrated Assessment of the Environmental and Economic Impact of Offshore Oil Platform Electrification". In: *Energies* 12.11 (11 Jan. 2019), p. 2114. ISSN: 1996-1073. DOI: `10.3390/en12112114`. URL: `https://www.mdpi.com/1996-1073/12/11/2114` (visited on 05/25/2022).

[80] Sullivan Royer et al. "Black-Box Modeling of Buildings Thermal Behavior Using System Identification". In: *IFAC Proceedings Volumes.* 19th IFAC World Congress 47.3 (Jan. 1, 2014), pp. 10850–10855. ISSN: 1474-6670. DOI: `10.3182/20140824-6-ZA-1003.01519`. URL: `https://www.sciencedirect.com/science/article/pii/S1474667016433392` (visited on 05/23/2022).

[81] "Mean Absolute Error". In: *Encyclopedia of Machine Learning.* Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 652–652. ISBN: 978-0-387-30164-8. DOI: `10.1007/978-0-387-30164-8_525`. URL: `https://doi.org/10.1007/978-0-387-30164-8_525` (visited on 05/12/2022).

[82] "Mean Squared Error". In: *Encyclopedia of Machine Learning.* Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 653–653. ISBN: 978-0-387-30164-8. DOI: `10.1007/978-0-387-30164-8_528`. URL: `https://doi.org/10.1007/978-0-387-30164-8_528` (visited on 05/12/2022).

[83] Ronald W. Schafer. “What Is a Savitzky-Golay Filter? [Lecture Notes]”. In: *IEEE Signal Processing Magazine* 28.4 (July 2011), pp. 111–117. ISSN: 1558-0792. DOI: `10.1109/MSP.2011.941097`.

[84] *Scikit-Learn: Machine Learning in Python — Scikit-Learn 1.1.0 Documentation.* URL: `https://scikit-learn.org/stable/index.html` (visited on 05/13/2022).

[85] Francesco Scotton. “Modeling and Identificaiton for HVAC Systems”. Stockhom, Sweeden: KTH Electrical Engineering, Aug. 2012. URL: `http://www.diva-portal.se/smash/get/diva2:547622/FULLTEXT01.pdf`.

[86] *Smart Building.* IOT Factory. URL: `https://iotfactory.eu/solutions/smart-building/` (visited on 05/26/2022).

[87] Nivethitha Somu, Gauthama Raman M r, and Krithi Ramamritham. “A Hybrid Model for Building Energy Consumption Forecasting Using Long Short Term Memory Networks”. In: *Applied Energy* 261 (Mar. 1, 2020), p. 114131. ISSN: 0306-2619. DOI: `10.1016/j.apenergy.2019.114131`. URL: `https://www.sciencedirect.com/science/article/pii/S0306261919318185` (visited on 02/25/2022).

[88] Keras Team. *Keras Documentation: About Keras.* URL: `https://keras.io/about/` (visited on 05/15/2022).

[89] *Temperatur - varme og kulde på jobben.* URL: `https://www.arbeidstilsynet.no/tema/temperatur/` (visited on 05/21/2022).

[90] *TensorFlow.* TensorFlow. URL: `https://www.tensorflow.org/` (visited on 05/15/2022).

[91] Robert Tibshirani. “Regression Shrinkage and Selection via the Lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 0035-9246. JSTOR: `2346178`.

[92] *Tuning Parameters — GEKKO 1.0.4 Documentation.* URL: https://gekko.readthedocs.io/en/latest/tuning_params.html#status-1 (visited on 05/20/2022).

[93] Saurabh Vaidya, Prashant Ambad, and Santosh Bhosle. "Industry 4.0 – A Glimpse". In: *Procedia Manufacturing.* 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA 20 (Jan. 1, 2018), pp. 233–238. ISSN: 2351-9789. DOI: 10.1016/j.promfg.2018.02.034. URL: https://www.sciencedirect.com/science/article/pii/S2351978918300672 (visited on 02/24/2022).

[94] D. J. Wagg et al. "Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications". In: *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg* 6.3 (Sept. 1, 2020), p. 030901. ISSN: 2332-9017, 2332-9025. DOI: 10.1115/1.4046739. URL: https://asmedigitalcollection.asme.org/risk/article/doi/10.1115/1.4046739/1081999/Digital-Twins-StateoftheArt-and-Future-Directions (visited on 02/24/2022).

[95] Z. Wang, T. Hong, and M.A. Piette. "Building Thermal Load Prediction through Shallow Machine Learning and Deep Learning". In: *Applied Energy* 263 (2020). ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2020.114683.

[96] Zeyu Wang and Ravi S. Srinivasan. "A Review of Artificial Intelligence Based Building Energy Use Prediction: Contrasting the Capabilities of Single and Ensemble Prediction Models". In: *Renewable and Sustainable Energy Reviews* 75 (Aug. 1, 2017), pp. 796–808. ISSN: 1364-0321. DOI: 10.1016/j.rser.2016.10.079. URL: https://www.sciencedirect.com/science/article/pii/S1364032116307420 (visited on 02/25/2022).

[97] *What Is Microsoft Azure and How Does It Work?* SearchCloudComputing. URL: https://www.techtarget.com/searchcloudcomputing/definition/Windows-Azure (visited on 02/07/2022).

[98] *XGBoost Documentation — Xgboost 1.6.1 Documentation.* URL: https://xgboost.readthedocs.io/en/stable/ (visited on 05/11/2022).

[99] Lu Yan and Meng Liu. "A Simplified Prediction Model for Energy Use of Air Conditioner in Residential Buildings Based on Monitoring Data from the Cloud Platform". In: *Sustainable Cities and Society* 60 (Sept. 1, 2020), p. 102194. ISSN: 2210-6707. DOI: 10.1016/j.scs.2020.102194. URL: https://www.sciencedirect.com/science/article/pii/S2210670720301815 (visited on 05/20/2022).

[100] Ibrar Yaqoob et al. "Big Data: From Beginning to Future". In: *International Journal of Information Management* 36 (6, Part B Dec. 1, 2016), pp. 1231–1247. ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2016.07.009. URL: https://www.sciencedirect.com/science/article/pii/S0268401216304753 (visited on 02/24/2022).

[101] Kaile Zhou and Shanlin Yang. "5.11 Smart Energy Management". In: *Comprehensive Energy Systems.* Ed. by Ibrahim Dincer. Oxford: Elsevier, Jan. 1, 2018, pp. 423–456. ISBN: 978-0-12-814925-6. DOI: 10.1016/B978-0-12-809597-3.00525-3. URL: https://www.sciencedirect.com/science/article/pii/B9780128095973005253 (visited on 02/07/2022).